New Orleans JAZZED about Engineering Education

Paper ID #16264

Software Defined Radio: Choosing the Right System for Your Communications Course

Mr. Kurt VonEhr, Grand Valley State University

Kurt VonEhr graduated from Grand Valley State University with a B.S.E.E. and minor in Computer Engineering. He is currently attending Oakland University in Rochester, Michigan for a M.S. in Embedded Systems. Kurt's engineering interests include: Embedded System Design, Digital Signal Processing, Communication Theory, Software Defined Radios, Sustainability and Alternative Energy.

William Neuson, Grand Valley State University

William Neuson is an undergraduate student at Grand Valley State University, pursuing a degree in Computer Engineering. He has a deep-seated interest in software defined radio due to its proximity to both software engineering and amateur radio.

Dr. Bruce E. Dunne, Grand Valley State University

Bruce E. Dunne received the B.S.E.E. (with honors) and M.S. degrees from the University of Illinois at Urbana-Champaign in 1985 and 1988, respectively, both in Electrical and Computer Engineering. He received the Ph.D. degree in Electrical Engineering from the Illinois Institute of Technology, Chicago, in 2003. In the Fall of 2003, he joined the Padnos College of Engineering and Computing, Grand Valley State University, Grand Rapids, MI, where he is currently Professor of Engineering and Chair of the Electrical Engineering Program. Prior to this appointment, he held several research and development positions in industry. From 1991 to 2002, he was a Staff Engineer with Tellabs, Naperville, IL. Additionally, in 1991, he was with AT&T Bell Telephone Laboratories, Naperville; from 1988 to 1991, he was with R. R. Donnelley & Sons, Lisle, IL; and from 1985 to 1986, he was with Zenith Electronics, Glenview, IL. His interests include adaptive filtering, speech enhancement, wireless and wireline communications, and engineering education. Dr. Dunne is a senior member of the IEEE and a member of Eta Kappa Nu and the ASEE.

Software Defined Radio: Choosing the Right System for Your Communications Course

Kurt VonEhr, William Neuson and Bruce E. Dunne
School of Engineering, Grand Valley State University

Abstract

Software Defined Radio (SDR) has recently been popularized as a powerful and full-featured alternative to delivering instruction in the area of analog and digital communications. Fortunately, there is a wide array of hardware to support SDR instruction, spanning a range of capabilities as well as price. Such higher-capable systems include the networked series of the Universal Software Radio Peripheral (USRP) platform that allow for complete stand-alone radio systems, able to acquire and process large portions of the RF spectrum. A mid-point system is the HackRF One, with both receive and transmit capability, sample rates of up to 20 MS/sec, and operating up to 6 GHz. At a very modest price but with surprising capability are systems such as the RealTek RTL2832U stick, sampling up to 2.4 MS/sec, operating up to 2 GHz (where these systems double their usable spectrum through the use of Complex Sampling). Software support for these systems includes the ability to write custom routines in various programming languages such as C++ or Python or the option of using the GNU Radio signal processing package to link the routines to the hardware. Of particular interest for instructional purposes is the use of graphical development tools such as GNU Radio Companion (GRC) or MATLAB Simulink to allow students to configure and link communication blocks to create communication systems. Additionally, open source modules are available to seamlessly and easily connect these communication system flow graphs to many compatible hardware devices; with a low-cost antenna, students are transmitting and receiving communication signals while examining their characteristics on standard laboratory test equipment. Additionally, powerful and easy to use analysis tools such as SDR# augment the experience.

In this paper, we describe and compare the features, cost and capabilities of several of the more popular SDR systems typically used for instructional purposes. We further discuss how these systems are configured and programmed with several of the more popular software programs. We consider such factors as ease of use, cost and features. In short, our goal is to provide other educators with a "quick-start" guide to implementing SDR in their communications course.

As these tools have been used for communications instruction at our university, we describe several of the more interesting laboratory exercises. These include transmission and reception of signals for both analog and digital communications systems. Finally, we include survey results demonstrating our students' perceptions in comparing SDR-based instruction to more conventional methods.

Introduction

Software Defined Radio[1,2] (SDR) offers a powerful alternative to conventional communication system design. In conventional design, specially-built hardware is implemented to perform communication for a particular, radio-specific modulation scheme, usually over a limited frequency range. In contrast, SDR systems offer much more flexibility by implementing the modulation/demodulation functionality in software. Connected to the antenna through an RF mixer is a highspeed ADC/DAC (for receiver/transmitter, respectively) such that the SDR processes the communication signals using DSP algorithms implemented in software. Depending on sampling rates, large segments of the spectrum can be manipulated for a wide variety of simultaneous processing. Particularly powerful is the concept of flexibility; if the radio modulation scheme changes, new DSP software is loaded to perform the necessary processing and no hardware modification is required. This approach allows for ease of adaptability, shortens development effort and greatly reduces cost and complexity. The generic architecture of an SDR system is shown below in Figure 1.

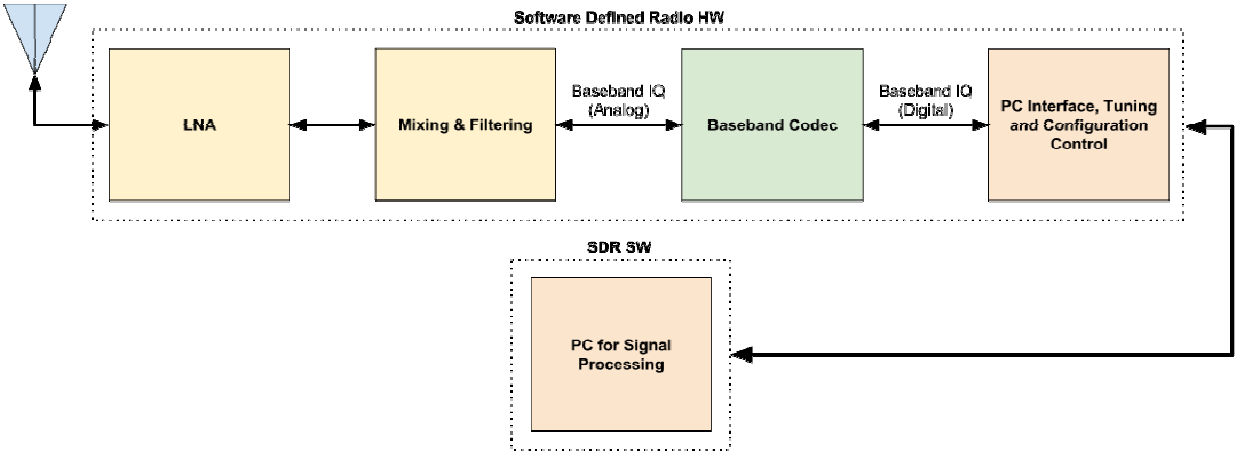

Figure 1: General SDR Architecture

The advantages of using SDR in commercial applications also applies to its use in an academic laboratory setting. Many recent papers describe the use of SDR systems for academic instructional purposes. These include various analog and digital communications experiments and projects[3,4,5,6] based on the popular Universal Software Radio Peripheral (USRP) platform (available from Ettus Research/National Instruments[7]), implementing the communication algorithms using GNU Radio[8] software suite or with software such as LabVIEW[9]. A recent approach[10] implements SDR-based instruction using the ultra low-cost RTL-SDR dongle, again using GNU Radio. Finally, other approaches[11,12,13] adapt various hardware platforms and software to perform the SDR functions.

We begin by discussing the range of hardware available for SDR classroom use, comparing features and cost. Next, a discussion of software for signal analysis and communication algorithm implementation is included. We then describe a range of analog and digital communications laboratories that are easily duplicated with the suggested hardware and

software. Lastly, we present student observations on the use of these SDR systems in a recent introductory course on analog and digital communications.

SDR Hardware

In this section, we consider a range of purpose-built SDR systems that are most easily adapted to classroom laboratory use. One way to classify these systems is by their capabilities (and associated price); more or less, we see there being three ranges of these device performance, described below. One very important concept to keep in mind when considering the performance of an SDR, is the inherent inverse relationship of ADC / DAC resolution and sample rate[14] – a higher bandwidth does not always mean higher performance.

The high-end class of SDR systems are characterized by their wideband operation, full-duplex RX/TX, higher signal to noise ratio (relative to other SDRs), higher bandwidth host interface (USB 3.0, GigE) and modular hardware (swappable RF components). The intermediate class of SDRs feature some of the high-end characteristics, but not all of them. For example, these may only be capable of half-duplex communication or have lower quality RF and ADC/DAC components. Finally, the low-end SDR systems feature only a few of the high-end characteristics and typically function as RX only. Examples of SDRs that fall into each of these categories are described below.

Full-Featured Systems: The USRP

The high-end systems considered herein are the previously mentioned USRP[7] variants from Ettus Research. The USRP series are open-source and full-featured devices, including some models capable of stand-alone operation. All models function as transceivers, utilizing high performance DAC/ADC hardware that supports a wide spectral bandwidth when coupled with a specialized wideband RF front end daughterboard. It should be noted that high-Q, narrow-band daughterboards are also available for ISM bands.

The hardware architecture of the USRP includes a FPGA for digital up/down-conversion to/from the DAC/ADC for the IF signal to/from the RF daughterboard, respectively. This technique allows for the highest resolution ADC/DAC conversion between the baseband and IF signals. Shown below in Figure 2 is the generalized architecture of the USRP series. The FPGA can also be used to implement additional DSP functionality, however this may reduce FPGA digital sample rate conversion performance.

There are essentially three models of USRP devices. The first two are host-connected devices, differing on the rate of the communication link to the host, while the third device is a stand-alone device capable of operating without a host connection. The lowest-priced model is the USRP Bus Series, for which the single transceiver B200 model is below $750 and interfaces to the host through a USB 3.0 connection. The USRP Network Series is the next highest priced model, priced near $1500, and, in this case, communicates to the host via Gigabit Ethernet. The USRP Embedded Series includes stand-alone models, capable of full SDR functionality without host processing or connection (beyond configuration) such as the E110, near $1500. To deliver this embedded functionality, the E110 includes a TI OMAP processor[16] featuring an ARM Cortex A8 running at 800 MHz and a TI C64 DSP. The ARM processor runs Linux, with GNU Radio pre-

installed. Further, for each series, models with additional features (such as dual transceivers, increased processing capability) are available at a higher cost.

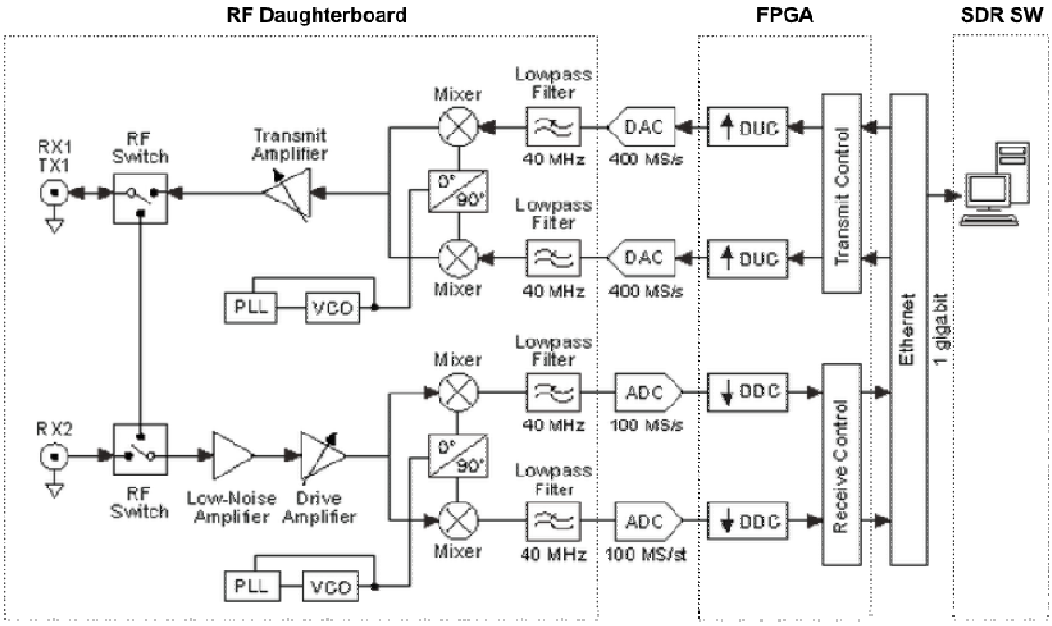

Figure 2: General USRP Architecture[15]

As delivered, all of the USRP systems must be augmented with one of the RF front end daughterboards for operation. ADC/DAC functionality is provided on the mainboard while these daughterboards provide the interface to the sampler, possibly including other RF processing such as frequency conversion and filtering. These daughterboards start with Basic-RX (for receive) and Basic-TX (for transmit), each below $100. As the name implies, these boards only offer an interface to the mainboard, potentially requiring external RF frontend hardware for most applications (RX may require filtering/gain; further, it supports a limited receive frequencies up to 250MHz and does not include downconversion or filtering; TX output is low-power at 1mW, supporting carrier frequencies up to 250MHz and providing a 50Ω output impedance SMA connection). For academic applications, either the SBX or WBX wide bandwidth transceivers are more appropriate, each priced just below $500. The SBX transceiver provides 40MHz of bandwidth over the frequency range of 400MHz – 4.4GHz, along with a transmitter with a 100mW output. The WBX provides similar capabilities but at the lower frequency range of 50MHz – 2.2GHz. These daughterboards include LO/mixer functionality, up/down-conversion, filtering and amplification.

Interpreting the usable bandwidth of these systems varies dependent upon the series model as well as the RF daughterboard. The RF front end daughterboards include downconversion and filtering to translate the desired spectral band center frequency to DC, with a frequency resolution equal to the sampling frequency. Note that the presented data is complex-valued (i.e., I/Q components represented by the real and imaginary parts, respectively). This so-called Complex Sampling representing data up to the sampling frequency, seems to violate the Nyquist Sampling Theorem; an explanation of this approach is discussed later in this paper as it is commonly used for many SDR systems. There are three factors to consider when determining

usable bandwidth: the Analog Bandwidth (based on the filtering characteristics of the chosen RF frontend daughterboard), the FPGA Processing Bandwidth (determined by the sampling rate of the FPGA on the chosen USRP series motherboard), and the Host Bandwidth (based on the maximum communication rate between the chosen USRP series and the host; i.e., e.g. USB, Gigabit Ethernet or to the embedded OMAP processor). The maximum bandwidth is therefore limited by the slowest throughput of these three elements. For example, the choice of the SBX RF front end teamed with the B200 USRP implies 40MHz of input bandwidth, an FPGA throughput capable of 28MS/s (when supporting both TX and RX) and a host maximum bandwidth similarly of 20MS/s (half-duplex) via the USB 3.0 interface. Hence, the host connection limits the maximum bandwidth to 20MHz of complex sampled I/Q data.

Mid-Range Systems

Mid-range systems, as with the full-featured systems, generally function as transceivers, with fairly wide sampling rates and spectral bands. However, these devices offer fewer choices in terms of RF frontend configurations and host interfaces. Furthermore, mid-range systems rely more on host processing and do not offer stand-alone systems. However, these devices still offer impressive SDR functionality and are fully compatible with the same tools as their more expensive counterparts at a substantially lower price point.

Examples of these systems include: the AD-FMCOMMS4-EBZ[17] (frequency range of 70MHz to 6GHz, 56MHz half-duplex bandwidth, USB/GigE host I/F, $399), the bladeRF[18] (frequency range of 300MHz to 3.8GHz, 28MHz full-duplex bandwidth, USB 3.0 host IF, $420) and the HackRF One[19] (frequency range of 1MHz to 6GHz, 20MHz half-duplex bandwidth, USB 2.0 host IF, $299).

We will describe the HackRF One system in more detail, as this is the system we have used in our laboratory environment. One important consideration is that the HackRF One can operate in either TX or RX mode, but cannot perform both simultaneously like the USRP (half-duplex versus full-duplex). The HackRF One architecture is illustrated below in Figure 3.

The HackRF One is an open source hardware platform that can be used as a USB peripheral or programmed for stand-alone operation. The schematics are available on the Github Repository[20], so that the architecture can be studied in great detail and improved upon if desired. The RF frontend features a programmable RX/TX gain and baseband filter. If more advanced applications (such as radar) are to be used, Clock In and Clock Out lines are provided on the HackRF One for timing synchronization across multiple devices.

Unlike the USRP, all RF and digital hardware components are located on the same PCB. This means that specialized RF filtering is not easily swapped in and out. This will reduce the overall signal to noise ratio for many applications, as this device is meant to be used as a general purpose, wideband SDR. However, it should be noted that custom built external front end filters is one commonly used technique to get the most out of a general purpose SDR. By placing the filter between the antenna and the SMA input of the SDR, reception can be greatly improved for the bands of interest.

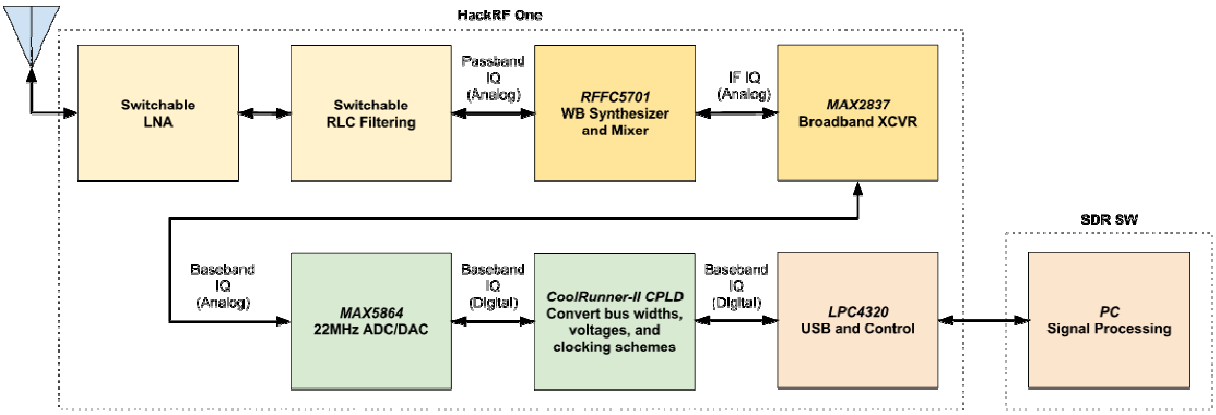

Figure 3: HackRF One Architecture[19]

Lower-Range Systems

Lower-range systems are an extremely cost-efficient SDR choice. Such systems typically function as receive-only devices, hence avoiding the cost/power-requirement needs of a transmit-capable system. Despite their low cost, these devices offer considerable SDR processing capability.

The RTL-SDR[21] dongle and the FUNcube Dongle[22] are two popular receive-only low-cost SDR units with significant online support communities. The RTL-SDR is based on the RTL2832U[23] IC. The RTL2332U IC is originally a European TV reception IC; employing a modified software driver, it is configured to function as a quite capable SDR (the RTL2832U chipset functions as the USB interface, ADC, filtering and demodulation). Additionally, the RTL-SDR dongle includes an RF frontend tuner IC such as the Elonics E4000, able to tune in signals from 24-1766MHz. The tuner outputs this signal at an Intermediate Frequency (IF) around 4.57MHz, which is then processed by the RTL IC. The RTL-SDR is extremely cost effective, with prices noted below $10 by some suppliers, which even includes an antenna capable of operating over the tunable range (we have found that reception is much improved if the magnet on the bottom of the antenna is attached to a large, flat metal surface to act as a reception ground plane).

The FUNcube Dongle is priced above the RTL-SDR at about $175. This device is tunable over the frequency ranges of 150kHz to 240MHz and 420MHz to 1.9GHz, with a USB 2.0 host connection. The Funcube Dongle Pro+ (FCD) operates in much the same way as the regular FUNcube Dongle, with the E4000 Tuner IC at the frontend, but improved performance. In the Pro+ device, higher quality components are used for the ADC and LNA, along with a separate microcontroller.

Both systems mix the RF spectrum directly to baseband frequencies, using an SDR architecture technique known as “direct conversion” or “zero-IF”, as no intermediate frequency is used. This contrasts the method used in the USRP and HackRF One, which utilizes mixing from passband to an IF, then downsamples or upsamples from the IF using the FPGA. The hardware architecture matches very closely to that given above in Figure 1: General SDR Architecture.

Other low-cost implementations may use variations on the functional block grouping – adding amplification stages and filtering dependent upon the bandwidth, noise and cost requirements.

Antenna Selection

The final element needed in this system is the antenna. Clearly, there are many choices depending on many factors. In our experience, a low-cost antenna such as the ANT500 (around $30) is suitable for most classroom laboratory experiments. The ANT500 is a telescoping antenna, providing a 50Ω impedance and operating over the range of 75MHz to 1GHz, hence providing capability to operate from VHF (including FM radio) up to the 900MHz ISM band and beyond. For those experiments requiring higher frequencies, Ettus Research does offer the VERT2450 antenna, providing dual-mode operation for both the 2.4GHz and 5.0GHz+ bands.

SDR Software

In this section, we discuss both software for communication system analysis as well as communication system development. Parallel to this discussion is the associated requirements of the host system's configuration. Fortunately, all of these valuable tools are easily available and free for non-commercial use.

Analysis Software

Freeware communication system analysis programs are available to easily interface with and support most SDR hardware. Two such programs are SDR#[24] and Gqrx[25]. SDR# is primarily a Windows-based program (although Linux and Mac OS are supported) while Gqrx is intended for Linux or Mac OS systems. Both of these programs support all SDR systems described in this paper (installation of particular SW drivers may be required).

When connected to an SDR through the host interface, these programs provide multiple signal processing functions (some functionality may require the installation of additional plugins). Useful functions include crystal frequency correction (of, in this case, the RTL-SDR tuner), which may be performed by using the precise narrowband frequency of the NWS Broadcast station. Additionally, various demodulation schemes are available, including broadcast AM and FM (mono and stereo), NBFM, SSB-AM and some digital formats. Analysis capabilities include a variety of spectral analysis tools such as the FFT and waterfall plots as well as adjustable filtering and spectral conversion. A screenshot capture of an SDR# session (while connected to the RTL-SDR) demodulating broadcast FM is shown below in Figure 4.

Development Software

The most popular signal processing tool used for SDR development is GNU Radio; a free open-source program with a large online support community. Typically, GNU Radio serves as the signal processing engine (executing on the host computer) while the SDR hardware provides the RF front end and digitization (note that GNU Radio can also run in a simulation mode without any SDR hardware connected or from recorded data). This software provides the necessary drivers for communicating with the SDR hardware and host system I/O, as well as signal processing blocks for encoding, modulation, filtering, packet handling, stream manipulation and other functions.

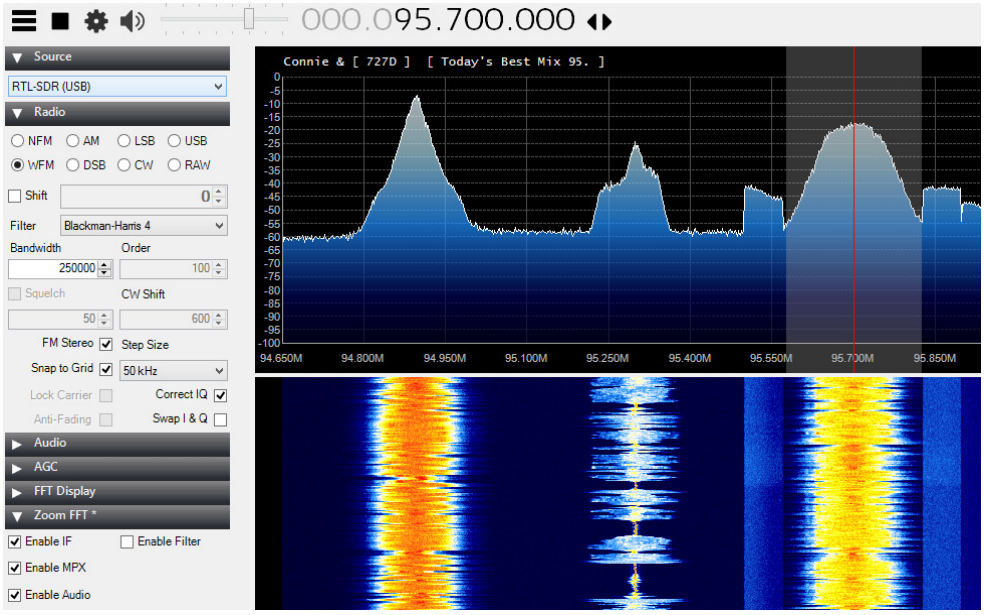

Figure 4: SDR# Screenshot connected to the RTL-SDR receiving Broadcast FM

While installation of GNU Radio on Windows is possible, it is very involved and prone to issues. It is also not supported by the GNU Radio community, therefore if a bug occurs during operation, the user is left to create their own solution. Fortunately, there is an elegant workaround for this problem (as many campus computer labs are Windows-based): a Linux OS is compact enough to be installed on a low-cost USB thumb drive (such as 4GB and larger) and still leave enough space for the full GNU Radio suite, test data and other storage. The boot order of the computer must be configured to check the USB drives first, and the user simply uses the thumb drive as the primary storage device; operation speed is not negatively affected by using the thumb drive. Suitable freeware choices for the Linux OS include Pentoo[26] and Ubuntu[27], as well as others. Creation of a USB-bootable Linux image is typically done using a third-party program (such as Win32 Disk Imager[28]) after obtaining the Linux OS image from their respective distribution website.

The software for GNU Radio is written in Python and C++, where Python is the glue code and C++ performs most of the heavy signal processing. Additional user-generated custom signal processing blocks can be written in either Python or C++ and added to the signal chain. Fortunately, GNU Radio is relatively mature; most functions for general signal processing applications and communications have already been written and optimized, making GNU Radio highly modular. GNU Radio also has the advantage of a run-time scheduler, which optimizes the data flow between processing blocks, creating a more robust real-time test environment.

Communication system development using GNU Radio is greatly simplified with the use of GNU Radio Companion (GRC). GRC is the graphical user interface for GNU Radio, where users place functional blocks into a processing chain known as a flowgraph. Blocks exist for the vast majority of communication system functions, requiring users to simply configure the block with a handful of parameters particular to their system. If desired, a user can create a custom graphical

flow block (in either Python or C++); many such blocks are available via the user community and by default in GRC. Once a flowgraph has been created, the user generates a Python file with a click of a button in GRC, where the signal processing blocks (written typically in C++) are connected together, hence Python glue code.

It should be mentioned that one drawback of using GRC is the lack of formal, organized documentation. As GRC is a free open-source initiative, the motivation to create and maintain detailed exhaustive documentation is understandably low. In a way, the GRC blocks themselves are somewhat self-documenting in that any user can study the underlying code that creates the block to better understand its functionality. There are also many detailed online tutorials, which explain most of the underlying idiosyncrasies of the program.

The drivers for many of the SDR transceivers exist in an easily implemented "source block" in GRC. The source block simply is dragged into the flow diagram, and configured according to the user's preference and particular hardware in use. All SDR devices mentioned in this article have source blocks available in GNU Radio and GRC, where most use the Osmocom source. Figure 5 illustrates the configuration for the FUNcube Dongle and an Osmocom source.

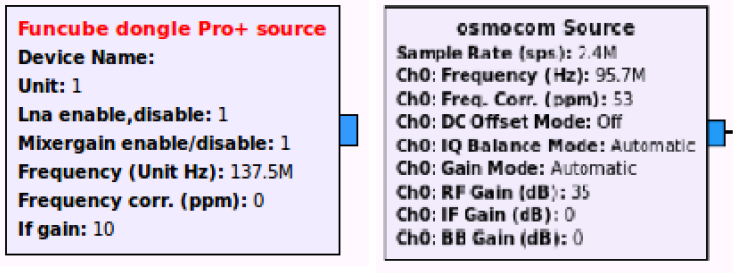

Figure 5: FUNcube Dongle and Osmocome Source Block Configurations

Decimation and Choosing Flow Graph Sampling Rates

In order to select the sample rate to configure the RF source block, considerations must be made about the bandwidth of the signal that is to be received. In addition, neighboring signal bands must be taken into consideration as well if a system is to take advantage of the wideband visualization capabilities of SDR. After the desired bandwidth is determined, decimation can be used to ease the signal processing workload on the host machine. While visualization of a whole band of FM radio signals is beneficial, there is no practical need to feed all of this data into an FM demodulator. A decimation value is chosen such that the final sample rate is just high enough to support the bandwidth required by a given communications channel.

The center frequency value of a software defined radio system corresponds to the frequency that will be exactly in the middle of the chosen bandwidth. When demodulating simple schemes such as analog FM, the center frequency should be chosen to match the advertised frequency of the transmitted signal.

Certain devices have a discrete selection sample rates. For instance, sound cards typically operate with several sample rates ranging from 8 kHz and up. At each connection stage in a software defined radio signal chain, the sample rate must remain consistent. If the rate is

different from the output of one device to the input of another, errors will occur, or the system will not behave as intended. When two steps in a signal chain require different sample rates, decimation or interpolation must occur. One of the easiest mistakes to make when assembling a system is forgetting to match sample rates, and is one cause of choppy, sped up, or slowed down audio.

In a system that is not interfacing with rate-limited hardware, the CPU may accelerate samples through the signal chain faster than the incoming sample rate. In GNU Radio, a "Throttle block" is used to control this situation. The throttle block takes a sample rate parameter, and will limit the rate of samples passing through the block to the rate specified. Most blocks do not check that a sample is arriving at the correct time according to the sample rate specified, so this throttle block can prevent the high CPU usage and errant results that take place when a system has an otherwise unbounded rate.

Complex Sampling

A central concept of the systems described is their maximum sampling rate and associated bandwidth. In order to maximize the bandwidth and highest frequency available to the device, complex sampling is employed. Essentially, complex sampling borrows the concept of SSB-AM in that if only one sideband of the sampled spectrum is maintained, aliasing can be avoided as long as

$$f_{\max} < f_{samp},$$

where $f_{\max}$ is the maximum frequency component in the signal and f_{samp} is the sampling frequency. Thus, the chosen sampling rate is the maximum signal bandwidth. This arrangement is achieved by creating a complex I/Q representation $x_{IQ}(t)$ of the original signal real-valued $x(t)$, expressed as

$$x_{IQ}(t) = x(t) + x_h(t).$$

The signal $x_h(t)$ is an imaginary-valued signal known as the Hilbert Transform[29] of $x(t)$, and can be found by passing $x(t)$ through a filter $H(\omega)$ with a response of

$$H(\omega) = -j\,\mathrm{sgn}(w) = \begin{cases} e^{-j\pi/2} & \omega \geq 0 \\ e^{j\pi/2} & \omega < 0 \end{cases};$$

that is, $H(\omega)$ introduces a $\pm 90^{\circ}$ phase shift over all frequencies, which is possible in the narrowband sampling schemes done for SDR. Note that complex sampling can be done in continuous time (as expressed above) or done in discrete time (easier with digital filters, but would require a temporary doubled sampling rate).

SDR Experiments and Projects

For a typical academic environment laboratory to offer instruction to students in the areas of analog and digital communications, we recommend the following configuration:

- Every workgroup or student should have their own SDR-RTL (comes with antenna) for receiver development;
- For every 2-4 workgroups, one HackRF One (primarily for transmitter duties) is recommended, additionally outfitted with the ANT500 antenna;
- Each workgroup needs access to a computing workstation, typically running the Windows OS; this computer should have SDR# installed for signal analysis, and should be configured to boot first from a USB drive;
- Each workgroup requires an 8GB thumb (flash) drive with Pentoo Linux installed, which includes a pre-installation of GNU Radio, along with all the necessary SDR hardware drivers; enough space exists on the flash drive for data storage and multimedia files for experimentation.

This recommended setup allows the students to perform analysis and to generate transmitter and receiver communication systems at a very reasonable price point, around $100 per student group.

We successfully used the recommended configuration for a variety of analog and digital communications experiments in our EGR 415 Communication Systems course offered at Grand Valley State University in the Fall of 2015. This course is taught at the senior-undergraduate/first-year-graduate student level, and it is an EE elective in the Signals and Systems emphasis area. The course is composed of a traditional lecture session of 3 hours/week with a weekly 3 hour laboratory. Topics follow a traditional progression through analog modulation techniques such as AM and FM, followed by baseband digital communications and pulse-BW/shaping into bandpass digital communication methods such as ASK and PSK. Performance in the presence of noise is likewise considered. If time allows, advanced topics such as Spread Spectrum or OFDM are introduced.

We now discuss examples of the experiments and projects that the students were able to complete with this setup. We consider experiments in both analog and digital communications.

Analog Labs

Lab #1: Introduction to SDR

This introductory lab is intended to familiarize students with SDR. The lab explores a few different methods of using SDR, first using Windows, and then transitioning to a Linux distribution. This lab employs the RTL-SDR using both analysis and development tools.

In Windows, students begin experimenting with SDR#. Because FM radio is one of the strongest common signals present within the tuning range of the RTL-SDR, it is chosen as the signal to

demodulate in this introductory lab. Students experiment with the many views and controls of SDR#, and explore the RF spectrum using this simple tool.

Moving onto Linux, GNU Radio Companion (GRC) is explored by creating a simple FM demodulator. The significance of each block in the sequence is demonstrated, and students are required to use what they learned in SDR# regarding RF visualization techniques to understand the outputs of GRC. A GRC flowgraph of the FM demodulator is given below in Figure 6.

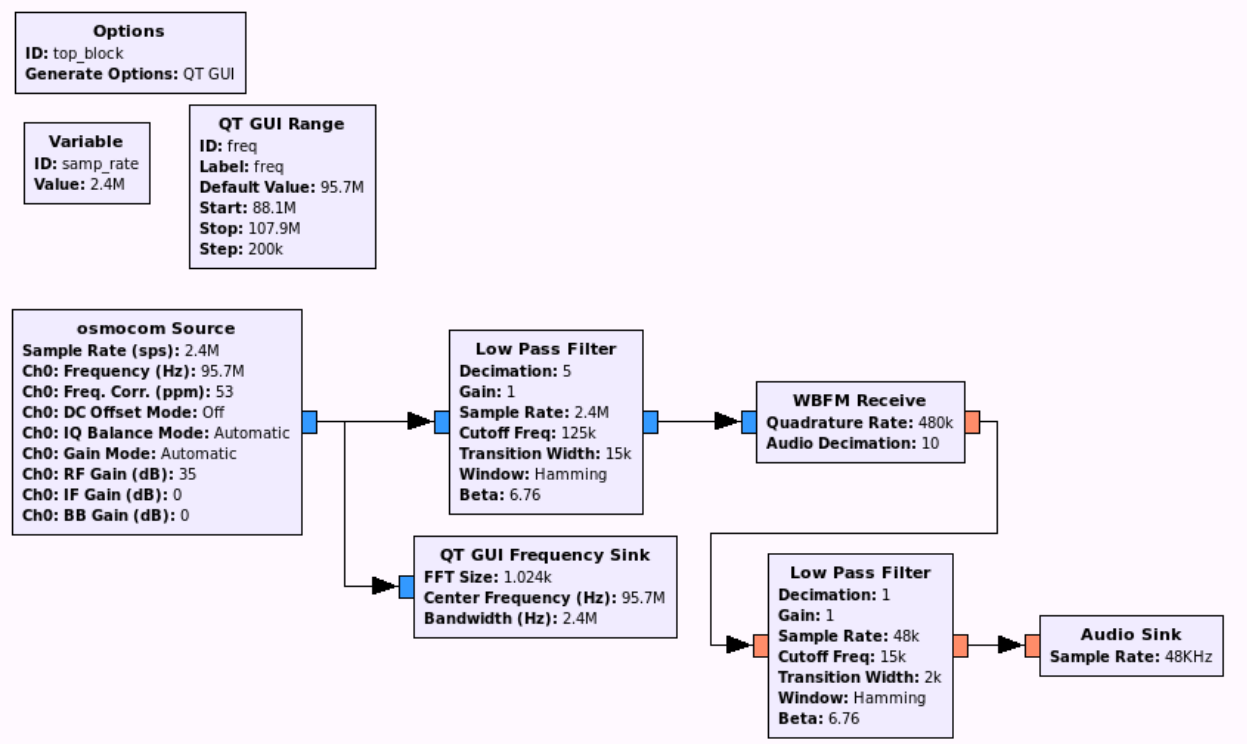

Figure 6: GRC Flowgraph for RTL-SDR FM Demodulation

Lab #2: AM

This lab is focused on the modulation and demodulation of double-sideband suppressed carrier (DSB-SC) and broadcast AM signals and utilizes the HackRF One for transmitting, and the RTL-SDR for receiving. Students are instructed to create a DSB-SC signal, and demodulate it (as shown below in Figure 7). They are then asked to observe the effects of phase error in a communications system that requires coherency. Once DSB-SC AM has been exercised, students are asked to move on to broadcast AM. Utilizing blocks in GRC, students are asked to create and test a modulator and demodulator for this format. Finally, as an extra requirement, they attempt to tune in broadcast AM signals in the regular band, using the HackRF and an improvised antenna capable of receiving these lower frequencies.

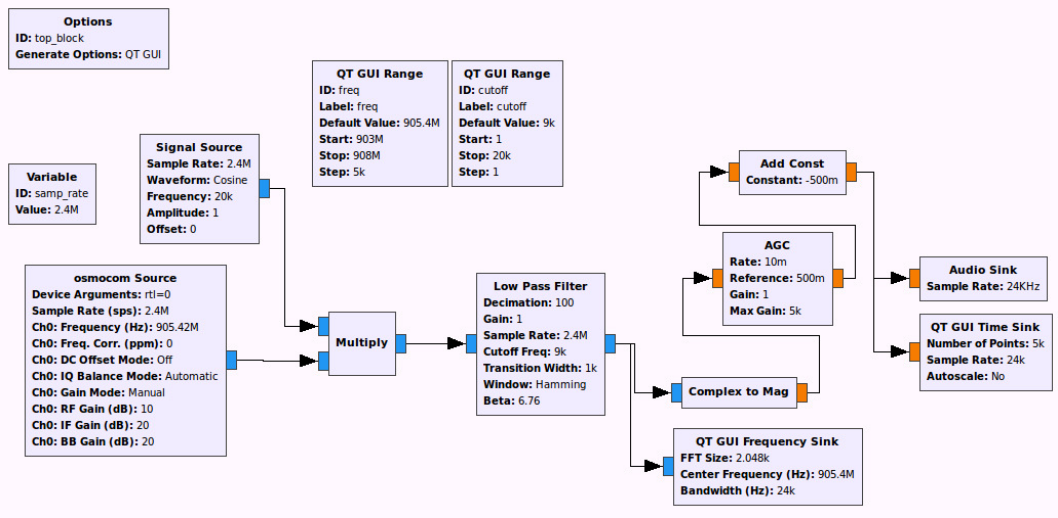

Figure 7: GRC Flowgraph for DSB-SC AM Demodulation

Lab #3: FM

This lab dives deeper into frequency modulation. In the *Introduction to SDR* lab, students simply created an FM receiver using a limited number of blocks. The requirements of this lab are greater, requiring students to create narrowband FM using a standard block, but increasing the modulation index by way of the Armstrong method. Next, students are asked to generate wideband, stereo FM and transmit it with the HackRF One SDR. The students broadcast (a low power version) of their signal on a locally unused FM broadcast frequency and then tune and listen to it on a conventional FM radio. Finally, students are asked to use what they have learned throughout the prior steps in the lab to create a stereo FM demodulator. A flowgraph for the WBFM generation is included below in Figure 8.

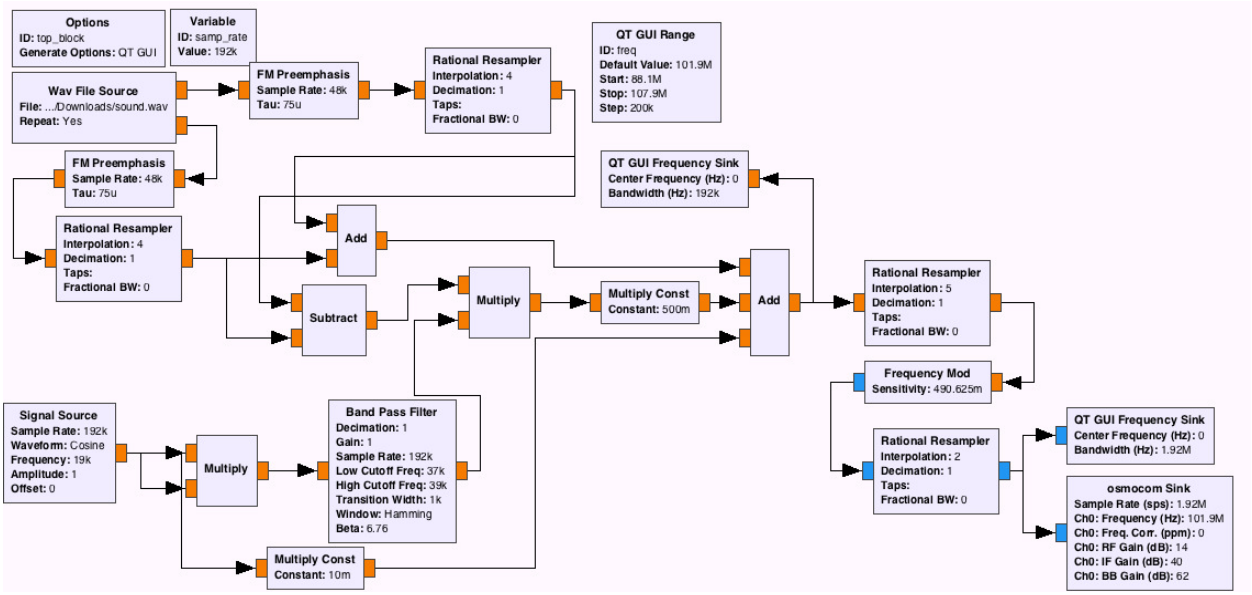

Figure 8: Wideband FM Modulation on the HackRF One SDR

Digital Projects

Project #1: BFSK Transceiver

This project is focused on modulating and demodulating a data stream using Binary Frequency Shift Keying (BFSK). Using a vector source to provide data, a transmitter is created using non-continuous phase frequency shift keying. After resampling to reach an adequate rate for the HackRF One, the FSK signal is transmitted.

On the receiving end, a BFSK detector is implemented using an integrate-and-dump method. The result is shown on a time sink. Because of the difficulty quantifying delays encountered in transmitting and receiving through the SDR, the exact system delay is unknown and hence, accurate bit error rate calculation was not possible. Most of the BFSK receiver is shown below in Figure 9.

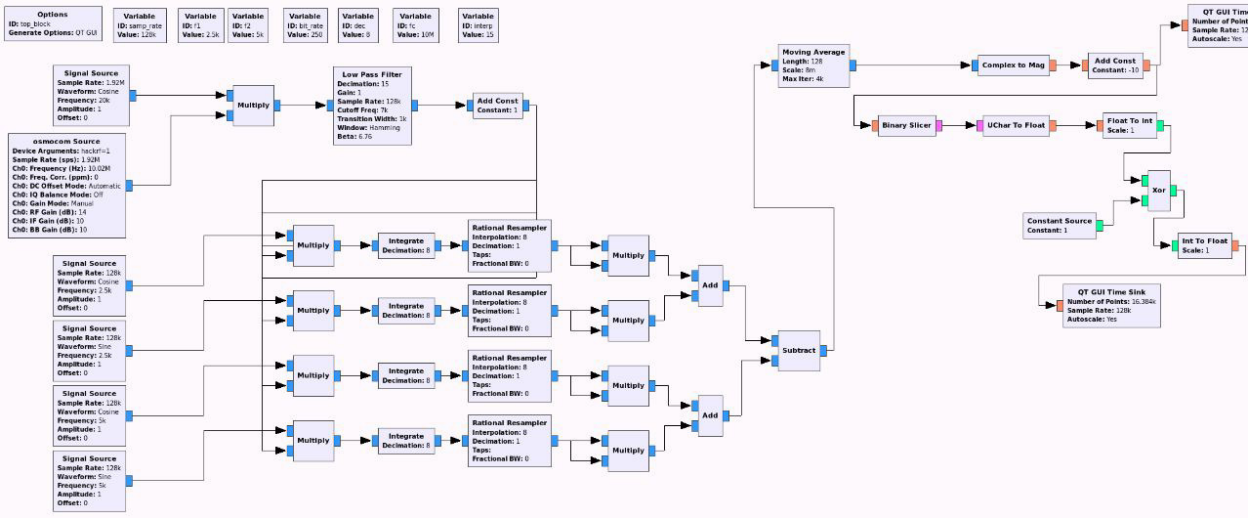

Figure 9: BFSK Receiver

Project #2: ATSC Transmitter

This project focuses on the generation of ATSC digital television. Using the sequence of blocks shown below, a video encoded using FFmpeg[30] can be converted into an MPEG transport stream for transmission to a digital television set. Because of the current requirements of the FCC, most televisions in use today contain a tuner that is able to receive this signal. This project is well suited for demonstrating applications of SDR because of the tangible output that can be viewed by a public audience. An advantage of using SDR for this application is that any frequency can be selected, which allows for a wide range of TV channels to be selected. A flowgraph of this project is given below in Figure 10.

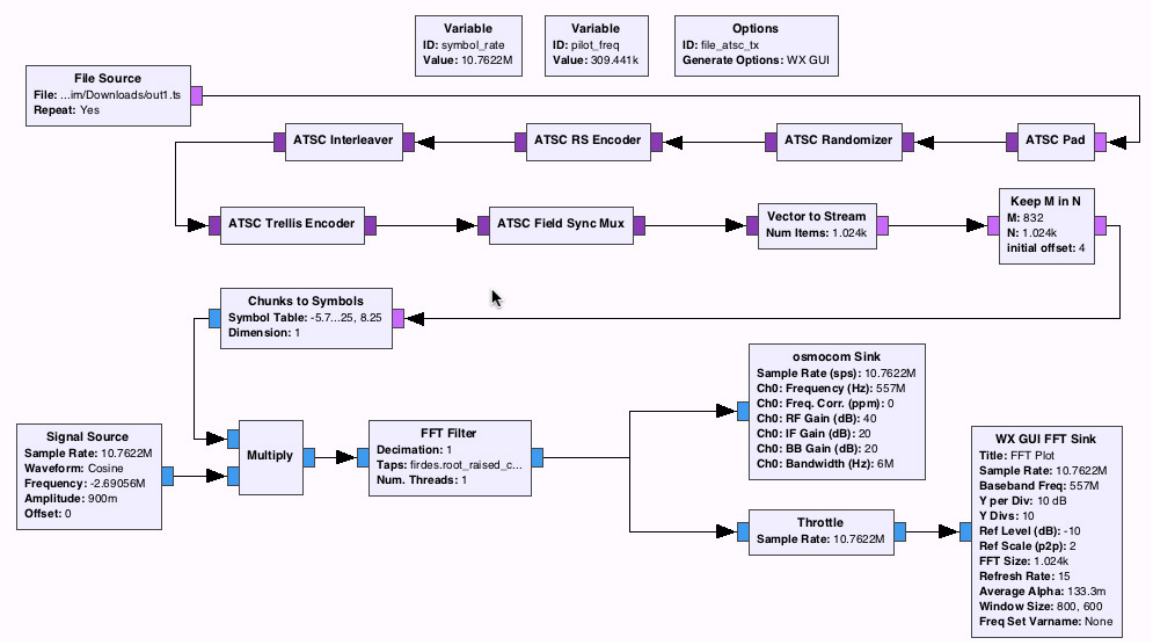

Figure 10: ATSC Transmitter

Assessment

Assessment was confined to an anonymous student survey with open-ended comments. These results are summarized below. The following questions were asked of the students (7 students elected to answer the survey) using the answer key: 5: strongly agree; 4: mostly agree; 3: somewhat agree; 2: disagree; 1: strongly disagree:

Q1: The use of SDRs in a course laboratory setting is beneficial to learning concepts in analog and digital communications.

Q2: The use of SDRs is preferred over simulation exercises, because the experiment better matches a real communication environment.

Q3: SDR provided an opportunity to gain a rounded experience in communications.

Q4: Please provide comments about the use of SDR in a course laboratory environment.

The mean and standard deviation of the students' responses to the first three questions were computed. These results are displayed below in Figure 11, where the center point is the mean and the error bars represent the standard deviation, and the x-axis identifies the question number.

The survey results indicate that the students felt that SDR was a good way to learn concepts in communication theory, especially analog communications. The variance on the second question is higher because several students felt that simulation (in particular, MATLAB Simulink), and not the use of SDRs, was a preferred method to learn certain concepts in digital communications.

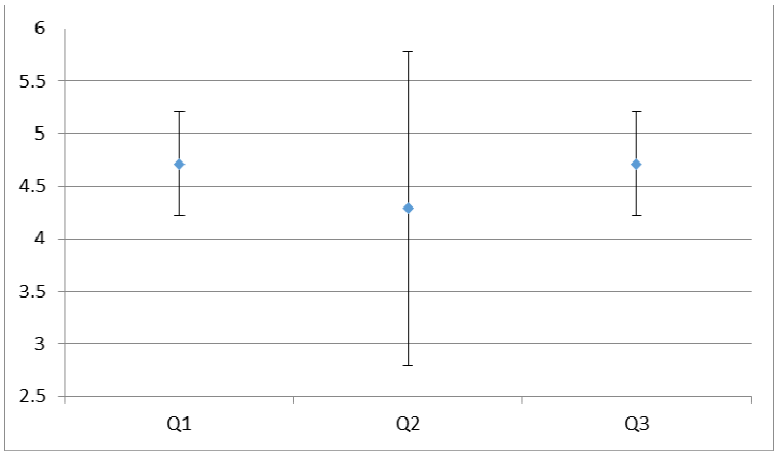

Figure 11: Student Survey Results Summary

The fourth question solicited student comments. A selection of insightful comments is given below:

1. *SDR allows for an easy application of communication theories that are sometimes difficult to conceptualize. AM & FM techniques as studied through GNU radio allowed for the concept to click much quicker. Furthermore, projects are easier to perform due to the wide variety of techniques that SDR can do.*

2. *I thought using the SDR was a good way to physically see the effects of using analog and digital communication.*

3. *It is interesting to see how it works with radio FM/AM. But, for more complex things, Simulink is fine.*

4. *SDRs made laboratory exercises much more straightforward--it takes a lot of the random variables out of the equation when troubleshooting, which is invaluable when one is trying to implement concepts taught in class. SDR probably doubly stands for "sigh of delightful relief".*

5. *My overall experience was that Simulink was a lot harder to use than GNU Radio, but was more effective in teaching communication concepts than GNU Radio.*

The student responses align with the scores seen on the questions. While there seems to be particular advantage for the ease of using GRC, some communications concepts are better explored using a simulation tool such as MATLAB Simulink.

Conclusion

In this paper, we discussed some of the more popular SDR hardware systems as well as the features and considerations of the software available to control these systems. By comparing the features and cost of these systems, we were able to recommend an easily acquirable grouping of hardware and software for the typical academic professional to offer communications instruction

via SDR at a very reasonable price point of about $100 per student workgroup. We were then able to describe a sampling of the experiments that can easily be performed with this setup for both analog and digital communications. Finally, we presented the results of a student survey which supports the use of SDR equipment in the academic laboratory setting.

Bibliography

1. Bard, J. & Kovarik, V., "*Software Defined Radio: The Software Communications Architecture*," Wiley Series in Software Radio, 2007.
2. Reed, J., "*Software Radio: A Modern Approach to Radio Engineering*," Prentice Hall, 2005.
3. Mao, S., & Huang, Y., & Li, Y. (2014, June), *On Developing a Software Defined Radio Laboratory Course for Undergraduate Wireless Engineering Curriculum* Paper presented at 2014 ASEE Annual Conference, Indianapolis, Indiana. https://peer.asee.org/22880
4. Wu, Z., & Wang, B., & Cheng, C., & Cao, D., & Yaseen, A. (2014, June), *Software Defined Radio Laboratory Platform for Enhancing Undergraduate Communication and Networking Curricula* Paper presented at 2014 ASEE Annual Conference, Indianapolis, Indiana. https://peer.asee.org/23023
5. Hoffbeck, J. (2009, June), *Teaching Communication Systems Using The Universal Software Radio Peripheral (Usrp) And Gnu Radio* Paper presented at 2009 Annual Conference & Exposition, Austin, Texas. https://peer.asee.org/5126
6. Wyglinski, A. M., & Cullen, D. J. (2011, June), *Digital Communication Systems Education via Software-Defined Radio Experimentation* Paper presented at 2011 Annual Conference & Exposition, Vancouver, BC. https://peer.asee.org/17783
7. Universal Software Radio Peripheral, http://www.ettus.com/
8. GNU Radio Wiki, http://gnuradio.org/redmine/projects/gnuradio/wiki
9. Welch, T. B., & Shearman, S. (2011, June), *LabVIEW, the USRP, and their Implications on Software Defined Radio* Paper presented at 2011 Annual Conference & Exposition, Vancouver, BC. https://peer.asee.org/18249
10. Prust, C. J., & Holland, S., & Kelnhofer, R. W. (2014, June), *Ultra Low-Cost Software-Defined Radio: A Mobile Studio for Teaching Digital Signal Processing* Paper presented at 2014 ASEE Annual Conference, Indianapolis, Indiana. https://peer.asee.org/23216
11. Morrow, M., & Kent, T., & Welch, T., & Wright, C. (2009, June), *Teaching With Software Defined Radios* Paper presented at 2009 Annual Conference & Exposition, Austin, Texas. https://peer.asee.org/4938
12. Wright, C., & Morrow, M., & Welch, T. (2010, June), *Using Inexpensive Hardware And Software Tools To Teach Software Defined Radio* Paper presented at 2010 Annual Conference & Exposition, Louisville, Kentucky. https://peer.asee.org/15922
13. Azarmanesh, O., & Bilen, S. (2010, June), *Experiences With Student Developed Software Defined Radios In The Smart Radio Challenge* Paper presented at 2010 Annual Conference & Exposition, Louisville, Kentucky. https://peer.asee.org/16490
14. Online article: "Choose the right A/D converter for your application," http://www.ti.com/europe/downloads/Choose%20the%20right%20data%20converter%20for%20your%20application.pdf
15. Bandwidth Capability of USRP Devices, http://www.ettus.com/kb/detail/usrp-bandwidth
16. TI OMAP Wiki, http://processors.wiki.ti.com/index.php/OMAP
17. AD9364 Wideband Software Defined Radio Board, http://www.analog.com/ad-fmcomms4-ebz
18. Nuand bladeRF - the USB 3.0 Superspeed Software Defined Radio, http://www.nuand.com
19. Great Scott Gadgets HackRF One SDR Peripheral, http://greatscottgadgets.com/hackrf
20. HackRF One KiCAD Schematics, https://github.com/mossmann/hackrf/tree/master/doc/hardware

21. RTL-SDR Specifications, http://sdr.osmocom.org/trac/wiki/rtl-sdr
22. FUNcube Dongle description, http://www.funcubedongle.com/
23. RTL2832U General Description, http://www.realtek.com.tw/products/productsView.aspxLangid=1&PFid=35&Level=4&Conn=3&ProdID=257
24. SDR# SDR Software, airspy.com
25. Gqrx SDR Software, gqrx.dk
26. Pentoo Linux OS, www.pentoo.ch
27. Ubuntu Linux OS, www.ubuntu.com
28. Win 32 Disk Imager at the Ubuntu Wiki, https://wiki.ubuntu.com/Win32DiskImager
29. Proakis, J.G. & Salehi, M., "*Fundamentals of Communication Systems*," Pearson Prentice Hall, 2005, pp. 99-102.
30. FFmpeg multimedia framework, www.ffmpeg.org