Software Defined Radio for Communications

Ladimer S. Nagurney
Department of Electrical and Computer Engineering
University of Hartford
West Hartford, CT, USA
nagurney@hartford.edu

***Abstract*—Software Defined Radio (SDR) has evolved in the past decade from a topic that was only briefly mentioned in passing in graduate special topics courses to one that is now accessible to students at all levels. The rapid proliferation of cost effective SDR hardware has allowed not only the development of undergraduate SDR communication labs, but affordable test instrumentation that may be used in a variety of courses. This paper will outline a series of hardware choices and exercises that make SDR not just a topic for advanced communications, but a topic that can enhance undergraduate and graduate courses in communications, analog and digital signal processing, and instrumentation.**

***Index Terms*—Software Defined Radio, Digital Signal Processing, Communications Systems**

I. Introduction

Software Defined Radio (SDR) is now pervasive in the design of many communications systems. Use of SDR makes it straightforward to implement a wide variety of systems and, more importantly, by use of universal platforms these systems can be rapidly reconfigured to use different types of signaling, modulation, and frequency ranges. SDR allows new techniques to be rapidly implemented. From a pedagogical point-of-view, SDR techniques not only reinforce the classroom learning, but because systems may be implemented, the students can use these systems. In addition, the SDR devices themselves can be use as instrumentation to illustrate various topics discussed in the classroom. Another key feature of SDR is that it links topics discussed in DSP and other signal processing courses to real effects, again, very useful for student understanding. This paper will outline a series of hardware choices and exercises that make SDR not just a topic for advanced communications courses, but can enhance undergraduate and graduate courses in communications, analog and digital signal processing, and instrumentation. The focus is mainly on analog communications, since this topic is easily appreciated by the students.

Earlier discussions of using SDR in the classroom have appeared in [1]–[3].

Software defined radio hardware/software can be divided in 3 classes, based upon the order of magnitude in dollars of its cost. There are a number of SDR radio systems that cost on the order of $\$10^1$. These will be described as devices that can be used to wet the students appetites to SDR and provide useful lab equipment. Coupled with these is a plethora of freely downloadable software packages that can be used and extensions to more sophisticated packages like MATLAB. This class also includes DSP processor boards that can generate/detect baseband I and Q signals.

The next class, costing roughly $\$10^2$ per unit, includes several open source and vendor demonstration packages. These can be used to develop full-fledged SDRs.

The final class, costing at least $\$10^3$ per unit, includes high end laboratory devices, PC Cards, and complete turn-key SDR receivers and transceivers.

This paper is organized as follows. Section II provides a very brief mathematical overview of the mathematical of SDR systems. Section III describes the $\$10^1$ SDR systems, Section IV describes the $\$10^2$ systems, and Section V outlines the $\$10^3$ options. In each section, examples of student projects and reactions by the students to SDR use over the past several semesters.

II. Background for Software Defined Radio

The mathematical analysis of analog communications has been long based on modeling the transmitted and received signals as bandpass signals as outlined in texts such as [4]. A transmitted signal, $s(t)$, can be represented as a *Bandpass Signal*,

$$s(t) = \text{Re}\left\{g(t)e^{j\omega_c t}\right\} \qquad 1$$

where ω_c is the carrier frequency and $g(t) = g[m(t)]$ is a function that determines the type of modulation. The generalized transmitter implementation of an SDR is based upon an expansion of $s(t)$ into the sum of real quantities

$$s(t) = x(t)\cos\omega_c t \; + \; y(t)\sin\omega_c t \qquad 2$$

where $x(t)$ and $y(t)$ are the inphase and quadrature components of $g(t)$, respectively, often referred to as I and Q.

For detection, the received signal $r(t)$ is multiplied by $\cos\omega_c t$ and $\sin\omega_c t$ to recover the inphase and quadrature components of the $g(t)$, respectively.

In traditional communications courses, the analysis diverges here, since the implementation of the systems using discrete analog components does not directly map into this mathematical framework. This framework, however, leads directly to implementations of modulation and demodulation using SDR.

A naive implementation of a communication system using SDR would be to choose a sampling frequency that is at least twice f_c. By sampling the modulating signal, $m(t)$, at this rate all processing can be performed mathematically. As a

Funded by a Vincent Coffin Grant from the University of Hartford

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

result, only one High-speed D/A converter would be needed for the transmitter and only one High-speed A/D converter would be needed for the receiver. While straightforward, this approach results in many complications not germain to this work especially regarding the cost and availability of high speed converters and the design of sampling rate converters to convert the much slower varying $m(t)$ to the sampling rate of the A/D or D/A converters.

To counter these complexities, most SDR implementations use analog quadrature modulation and demodulation to generate and detect the final transmitted signal. That is, on transmit, the sampled m(t) is converted to a a sampled I and Q. A pair of D/A converters, operating at a much lower rate, are used to convert these sampled signals to analog $I(t)$ and $Q(t)$.. Equation (2) is then implemented by an Analog Quadrature Modulator. A receiver is implemented using the inverse of this.

III. Options for $\$10^1$

In this section, hardware/software options and associated student exercises that can be implemented for on the order of $\$10^1$ per student are discussed.

A. Software only approach

Notwithstanding the caveat in the previous section, there are software only approaches to SDR that are illustrative for student use. There is an informal industry standard that downconverts the signal to a carrier frequency of 12 KHz as the lowest analog IF frequency. This limits the spectrum of $m(t)$ to less than 6 KHz which is more than suitable for a voice-grade analog audio modulating signal. It is straightforward to use an A/D converter to sample this signal. If one assumes a sampling rate of 48 KHz, which is 4 times the Nyquist frequency, the quadrature detector may be implemented mathematically by multiplying by repeating strings containing $1,\ 0,\ -1\ 0$. The processing power of most computers is sufficient to implement various detection algorithms using almost any computer language including, C, MATLAB, Java, Python, etc. The main limitation is that the choice of lanuage has to support the necessary A/D and D/A converters.

To use this approach, it is not even necessary to procure a receiver that has the 12 KHz IF output. Many older receivers have a Panoramic or Spectrum Analyzer output that routes the IF to a connector on the rear panel. A downconverter consisting of a double balanced mixer, whose Local Oscillator is 12 KHz away from the receiver's IF frequency can be used to obtain the 12 KHz centered output. If the receiver does not have a panoramic output, the downconverter can be connected to the IF stage of the receiver using a high impedance amplifier to obtain this signal. Using this method, the students can easily turn the receiver volume down and compare their SDR detectors to the receiver's analog detectors. Even without a receiver, most Arbitrary Waveform Generators will generate analog AM, FM, and PM modulated signals at this 12 KHz carrier frequency allowing the students to easily demonstrate their efforts at creating a receiver.

A complementary approach to this is to use an inexpensive DSP evaluation board, such as one using the TI 5515 or 5535 processors. Each of these TI boards is programmed in C via the freely available Code Composer. Since the boards are designed for audio processing, the I/O is a stereo codec which can sample at 48 KHz. With some cleverness and breakout boxes, since the evaluation board can perform all the I/Q processing in real time, it is possible to directly implement a complete AM or FM transmitter on the board.

As another example of this software only approach, consider a student project to develop an AM Transmitter and Receiver in MATLAB. The students are first asked to create 1 second of 1 KHz cosine wave sampled at 48 KHz as an array, m. A carrier sequence, c, of repeated $[\ 1\ 0\ \ -1\ 0]$ is also created. An AM signal with any modulation index, mi, may then be created via the MATLAB statement:

```
s = (1 + mi*m) .* c;
```

The resulting transmitted AM signal is shown in Fig. 1.

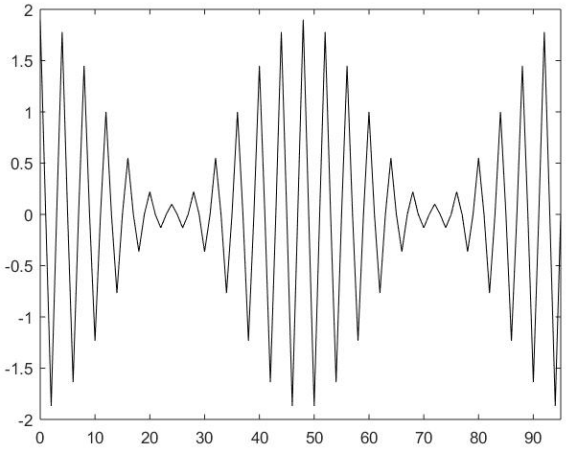

Fig. 1. Transmitted AM Signal

There are several options for detecting this AM signal. The students might begin implementing a half-wave diode detector, that removes the negative excursions of the signal. Following the detector, a crude downsampler can be inserted to convert the detected signal from a 48 KHz sampling frequency to one sampled at 12 KHz. The output of this exercise is depicted in Fig 2.

Another detector that can be simply implemented is a full wave, absolute value detector using the `abs` function. If the output of this detector is the array ra, using a relatively simple LP filter such as

$$ro(n) = .25ra(n) + .5ra(n-1) + .25ra(n-1)$$

produces an output illustrated in Fig 3

By normalizing the detected signal to a maximum value of 1, the MATLAB `sound` function can be used by the students to hear the detected signal. Occasionally, in this exercise the

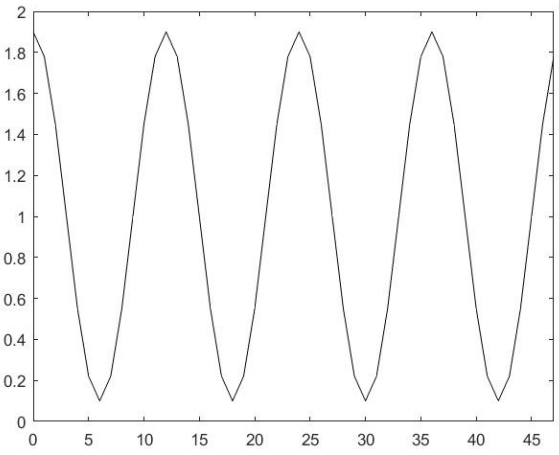

Fig. 2. Detected AM Signal

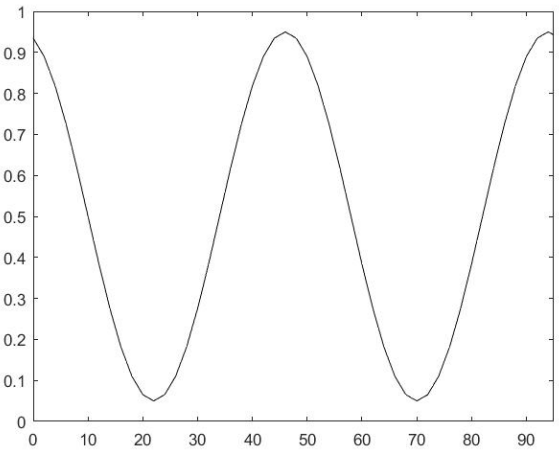

Fig. 3. Detected AM Signal using an Absolute Value Detector

sampling rate of the detected signal has to be adjusted to use the sampling rate of the computer's soundcard.

A variation of this exercise can include a non-zero threshold for the diode detector. The studnets can then investigate the interrelationship between signal level, threshold, and modulation index.

B. Softrock Approach

The publication of [5]–[8] outlined a miriad of possibilities for SDR using relatively simple RF circuitry and a PC with the I/Q signals being sampled by the soundcard. The Tayloe Detector [9] can be used with a oscillator that is $4\times$ the signal frequency to produce analog I/Q outputs that can then be sampled and detected. Typical projects are documented on [10]. A version can be constructed using non-SMT components, although many students have used these ideas to designa and build circuit boards. Some previous examples of this were described in [1].

C. RTL-SDR Approach

Probably the simplest method to illustrate a wide range of SDR is using the so-called RTL-SDR dongles that are available from a variety of sources for under $30. These dongles are based upon the identification that European DVB-T dongles, when used with alternative driver software, can be programmed to output the I and Q baseband signals and continuously tune from approximately 25 MHz to 1.5 GHz. [11]. This means that the dongles can not only be used as DVB-T receivers, but can be used as the RF portion and I/Q detector for receiver and instrumentation systems.

A variety of software is available for these dongles for Windows, Mac, and Linux [12]. The software also includes packages to use these dongles as spectrum analyzers. This allows the students to view a wider variety of communications systems in real life including traditional radio, digital TV and cellular systems.

Fig 4 illustrates the reception of an FM radio station using SDR# which is described in more detail in the next section. From the top part of the figure the students can clearly see not only the spectrum of the FM signal, but the HD radio sidebands. Looking at the FM Multiplex spectrum, the students can easily identify the L+R signal, the stereo pilot carrier, the L-R signal and the 57 KHz centered RDS signal that provides the short titles on RDS-equiped FM radios.

The students have been encouraged to investigate other signals using these dongles. The low end of Channel 30 DTV signal is illustrated in Fig 5. Note that the pilot carrier is clearly visible .31 MHz above the bottom of the channel.

The spectrum of LTE signals is also interesting to investigate. In Fig 6 the Band 13 LTE spectrum, used by Verizon, is illustrated. Note that the individual carries can be clearly distinguished.

The above examples were created using SDR# which is described in more detail in the next section. For stundents using MACs, the program GQRX [13] is freely available.

Students experimenting with the dongles in both our Analog, Communications Theory, and SDR courses have reported significant increases in their understanding of communications.

For more detailed programming and and analysis, these RTL-SDR dongles can also be progammed in MATLAB and/or SIMULINK [14]. Using MATLAB and/or SIMULINK, the students have signifincantly more control over dongles and the processing of the I/Q bitstreams than they would have using any of the precompiled programs. The support package can be downloaded from Mathworks that adds a number of functions to the Communications System Toolbox. The SIMULINK block to receive data from the dongle illustrated in Fig 7.

In general, using MATLAB, blocks of I/Q samples are received from the dongle. The blocks may be processed, to either demodulate the signal or to display its spectrum. Using MATLAB it is possible to convert the dongle to a spectrum analyzer with a broader bandwidth than can be received using a dedicated program such as SDR#.

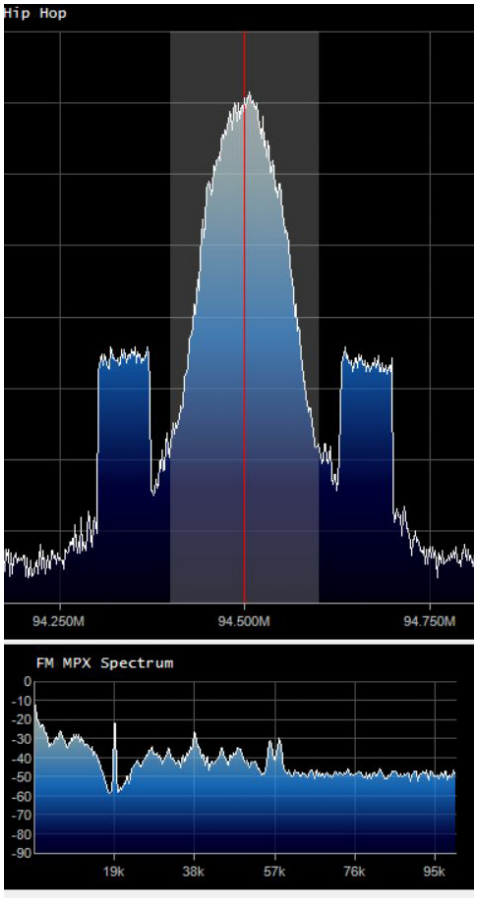

Fig. 4. RF and Baseband Spectrum of an FM Radio Station

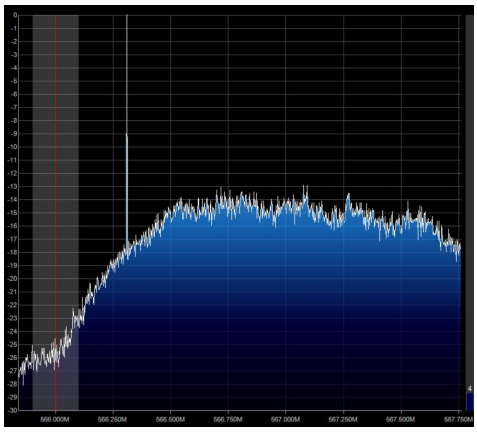

Fig. 5. Spectrum of a DTV Station

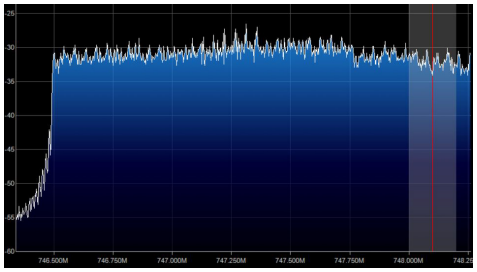

Fig. 6. Spectrum of Band 13 LTE

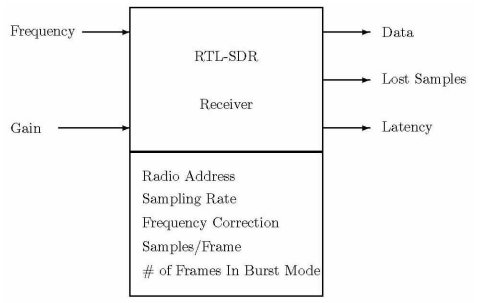

Fig. 7. Simulink Block for an RTL-SDR

D. SDR#

Probably the least inexpensive way to illustrate SDR techniques is through the use of the SDR# [15] software package. SRD# is public-domain package, that not only can control the dongle described in the previous sections, but, more importantly, has a host of other features that can be used to investigate SDR techniques. While SDR# can take processed I/Q samples from other receivers, it can also take analog I and Q signals sampled via the computer's soundcard as inputs. Thus, the package can be used with from very simple to very complex I/Q detectors to implement SDR.

On feature of SDR# that can be exploited to enhance student understanding of Software Defined Radio is it's ability to record the I and Q signals as a .wav file. If one is using SDR# to receive a signal, the I/Q samples may be saved. These .wav files can be played back by SDR# at a later time. Since the .wav files are just the raw I/Q samples, during playback, all options of SDR# such as the center frequency, bandwidth and type of detector may be changed/adjusted to illustrate various phenomina and enhance characteristics of the signal.

In the classroom, this feature provides a way to demonstrate various modes and illustrate the differences without having to set up a receiver dongle and hope that the signal of interest is on the air. One particularly interesting demo for the students is the generation of a DSB-SC signal. Mathematically the students show that the detected signal is a single tone only if the transmit and receive carrier frequencies are identical. Should the carrier frequencies differ, the single tone becomes a pair of tones shifted by the difference of the carrier frequencies.

The ability to read the I/Q samples as a .wav file also leds to a variety of student exercises. For example, after recording

a .wav file of a on-the-air signal, the students can first use the MATLAB function, `audioinfo('fn.ft')` to investigate the parameters of the .wav file. Analyzing an SDR# recorded .wav files shows that the sampling rate is 2400000 bits/sec, 16 bits/sample are used, and that there are 2 uncompressed channels. Knowing these parameters, the student may now generate his/her own I/Q .wav file.

The students were then asked to create an $n \times 2$ array of samples of the I/Q signals using MATLAB. This array is then converted to a .wav file using `audiowrite`. Because the sampling frequency is 2.4 MBps, the students will have a wide choice of carrier and modulating frequencies. For example, an AM signal can be generated with a carrier frequency of 500 KHz and modulating frequency of 1 KHz. If the students defines the I samples as the generated AM array and sets the Q samples to 0, the .wav file, when processed by SDR# appears as in the top image in Fig 8.

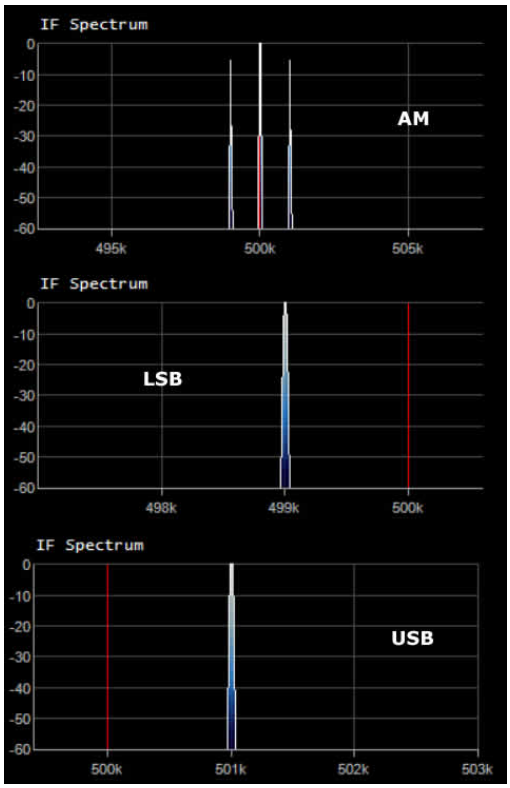

Fig. 8. I/Q signals generated by MATLAB detected by SDR#

Since the I and Q samples are the received samples, another type of I and Q arrays can be created as

$$I = s(n).*\cos\left(2\pi\frac{f_c}{f_s}n\right) \quad \text{and} \quad Q = s(n).*\sin\left(2\pi\frac{f_c}{f_s}n\right)$$

If $s(n)$ is chosed to be the single tone

$$s(n) = \cos\left(2\pi\frac{f_c + f_m}{f_s}n\right)$$

the signal detected as a LSB or USB is illustrated in the lower images of Fig 8.

The students can experiment further to discover a multitude of properties of Communications Systems. For example, it is easy to illustrate that an AM detector can be used to detect a NBFM signal and show how an NBFM detector responds as it's bandwidth is varied. By further experimentation, the students can determine that SDR# will process I/Q .wav files at lower sampling rates than 2.4MBps, adding further options for the exercises.

All students who have tried these exercises were convinced that these have significant educational value.

IV. Options for $10^2

At $10^2 per unit, the hardward can be divided into 2 classes. The first class includes mostly crowd-funded open-sourced projects that have been developed to highlight specific chips and frequency ranges. Many of these boards were developed for IOT applications, for ways to connect smart devices with the phone network, or even for small cells, themselves. All work with GNURadio [16], however most have their own software and drivers. In addition, most are supported through Linux. The second class is an educational product developed by Analog Devices.

A. Open sourced boards

- Lime SDR [17] is one of a family of SDR transcievers that use the Lime Microsystems transceivers. The Lime RF transceivers were initially developed as programmable RF transceivers for cellular, IOT, Zigbee, and Bluetooth designs. It is easy to construct an RAN for cellular systems using a Lime Transceiver and a specific version, LimeNET has been created for this application.
  The LimeSDR has a broad frequency range, from 100 KHz to 3.8 GHz and a bandwidth of 61.44 MHz making it useful for very high datarate applications.
- bladeRF [18] is a transceiver from 300 MHz to 3GHz using the LimeMicro LMS6002D integrated RF transceiver. There is a companion LF/MF/HF/VHF transverter that extends its frequency range down to 60 KHz. It is designed to be programmed using GNURadio and is supported by Linux, Windows and MAC. It's interface is USB 3.0.
- Hack RF One [19] is a 1 MHz to 6 GHz operating frequency half-duplex transceiver up to 20 million 8-bit I & Q samples per second. While often programmed via GNURadio, it also is supported by SDR#.

B. Analog Devices Pluto

In late 2017, Analog Devices introduced its SDR, Pluto, as part of its University Program [20]. It is based on the Analog Devices AD9363 transceiver and an FPGA and can be used from roughly 100 MHz to 4.0 GHz. It is designed to be programmed in MATLAB/SIMULINK and GNURadio, but also has support for C and Python. Because of its recent introduction, it has not been used yet by our students.

It has also been recently reported that a PLUTO has been used as an exciter for a digital TV Transmitter [21].

Each of the hardware devices at this price range have a relatively steep learning curve and, while useful for demonstrations, may not be thoroughly useful for student use. Without a dedicated lab and lots of student and instructor enthusaism, these devices may not been as useful as other choices.

V. Options for 10^3

At this price range for implementing Software Defined Radio, there are three general possibilites for inplementation, using an USRP, a Commercial Receiver/Transmitter, or Dedicated High Speed Converters with Digital Up and Down Converters.

A. Universal Software Radio Peripherals

The Gold Standard for SDR systems is the Universal Software Radio Peripheral (USRP) series developed by Ettus Research [22], now part of National Instruments. On transmit, USRPs take the I/Q streams, interpolate them to a higher rate, convert them to analog signals and then use an I/Q modulator to generate the final transmitted output. On receive, they use an I/Q detector to generate the I/Q signals which are then converted to digital streams and decimate them to a rate the can be transmitted to the host computer. The USRPs often contain multiple channels that may be configured separately. For example, one receive channel might be a transceiver whie the other channel is a GPS receiver to provide accurate timing. The simplified block diagram for one channel of a USRP is shown in Fig 9.

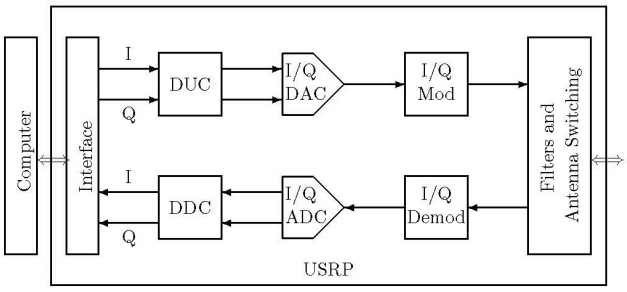

Fig. 9. Simplified Block Diagram of a USRP

Programming USRPs can be done by either gnuRadio [16], MATLAB/Simulink, or LabView Communications [23]. In addition, because the UHD drivers are open, the USRPs can be programmed in C, C++, or Python. There is extensive documentation and examples using LabVIEW Communications including a workbook with multiple worked examples. [24]. Students have little difficulty using the examples and programming their own algorithms.

An example of the programming of an FM transmitter using Labview Communications is shown in Fig 10 and 11. In this example, the program first generates a complex array representing

$$g(t) = e^{j\alpha m(t)}$$

This complex array is then fed into the `NI_USRP` vi which instructs the USRP to transmit the FM signal. The modulating frequency and the modulation index, α, may be changed during operation.

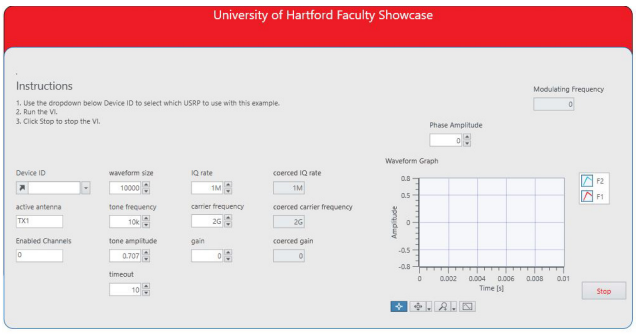

Fig. 10. Labview Communications Front Panel for an FM Transmitter Demo

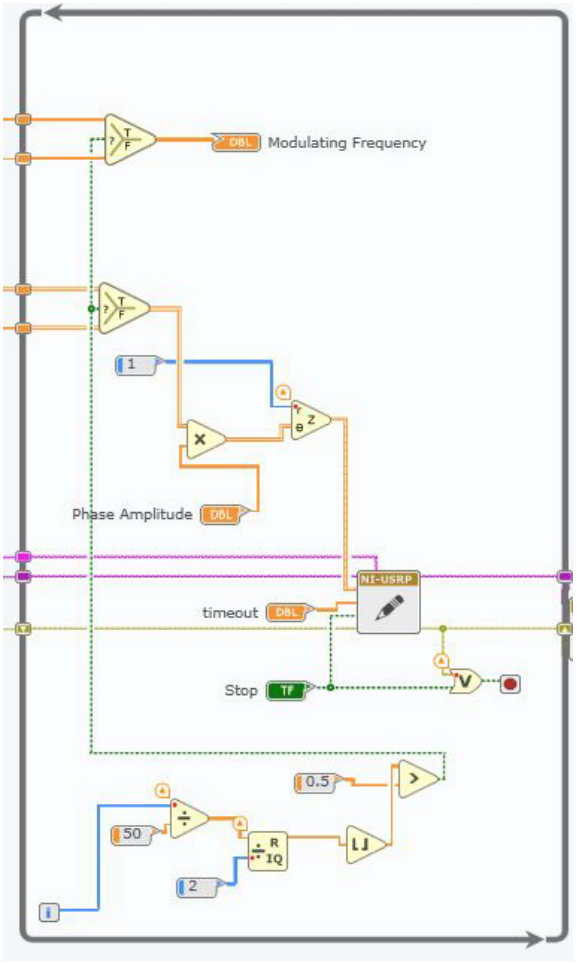

Fig. 11. Labview Communications Loop to Transmit FM

This demo was very useful, since the USRP could transmit in the frequency range of an FM broadcast receiver. Hearing the signal on a generic FM radio convinced the students that the USRP was transmitting a real radio signal.

For a demonstration of SDR at a faculty showcase, the transmitting USRP was modulated with alternating tones. A second computer was setup with an RTL-SDR receiving the signal and displaying the spectrum. A low-end protable FM radio was also used to receive the transmitted signal to *prove* that real radio signals were being transmitted. When the same demonstrated was performed in the communications lab, a Spectrum Analyzer and an ICOM R-7000 receiver were also used. The R-7000 illustrated how, as the modulation index was increased, the detected signal level first increased and then decreased using a NBFM detector.

B. Commercial Options

While Software Defined Radio is used daily in wireless and broadband markets, there are a number of SDR radio systems that are designed primarily for the analog HF communications market. While some are marketed primarily for the military and have exorbitant costs, others transceivers and receivers are marketed primarily to the commerical and amateur radio market and are in the $\$10^3$ cost range.

The market can be divided into 2 segments. SDRs that are basically just boxes with ethernet/USB connections that require a computer with display as their front panel and integrated SDR receivers/transceivers with front panels that appear to be normal radio systems.

Examples of the first segment has SDRs from FlexRadio [25], Elad Receivers [26], and Apache Labs [27]. Each of these use an Ethernet or USB connection to connect to the computer. Most of these use a thin rack-mount form factor. In general they cover from 1.8 to 30 MHz and segments near 50 MHz, 70 MHz, and 144 MHz. They are driven by dedicated software package, often provided by the manufacturer.

The second segment is led by ICOM [28] but also includes devices by Elad [26] and, more recently, FlexRadio who have all introduced a SDR radio system with a traditional front panel. Since, these receivers and transceivers have front panels and the user is unaware of the internal SDR technology. While most can be used and controlled by a computer, using a computer is not necessary.

In both these segments, SDR technology is used, but it is impossible for the devices to be reporgrammed by the user. However, a wide variety of application software can be run to implement various modes of communications. In addition, the ethernet controllable systems can be used as remote stations to take advantage of antenna and propagation characteristics not available at the controlling point.

C. Very High End Options

For completeness, I would like to include two alternatives for very high end systems, often costing over $\$10^4$ per system.

There is a wide variety of very high speed AD/DA cards with internal FPGAs for processing by firms such as Pentek [29]. The cards are just FPGAs and Converters, a subset of the USRP block diagram illustrated in Fig 9. While the learning curve is extremely long, these boards have been used for quite specific SDRs. Pentek has produced two handbooks that are available for free and thoroughly describe SDR [30], [31].

A final high-end SDR whose list price is on the order of $\$25,000$ is the Zurich Instruments Lock-in Amplifier [32]. The Lock-in can be configured as an AM or FM receiver for carrier frequencies up to 600 MHz.

VI. Summary

This paper outlines a variety of options for teaching Software Defined Radio, either as a separate course or as part of a Communications Engineering course. Hardware options over a wide cost range were discussed and examples of student exercises were illustrated.

Acknowledgment

The author would like to thank the ananoymous reveiwers and program committee for their constructive comments.

He would also like to thank his students in Digital Signal Processing, Communication Engineering, Communication Theory, and Software Defined Radio courses for performing the exercises and critiquing his demonstrations.

References

[1] L.S. Nagurney, Software Defined Radio in the Electrical and Computer Engineering Curriculum, Proceedings of FIE 2009, IEEE, San Antonio, TX, (October 2009)

[2] C.J. Prust, S. Holland, and R.W. Kelnhofer, Ultra Low-Cost Software-Defined Radio: A Mobile Studio for Teaching Digital Signal Processing, 121st ASEE Annual Conference and Exposition, Indinapolis, IN, 2014, Paper ID #10633.

[3] M. Bazdresch, Considerations for the Design of a Hands-on Wireless Communications Graduate Course Based on Software-Defined Radio, Proceedings of FIE 2016, Paper F1I-1.

[4] L. W. Couch II, **Digital and Analog Communication Systems** 8^{th} Edition, Pearson - Prentice Hall, Englewood Cliffs, NJ, 2013.

[5] G. Youngblood, Software-Defined Radio for the Masses: Part 1, QEX July/August 2002, pp. 13-21.

[6] G. Youngblood, Software-Defined Radio for the Masses: Part 2, QEX September/October 2002, pp. 10-18.

[7] G. Youngblood, Software-Defined Radio for the Masses: Part 3, QEX November/December 2002, pp. 27-36.

[8] G. Youngblood, Software-Defined Radio for the Masses: Part 4, QEX March/April 2003, pp. 20-31.

[9] D. Tayloe, A low-noise, high-performance zero IF quadrature detector/preamplifier. RF Design, March 2003, Available from: http://rfdesign.com/ar/radio_lownoise_highperformance_zero/index.htm.

[10] http://www.wb5rvz.org/

[11] C. Laufer, **The Hobbyist's Guide to the RTL-SDR** 2^{nd} Print Edition, Amazon, 2014.

[12] http://RTL-SDR.com

[13] http://www.gqrz.dk

[14] R.W. Stewart, K.W. Barlee, D.S.W. Atkinson, & L.H. Crokett, **Software Defined Radio using MATLAB™ & Simulink™ and the RTL-SDR**, Strathclyde Academic Media, Glasgow, Scotland, 2015.

[15] http://airspycom/download/

[16] https://gnuradio.org/

[17] http://www.limemicro.com

[18] http://www.nuand.com

[19] https://greatscottgadgets.com/hackrf/

[20] http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html#eb-overview

[21] D. Crump, ATV, RadCom, July 2018, pp 38-39.

[22] https://www.ettus.com/

[23] http://www.ni.com/labview-communications/

[24] B.A. Black, **Introduction to Communications Systems: Lab Based Learning with NI USRP™ and LabVIEW Communications**, National Instruments, Austin, TX, 2015.

[25] http://www.flexradio.com
[26] http://www.elad-usa.com
[27] http://apache-labs.com
[28] http://www.icomamerica.com/en/amateur/
[29] https://www.pentek.com/
[30] R.H. Hosking, **Software-Defined Radio Handbook**, Pentek, Inc., Upper Saddle River, New Jersey, 13th Edition, June 2017.
[31] R.H. Hosking, **Putting FPGAs to Work in Software Radio Systems**, Pentek, Inc., Upper Saddle River, New Jersey, 11th Edition, June 2017.
[32] http://zhinst.com