A Cost Efficient Software Defined Radio Receiver for Demonstrating Concepts in Communication and Signal Processing using Python and RTL-SDR

Boonyarit Uengtrakul[1], Dahmmaet Bunnjaweht[2]
Dept. of Electrical and Computer Engineering
Thammasat University, Pathumthani 12120, Thailand
[1]mezzoblues@hotmail.com
[2]dahmmaet@engr.tu.ac.th

***Abstract*— Communication and signal processing courses are the cornerstone of electrical engineering studies and other related fields. Due to the nature of the subjects, most of the concepts are modeled with mathematical equations. Experimentation is one of the important components that helps students gain more understanding in the practical applications of those concepts and get motivated. Software defined radio makes it possible to experiment and demonstrate concepts in communication and signal processing with real radio signals. However, software defined radio systems are usually expensive; providing software defined radio experimental sets that are accessible to every student can cost a considerable amount of budget. Therefore this work proposes a way to minimize the cost of using software defined radio as an educational tool by combining an ordinary digital TV tuner with a special software framework written in Python into an affordable software defined radio experimental kit. The way this experimental kit works is, a radio signal will be received into the computer via the digital TV tuner that operated with the custom driver called RTL-SDR, and then the received signal will be processed with a Python signal processing script. With the aid of the created software framework, writing a signal processing script can be much easier than using only the Python standard libraries. With this combination of inexpensive hardware and the provided software framework, the kit enables many possibilities for utilizing real world signals in classroom demonstrations and experiments. An example of using the kit for demonstrating the concept of narrowband FM demodulation is presented in this paper to show the overall workflow and how this work might be integrated into a curriculum.**

Keywords*— *Software Defined Radio; Cost Efficient; Affordable; RTL-SDR; Python; Communication; Signal Processing; Education; Demonstration;

I. INTRODUCTION

Communication and signal processing are very important courses in electrical engineering studies and many other related fields. The concepts in these two courses are fundamental knowledge that every student should master for their practical work and for continuing studies in more advanced courses. Due to the nature of the subjects, most of the concepts are illustrated with mathematical equations that are completely abstract. The students may do experiments using simulation software, but these simulated experiments are still not able to emulate many conditions that might be encountered in real world signal processing and communication systems. Software defined radio is one of the solutions that can be applied to be educational tools for enabling students to experiment with real radio signal. Unfortunately, software defined radio systems are usually expensive; providing software defined radio experimental equipments that are accessible for every student could cost a considerable amount of money. Therefore, this work is about research on hardware and software that can be applied together into an affordable software defined radio experimental kit. The kit can be used for demonstrating various concepts in digital signal processing and communication courses.

The experimental kit proposed in this work is a cost efficient software defined radio receiver. The kit comprises a digital TV tuner, operated with the custom driver called RTL-SDR [1], and a newly created software framework for programming an experiment, implemented in Python [2]. The overall system diagram of this software defined radio kit is illustrated in Fig. 1. A digital TV tuner is used as hardware for receiving a radio signal into the computer. All the signal processing processes are done with computer software via processing scripts which rely on the created software framework. The software framework provides convenient tools for data acquisition and radio signal manipulation. Students and instructors can easily create demonstrations and experiments with little or even no prior programming experience.

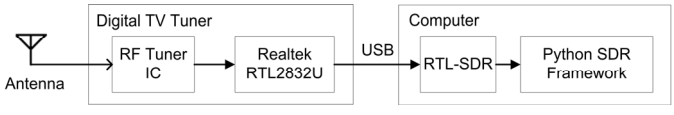

Fig. 1. The overall system diagram of the software defined radio kit

In section II, III, IV and V, the related background knowledge is presented along with the analytics details and the reasons for choosing each hardware or software to be compiled into the cost efficient software defined radio receiver kit. The implementation details of the created software framework are

ISBN: 978-1-4799-3724-0/14/$31.00 ©2014 IEEE

described in section VI. In section VII, an example experiment is presented for showing how to use the experimental kit for demonstrating the concept of wideband FM radio demodulation. The example experiment will provide readers the overview of how this work could be applied to a real classroom experiment.

II. Software Defined Radio in Education

Software defined radio is a radio system in which the demodulation scheme and signal processing are implemented using software. Unlike traditional hardware based radio systems, software defined radio is not designed for a specific modulation scheme, the signal processing processes can be completely defined in the software implementation. This innovation yields a lot of advantages in the field of communication. The communication systems which employ software defined radio technology do not need hardware reconfiguration when the modulation or coding schemes are changed, only software update is required. On the receiver side, software defined radio works on the principle that the received radio signal is converted to digital using an analog to digital converter, then signal processing processes will be done in a programmable processor e.g. FPGA, microprocessor, personal computer. On the transmitter side, software defined radio works on the same principle but in the reverse order, a digital signal is sent from a processor to a digital to analog converter and then transmitted.

Software defined radio not only benefits communication industries but it can also be applied for use as educational tools. A radio system with a programmable signal processor has opened many possibilities for organizing classroom experiments with physical radio signals. It allows students to freely generate radio signals in various forms of modulations and transmit them to the physical world. It also allows students to capture radio signals in the air and freely manipulate them with the mathematical models they have learned in their classes. This enables endless exploration and hands-on laboratory exercises that can be seamlessly integrated with the curriculum. The examples of using software defined radio for demonstrating concepts in communication and signal processing courses are exemplified in [3]. For example, when students have studied a modulation technique, they can generate the modulated signal by programming a software defined radio transmitter according to the theory of that modulation technique, the generated signal can be received by real world devices. They can also learn to demodulate signals generated from real world transmitters e.g. local radio station, wireless microphone, walky-talky by programming a demodulation technique into a software defined radio receiver.

III. RTL-SDR

RTL-SDR is a custom driver software that operates a USB digital TV tuner to function as a software defined radio receiver. It is an open source software developed by Osmocom [4]. The driver works on digital TV tuners based on the Realtek RTL2832U data acquisition chip.

In general, a USB digital TV tuner contains two main ICs. One is a tuner IC which is used for receiving a signal, a direct conversion receiver is usually implemented in a tuner IC. Another one is a data acquisition IC which is used for sampling the received signal and sending it to the computer. RTL-SDR works on the principle that the Realtek RTL2832U data acquisition IC has an undocumented mode that can directly sample the raw I/Q data to the host computer, which enables the software defined radio application on the DVB-T that is based on Realtek RTL2832U. The output of RTL2832U is an output of 8-bit I/Q-samples and the highest possible theoretical sample-rate is 3.2 MS/s. However, the highest sample-rate without lost samples depends on the performance of the host computer. The receivable frequency range is highly dependent on the tuner chip and the antenna, dongles that are equipped with the Elonics E4000 [5] offer the widest range (64 - 1700 MHz but exclude 1100 - 1250 MHz). More information on the list of compatible tuners and their receivable frequency ranges can be found on Osmocom website [4].

The tuner chip of an RTL2832U based dongle, as well as other software defined radio platform, usually employs a direct-conversion receiver, which quadrature sampling detectors are used in the mixer stages. The direct-conversion receivers are used for shifting the portion of a spectrum at the given center frequency down to baseband, the width of the portion is determined by the sampling frequency. It is a standard for software defined radio to convert the radio signal to in-phase and quadrature components, at the mixer stages, before being digitalized by the analog-to-digital converter on the data acquisition chip.

The educational applications of RTL-SDR are widely interesting to universities and commercial software companies. The University of California, Berkeley has started using RTL-SDR as one of the project assignments in their digital signal processing course [6]. MathWorks has also released the RTL-SDR official support package [7] for MATLAB and Simulink.

This work is chosen to be implemented on RTL-SDR because RTL-SDR works on very low price hardware, but has sufficient frequency range. However, the performance of RTL-SDR cannot be compared to a professional grade software defined radio like Ettus Research USRP [8] which has far higher performance in both frequency range and sampling rate, The cheapest model of Ettus Research USRP is at least twenty times more expensive than the prices of average RTL-SDR hardware. In terms of cost, there is no other software defined radio platform that can be compared to RTL-SDR. Therefore, it seems to be the most suitable choice for building a cost efficient software defined radio receiver kit.

IV. Python

Python is a high-level object-oriented programming language used for implementing the proposed framework. The main reason for choosing Python is it was designed with the philosophy of code readability that means programs written in Python are easier to understand by humans. Also programs written in Python usually have relatively fewer lines of code than the same programs written in other object oriented languages such as C++, C#, Java.

In a perspective of cross-platform development, Python runs on a broad range of platforms. Python script is

independent from operating system because the scripts are executed on a Python interpreter that can be run on various operating systems ranging from desktop operating systems like Microsoft Windows, Mac OS X, Linux to mobile operating systems like Android and iOS.

Comparing with industrial standard software like MATLAB, both Python script and MATLAB script are dynamically typed languages, variables do not need to be explicitly declared. Both languages can automatically inference the type of variables when their values are assigned. Both Python and MATLAB also come with a read–eval–print loop that is the command line user interface for algorithmic testing and debugging. The similarities between these two computational environments make it is possible to use Python as a free open source alternative to commercial software like MATLAB. Users require a few adaptations to migrate from MATLAB to Python. There are also lots of extension libraries that are available to Python for adding new functionalities, that will be discussed in the next section.

V. Python Software Libraries

One of the greatest advantages for using Python as a scientific computational environment is its extensive extension modules. Modules can be installed to add new functions and capabilities to the Python interpreter. The Python Community [9] is one of the great sources where some useful modules for a particular work can be found. Most of the available modules are under open source licenses that allow customization for special applications. In this work, the software framework is implemented on top of the four libraries which are explained as follows.

A. SciPy

SciPy [10] is a scientific computing library that can be used in a variety of different contexts. SciPy comes with essential signal processing functions e.g. convolution, filtering, filter design, window functions, spectral analysis, wavelets. These functions are useful for processing the radio signal from RTL-SDR.

B. NumPy

NumPy [11] is a numerical library that provides some useful data structures, such as multi-dimensional arrays and matrices, along with their mathematical-operation routines. NumPy makes sample manipulations e.g. addition, subtraction, multiplication more convenient rather than purely using primitive data types of Python. NumPy also contains some useful mathematical functions which are commonly used in signal processing e.g. average, trigonometry, truncation, etc.

C. matplotlib

matplotlib [12] is a data visualization library. The matplotlib can be used for generating many kinds of plots e.g. two-dimensional plot, three-dimensional plot, spectrogram, histogram, etc. It is also capable of showing the legend, labeling the axis and coloring the graph. The radio spectrogram and spectrum will be plotted via this library.

VI. The Work

This section is the most important section. It is about the implementation of the software framework based on the software and hardware described in the previous sections.

A. Python Wrapper For RTL-SDR

In fact, RTL-SDR is not a driver itself, it is a software library which takes control of a Realtek RTL2832U dongle to provide programming interfaces for software defined radio. Because the programming interfaces of the original driver from its manufacturer are not disclosed, RTL-SDR communicates with the dongle through a generic USB device driver, which provides basic functionalities for communicating with a device connected over the USB protocol.

The RTL-SDR was written in C language. It can interact with other software by compiling its source code into a shared library e.g. dll file in Microsoft Windows or lib file in Linux. A shared library can be loaded by the Python interpreter, however, directly interacting with RTL-SDR requires some knowledge of programming the hardware interfaces that are skills in computer science and computer engineering. Electrical engineering students should not dwell too much on these programming techniques which could take most of their time.

To provide a more convenient experimental environment, a software interface, called a Python wrapper for RTL-SDR, is created to be a part of the framework. The wrapper acts like a middleman communicating between the users and RTL-SDR. It provides the users all the common data acquisition routines, so the users do not have to directly program the RTL-SDR, they can just simply command the dongle through the routines provided in the wrapper. This way of abstraction also has advantages for source code maintenance, when the RTL-SDR is updated to a newer version, only the wrapper source code needs to be changed.

B. RF Scanner Program

RF scanner program is a program used for visualizing the whole frequency spectrum of the receivable frequency range. It is a useful tool for searching a signal of interest, like searching for a radio station. The program takes four parameters that are scanning frequency range, tuner gain, sampling frequency and FFT length as the function prototype given below.

```
rf_scanner(freq_range, gain, fs, NFFT)
```

When the above function is called, the program will set the center frequency of the tuner to the lowest possible value for obtaining the lowest portion of the radio frequency spectrum, then, the center frequency will be increased by half sampling rate for obtaining the higher portion of the radio frequency spectrum. The center frequency will be increased until it reaches the highest possible value, and the whole radio has been constructed.

C. Capture Program

Capture program is a program for recording the radio signal at a specified frequency range. The program takes five parameters that are, center frequency of the tuner, sampling

frequency for data acquisition, gain of the tuner, duration in second and file name. The captured portion of the spectrum is illustrated in Fig. 2. The recorded samples will be stored in I/Q sample format. The function prototype of the capture program is given below;

```
capture(fc, fs, gain, duration, filename)
```

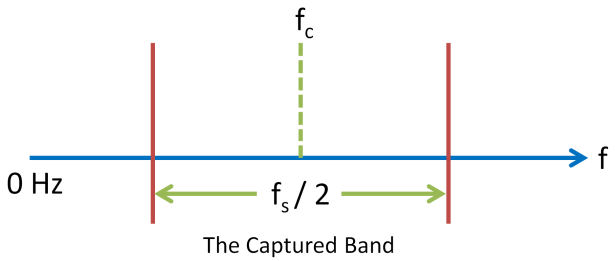

Fig. 2. The captured portion of the spectrum.

D. Sample Reader Program

Sample reader program is used for loading a sample file created with the capture program. Its function prototype is given below;

```
fs, iq_samples = read_iq(filename)
```

When the program is called, the sampling frequency will be loaded from the file header and the I/Q samples in the rest of the file will be loaded into an array of complex numbers, the complex number representation is provided by NumPy.

E. I/Q Modulator Program

I/Q modulator program is used for converting complex I/Q samples back to a real signal. The conversion can be done using quadrature modulation described in [13]. The block diagram of the quadrature modulator implemented in this program is illustrated in Fig. 3. The function prototype of I/Q modulator program is given below;

```
x = iq_mod(iq_sample, fs)
```

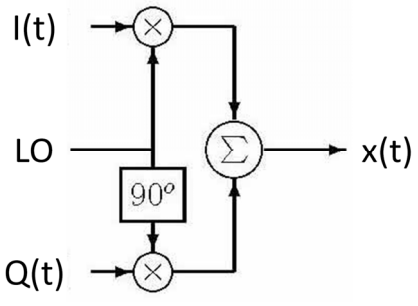

Fig. 3. The Quadrature Modulator Block Diagram

F. Power Spectrum Plotter Program

Power spectrum plotter program is used for plotting the power spectral density, it relies on the fast Fourier transform routine provided in SciPy and uses plot routine of matplotlib for generating the graph. The function prototype of power spectrum plotter is given below;

```
psd_plot(x, NFFT, fs)
```

G. Signal Processing Script

A signal processing script is a Python script written by a user in order to process a radio signal. The scripts are created from the cooperation between the software framework mentioned in the previous sub-sections and the signal processing functions provided in SciPy. The typical workflow for creating a signal processing script usually starts with scanning the radio spectrum, capturing the desired frequency range, reading the captured samples, converting the samples to a real signal and manipulating the real signal with SciPy.

VII. Example Experiment

This section shows an example of applying this work to a real classroom demonstration or student experimentation. The example experiment presented in this section is about demonstrating the concept of wideband FM demodulation. It is a situation when students are assigned to demodulate the FM signal with an unknown carrier frequency, generated by the instructor, in order to retrieve the original audio. Or it might be the case when the instructor demonstrates demodulating real FM signal to the class.

The real FM radio signal in this example experiment was generated from AKG UHF PT40 [14], a pocket wireless transmitter. The transmitter modulates a mono audio input to the frequency of 711.000 MHz. A music synthesizer [15] was used as a sound source for the transmitter. The overall setup of the example experiment is depicted in Fig. 4. The students will use the proposed software framework in cooperation with SciPy library to write their own signal processing script for demodulating the FM signal.

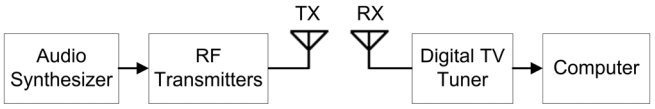

Fig. 4. The overall setup of the example experiment

For a real world FM radio, the result can be justified by listening to the demodulated audio and assessing the sound quality by ear. However, in this example experiment, students are required to make a formal comparison between the two signals in order to put the result in their lab reports. To achieve that goal, the original sound without modulation was recorded using a computer sound card at the sampling rate of 44.1 kHz for comparing with the demodulated signal. Students can use the power spectrum plotter program for generating the power spectrum of those two signals. The power spectral density of the original sound without modulation is shown in Fig. 5.

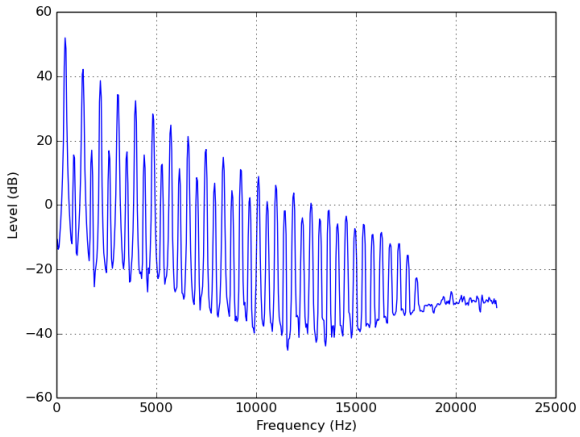

Fig. 5. The power spectrum of the original sound

Next step, the synthesizer was connected to the RF transmitter and started generating the sound, then the RF scanner program was called with the following command to scan the whole frequency range.

```
rf_scanner(freq_range="Full Range", gain=0,
fs=1.024e6, NFFT=1024)
```

The scanner program showed the radio spectrum in Fig. 6.

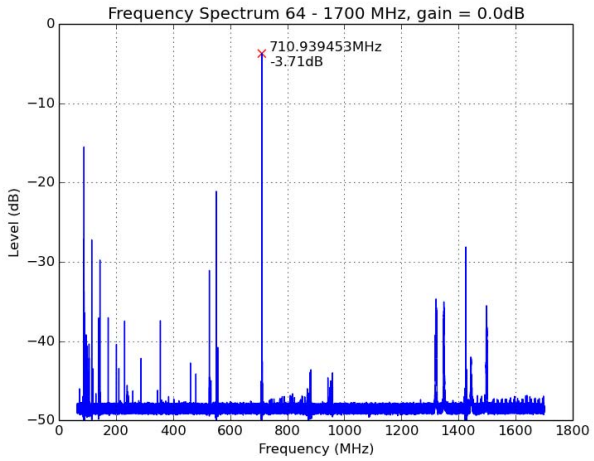

Fig. 6. The power spectrum of the whole receivable frequency range.

The signal sent from the transmitter was found at approximately 710.94 MHz, which is very close to the actual frequency specified in the transmitter datasheet. Next, the radio signal was captured using the following command.

```
capture(fc=711e6, fs=1.024e6, gain=10,
duration=2, filename="iq_data")
```

The captured I/Q samples will be loaded and converted back to real signal. Then the signal will be demodulated using frequency detector with FM-to-AM conversion adapted from [16]; the processes are described in Fig. 7. All of the steps can be straightforwardly implemented. There are only a few tricks on the envelope detector that can be implemented using Hilbert transform, described in [17].

Fig. 7. A Frequency Detector with FM-to-AM Conversion

The complete Python script of the example experiment is given in the following box.

```
#Loading and Converting
fs, iq_samples =
read_iq(filename="iq_data")
x = iq_mod(iq_sample, fs)

#Differentiator
dx_dt = diff(x, n=1, axis=0)/dt
dx_dt = concatenate([dx_dt,[0]])

#Envelope Detector
x_hilbert = signal.hilbert(dx_dt)
x_env = sqrt( power(x_hilbert, 2) +
              power(dx_dt, 2) )
x_env = abs(x_env)

#DC Block
dc_level = average(x_env)
x_demod = subtract(x_env, dc_level)

#Plot the PSD
psd_plot(x_demod, NFFT=1024, fs=fs)

#Downsampling to ~ 44.1kHz
x_down = signal.decimate(x_demod, 23)

#Write WaveFile
wavfile.write('demod.wav', 44100, x_down)
```

After the script was run, the power spectrum of the demodulated signal was generated as shown in Fig. 8, and the sound file of the demodulated audio signal was created.

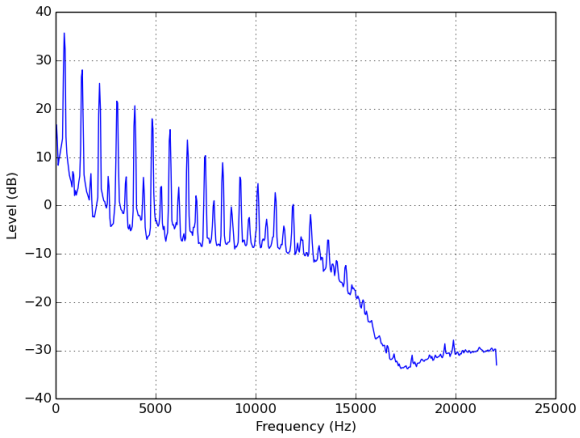

Fig. 8. The power spectrum of the demodulated signal

The comparison between the power spectrum of the original signal and the power spectrum of the demodulated signal are made in Fig.9. From the figure, both signals have exactly the same significant frequency components, but different power levels, showing that the wideband FM signal was successfully demodulated. This comparison shows one way of how the result can be illustrated using Python, students might do this for their reports.

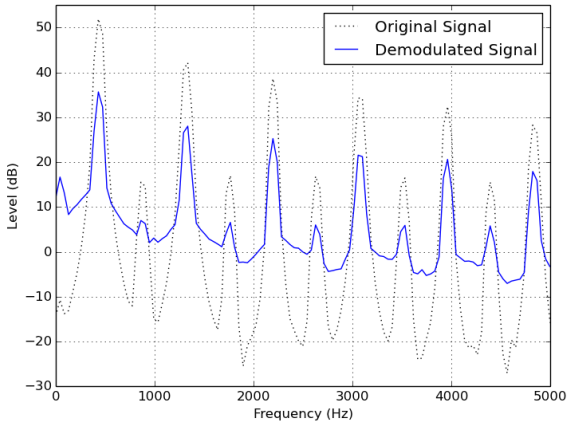

Fig. 9. The spectrum comparison between the demodulated signal and the original signal.

VIII. Suggested Signal Sources

The RTL-SDR dongles are only receivers. To use RTL-SDR as educational tools, a transmitter is needed in order to provide experimental signals to students. This section suggests a few signal sources that can be used as source signals for RTL-SDR dongles.

A. Electronic Circuit Transmitters

An electronic circuit can be built for generating signals of a particular modulation technique. Using electronic circuits as transmitters, is the best choice for cost advantages. The cost for building a modulation circuit is very small. An instructor can have a set of modulation circuits containing a few components, enough for undergraduate experiment.

B. Local Radio Stations

In terms of cost and convenience, using local radio stations as signal sources, is the best choice. No additional equipment is needed, however, there are problems in reliability and limited modulation scheme depends on the operated location.

C. A Professional-Grade Software Defined Radio System

Using a professional-grade software defined radio system as a signal source, is the best combination in all dimensions, except cost. One professional-grade software defined radio system can be used as a source for many RTL-SDR dongles and it can provide almost every kind of modulation, both analog and digital.

IX. Conclusions

In conclusion, this work proposes a way to organize a cost efficient software defined radio receiver for use as an educational tool for classroom demonstrations and student experimentations. An RTL-SDR dongle seems to be the most suitable hardware choice, in terms of cost, which provides a sufficient frequency range and bandwidth for educational purposes. Python is used as a programming environment because of its ease and its scientific computing libraries. RTL-SDR is linked to Python with the software framework proposed in this work. RTL-SDR, Python and the software framework combine together into a low cost experimental kit for experimenting with real radio signals from various sources. The typical workflow of using the kit is also demonstrated in the example experiment. It can be seen that a real radio signal can be captured and processed with a simple Python script. Integrating real world oriented activities into a curriculum, will motivate students to gain more understanding. This work makes all of these possible without spending money on expensive equipment sets.

References

[1] RTL-SDR. [Online]. Available: http://sdr.osmocom.org/trac/wiki/rtl-sdr. [Accessed: Jan. 24, 2014].

[2] Python. [Online]. Available: http://www.python.org. [Accessed: Jan. 24, 2014].

[3] Katz and J. Flynn, "Using software defined radio (SDR) to demonstrate concepts in communications and signal processing courses," in Proceeding of *the 39th Frontiers in Education Conference*, *Proc. 39th IEEE FIE*, Oct. 18–21, 2009, pp. 1-6.

[4] OsmocomSDR. [Online]. Available: http://sdr.osmocom.org/trac/. [Accessed: Jan. 24, 2014].

[5] Elonics Ltd., "Multi-Standard CMOS Terrestrial RF Tuner," E4000 datasheet, Aug. 2010.

[6] M. Lustig, "Slow-Scan TV over FM radio using software defined radio," a class project for EE123, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Nov. 16, 2012.

[7] MathWorks Inc., "RTL-SDR Support from Communications System Toolbox," [Online]. Available: http://www.mathworks.com/hardware-support/rtl-sdr.html. [Accessed: Jan. 24, 2014].

[8] Ettus Research. [Online]. Available: http://www.ettus.com. [Accessed: Jan. 24, 2014].

[9] Python Community, "Useful Modules, Packages and Libraries," [Online]. Available: https://wiki.python.org/moin/UsefulModules. [Accessed: Jan. 24, 2014].

[10] SciPy. [Online]. Available: http://scipy.org/scipylib/index.html. [Accessed: Jan. 24, 2014].

[11] NumPy. [Online]. Available: http://www.numpy.org/. [Accessed: Jan. 24, 2014].

[12] matplotlib. [Online]. Available: http://matplotlib.org/. [Accessed: Jan. 24, 2014].

[13] L.E. Couch, *Digital and Analog Communications Systems*, 4th ed. Macmillian, NY, 1993, pp. 252

[14] PT40 Manual, AKG Acoustics, Jun. 2005.

[15] *MM6 Owner's Manual*, Yamaha Corporation, Jan. 2007.

[16] A. B. Carlson and P. B. Crilly, *Communication Systems*, 5th ed. New York, NY: McGraw-Hill, 2009, pp. 240.

[17] MathWorks Inc., "Envelope Detection," [Online]. Available: http://www.mathworks.com/help/dsp/examples/envelope-detection.html. [Accessed: Jan. 24, 2014]