Software-defined radio for undergraduate projects

Conference Paper *in* ACM SIGCSE Bulletin · March 2007

DOI: 10.1145/1227504.1227414 · Source: DBLP

William Peter Birmingham
St. Vincent College
147 PUBLICATIONS **2,298** CITATIONS

Software-Defined Radio for Undergraduate Projects

William Birmingham
Grove City College
100 Campus Drive, Mailbox 3123
Grove City, PA 16127
011 724 458 3794
wpbirmingham@gcc.edu

Leah Acker
Grove City College
200 Campus Drive, Mailbox 971
Grove City, PA 16127
011 724 662 2393
ackerlc1@gcc.edu

ABSTRACT

Software-defined radio (SDR) is a self-contained embedded software system with hardware components and real-time constraints. **SDR is** the basis for many of today's wireless communications systems. Because SDR combines basic digital signal processing, circuitry, and software elements, it is the perfect project for multi-disciplinary, undergraduate teams. This paper explains both the SDR system created by CS and engineering undergraduates at Grove City College and how SDR can serve as a project for multi-disciplinary engineering teams. .

Categories and Subject Descriptors
C3 SPECIAL PURPOSE AND APPLICATION-BASED SYSTEMS, embedded and real-time systems

General Terms
Algorithms, Design

Keywords
Software-defined radio, pedagogy, multidisciplinary project

1. INTRODUCTION

Software-defined radio (SDR) is an important technology that underlies many modern wireless communication systems. For example, most cell-phone handsets and wireless routers are based on SDR technology. This technology is popular among computer scientists, with several interesting software projects underway, such as the GNU radio [1] and Flexradio's SDR 1000.

The idea behind SDR is to replace much of the electronics in a radio with software, thereby gaining flexibility with baseband waveforms, frequencies, and features. The software necessary to drive an SDR includes real-time signal processing, user interface, and a variety of other processing tasks. Therefore, an SDR is a self-contained embedded software system with real-time deadlines and hardware interfaces.

SDRs rely on some hardware, minimally a downconverter and A/D for receiving and a D/A and an upconverter for transmitting. While the hardware circuitry is well defined, it is not trivial to build because operating frequencies of radio are, naturally enough, in the radio frequency (RF) spectrum. While cell phones are typically in the GHz range, our applications are in the 0.1 MHz – 100 MHz range, specially, the amateur, public service, and commercial radio bands.

The combination of software and hardware makes SDR an excellent project for undergraduates. Junior and senior computer science (CS) students have enough background in programming to handle the software development; electrical engineering (EE) seniors have sufficient communications background to understand the basics of radio and signal processing, and circuit design; computer engineering (CE) seniors have sufficient skills in both programming and computer interfacing to integrate and test the hardware and software components. For those schools without EE or CE programs, SDR is still an excellent project testbed, as the hardware components can be purchased, thus making SDR completely a software project.

At Grove City College (GCC), we have an on-going design project in SDR. Presently, we are developing a general coverage SDR receiver capable of decoding AM with several popular baseband waveforms FM, SSB, and PSK/QPSK coming soon. In addition, we are also developing an SDR transmitter for these baseband waveforms to transmit on amateur radio bands.

In this paper, we describe the AM radio system \and how we organized the project and student teams.

2. SDR ARCHITECTURE

The SDR architecture contains hardware and software components. In order to effectively design and implement this technology, a team of six undergraduates (two CS students, two EE students, and two CE students) divided into three pairs based on their field of study. The entire group met weekly so that each pair understood the others' work. This fostered greater cross-disciplinary understanding and ensured coherence in the project as a whole. Figure 1 shows the physical components of the SDR architecture.

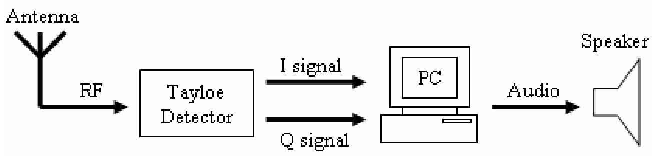


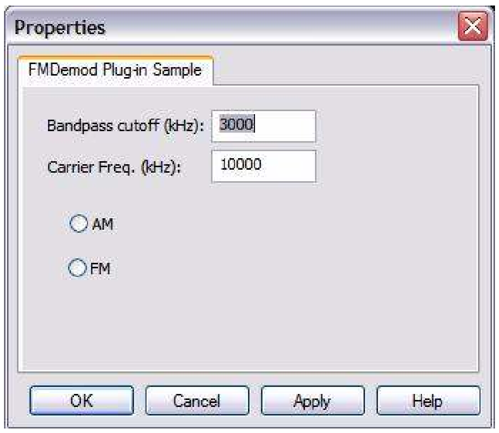
Figure 1: The physical components of the SDR architecture for a receiver

The SDR requires a hardware “downconverter” that converts the RF signals at the received frequency into two parts: the I signal, which is in-phase, and the Q (quadrature) signal, which is 90 degrees out of phase. The students built a Tayloe detector.[1] The Tayloe detector is a simple, inexpensive circuit that provides a complete quadrature downconverter with only a few integrated chips [2, 5]. The I and Q signals produced by the Tayloe detector feed directly into the sound card of the PC, where they are converted from analog to digital signals.

Because the CE students could not get the Tayloe detector to function as they had simulated, the I and Q signals were produced using FlexRadio SDR-1000, a commercial SDR. Section 3.3 explains the attempt at creating a Tayloe detector in greater detail.

Feeding the I and Q signals to the sound card is more complicated than it might seem because the PC used for the project used does not have stereo input; thus, the students used a Philips USB audio device with line-in stereo to put the I and Q signals into the sound card. As described in the next section, the students used a software development kit (SDK) from Microsoft for Windows Media Player 10. The SDK, which is available on the Microsoft website, let the students write digital-signal processing (DSP) software plug-ins for the Media Player. The user-interface for our plug-in lets the user specify information about the signal, such as the carrier frequency, lower limit of the signal (band-pass cut-off on UI), and whether the signal is AM or FM. Figure 2 is a screenshot of this GUI.

Given the sampling rate of the soundcard and the types of waveforms we are converting, the plug-in must process each digital sample within 250 mS. This hard, realtime constraint must be met to keep the sound quality good. Meeting this constraint is an important factor in the design of the plug-in software.



Bandpass cutoff (kHz): 3000

Carrier Freq. (kHz): 10000

AM

FM

Figure 2: GUI that allows for user-input specifying input signal properties

[1] US Patent #6,230,000

Figure 3 is a block diagram giving the demodulation process. While I and Q are in the time domain, their frequency characteristics have been modulated to conform to a baseband waveform. In order to demodulate these signals, they must be converted to the frequency domain using a Fast Fourier Transform (FFT) with 4096 bins [2, 4]. Next, the signal is low-pass filtered to extract the needed frequency components and to remove redundant information. Then, the FFT bins are shifted to prevent attenuation at the ends of the signal. Then, the signals are high-pass and low-pass filtered so that only the desirable frequencies in the FFT remained [2,4,5]. After this, the FFT is shifted back so that it is in the correction position to reflect the starting data. This shift is necessary to prevent distortion around the beginning and end of the spectrum [4]. Finally, an inverse Fast Fourier Transform (IFFT) is performed on the data to get it back to time-domain I and Q components.

All these operations are done in software operating under the 250 mS constraint.

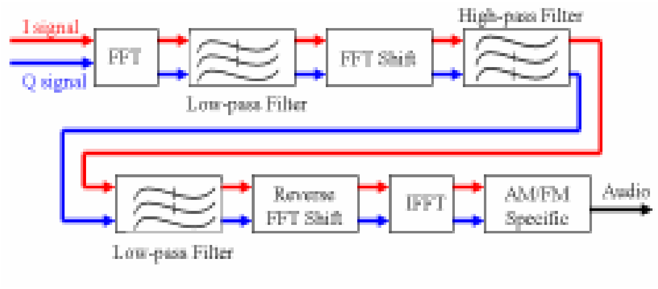


Figure 3: Demodulation process for AM and FM signals

At this point, the data is ready for AM or FM specific demodulation. The instantaneous magnitude of the sound signal is the result of demodulated AM [2]. This is also called the envelope of the signal. The equation for AM demodulation is shown in Equation 2.1.

$M_t = sqrt\ (I_t\ \hat{}\ 2 + Q_t\ \hat{}\ 2)$ *(Equation 2.1)*

Where I_t is the demodulated I signal with respect to time, Q_t is the demodulated Q signal with respect to time, and M_t is the magnitude with respect to time

The students have only successfully implemented AM; however, the same approach works for FM demodulation as well. The only difference is the final equation. Instead of calculating the instantaneous magnitude of the sound signal, the students calculate the instantaneous phase [2]. The equation for FM demodulation is show in Equation 2.2.

$\Phi_t = tan^{-1}(Q_t / I_t)$ *(Equation 2.2)*

Where I_t is the demodulated I signal with respect to tme, Q_t is the demodulated Q signal with respect to time, and Φ_t is the phase angle with respect to time

Depending on whether AM or FM signals are being demodulated, the magnitude or phase are calculated by the Media Player plug-in. The plug-in sends its output to the sound card, which plays the demodulated radio signal on the system speakers. The students are currently working on transmitting using our SDR, but the working components only receive at this time.

While the SDR works in real time, much of its development work was done using pre-recorded AM and FM samples in MATLAB. MATLAB does not readily allow for real-time implementation; however, its built-in functionality and easy command-line prompts make it a natural choice for students who are testing out different demodulation schemes.

3. PEDAGOGY

We encourage our students to participate in projects outside of normal class assignments. There are two reasons for this: we want our students to gain additional knowledge in areas related to CS in which they are interested, and we want our better students to have research experience to prepare them for graduate school. Thus, CS students participate in projects that extend our core curriculum and contain at least some research. One of the big draws for these types of projects is that students get to work on things in which they are interested, and they get to drive the direction the projects take.

We also believe that it is important for students to gain experience in multi-disciplinary projects. Much of the interesting technology being developed today—and for years to come—will require expertise from many different areas. For example, mobile electronics require many engineering disciplines in addition to computer science (along with a host of experts outside the engineering disciples!). SDR provides a perfect project around which to organize such multi-disciplinary teams.

3.1 CS Students

By the start of their senior year, GCC CS students have taken four courses directly dealing with programming, not including algorithms, data structures, and software engineering. These courses range from introductory c++ programming to advanced object-oriented design and large-scale programming.

In addition, many CS students gain experience with the Windows programming model (in the second programming course), as well as DirectX and COM in the game-programming course sequence. These students, by and large, are capable of developing large Windows programs that have to meet some hard real-time constraints, particularly those for gaming applications.

SDR makes students use all the programming and algorithms knowledge they picked up over their years of study. Moreover, the requirements of the project extend those typically given in other course projects: the radio must operate under tight real-time constraints, runs in Windows, and requires user interaction. The SDR software, as mentioned in Section 2, must be able to process the input on the order of hundreds of milliseconds. Furthermore, the RF must be read in real time from the computer's sound card, and then the processed signal must be returned to the sound card to produce audio to play through the speakers.

Capturing audio from the sound card turned out to be a challenge. The students' experience was limited to opening MIDI and WAV files in their game programs and then using Direct Music and Direct Sound to play the sound through buffers. This system works well when playing files, but does not work, or at least is not easily implemented, when capturing audio through the sound card in real time.

The students decided to package their code as a Windows Media Player 10 plug-in. The advantage to this approach is that Media Player handles much of the task of getting input in real time from the sound card, and it presents a nice interface to the user through a skin. Eventually, custom skins will be developed by the students.

Media Player 10 is well documented, and with some careful work, the students were able to create a plug-in for AM demodulation. As we code different demodulation schemes, they will be added to the plug-in. Figure 4 is a screenshot showing the SDR demodulating an AM signal on the 75m amateur radio band.

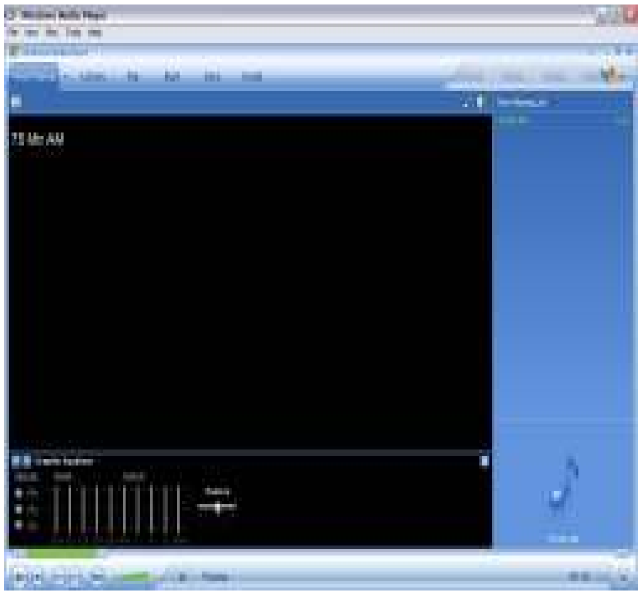


Figure 4: Screenshot of the SDR plug-in tuned to the amateur radio 75 M band decoding an AM signal.

The CS students gained additional valuable experience with Media Player and Windows. Furthermore, they learned how to program in c# and gained experience with a variety of SDKs.

The implementation of the SDR requires significant signal processing. The general ideas of algorithms the students used all came from Youngblood [2-5], an excellent series of papers on implementing SDR. As we describe in the next section, the EE students developed all the signal processing routines and proved them in MATLAB. This was necessary because implementing Youngblood's algorithms in our environment required significant experimentation and some handcrafting.

Once proven, these routines were then translated into c# by the CS students. The AM decoding code is surprisingly simple because most of the hard signal processing is done through a set of DSP routines written by Intel. These routines handled such functions as the Fast Fourier Transform, the inverse Fast Fourier Transform, and filtering. In addition, the Intel routines are very fast, designed to work in realtime.

The process of proving routines in MATLAB and translating them into the Windows environment required significant effort, but was invaluable. The MATLAB routines were the gold standard to test the c# code, and the ease of prototyping DSP routines in MATLAB makes it a convenient tool for testing out ideas.

3.2 EE Students

During their senior year, EE students build upon their signal-processing background with additional courses in communications and control theory. At this point, they have also completed 18 credit hours of mathematics and at least two computer programming courses, both in c++. Therefore, senior EE students have a enough background to successfully tackle the signal processing required for SDR, which can be mathematically intense.

Additionally, their junior and senior experimental electrical engineering labs include radio-related exercises, such as antenna design, modulation, filtering, transmission, and distortion. The senior EE students feel comfortable with radio and understand the physical principles behind transmission and modulation.

EE students were primarily responsible for developing signal-processing techniques and the methods for quadrature modulation. While they understood the concepts behind signal processing and modulation, literature reviews proved challenging, as they often do for undergraduates exploring advanced signal processing.

Consequently, the students used MATLAB as a tool to better understand the algorithms and find solutions to various often low-level implementation issues. This let them to attempt to reproduce methods suggested in the literature using a forgiving software platform and to modify these methods for our specific project. Because many functions, such as the FFT and the IFFT are built into MATLAB, the students did not have to code these functions themselves until they were sure they worked correctly for their problem.

MATLAB is a resource intensive programming platform so it did not make sense for real-time implementation. Additionally, it required additional software packages to allow for direct sound card access. Because of these impediments, recordings were used to test the MATLAB code. Another advantage of using recordings in MATLAB, along with ease of use, is that they provided a consistent sample against which the students could compare different signal processing strategies to determine which was the most effective in producing clear, crisp demodulation.

3.3 CE Students

The CE students that participated in the project were juniors, who had a good background in analog and digital circuits. The simple electronics used for the Tayloe detector were easily understood by the students; the only vexing section of the detector was the RF mixer. CE students do not have much experience with high-frequency RF signals (although they are familiar with high-speed clock signals in computers). Thus, some of the subtle choices for component values and construction details were difficult for them to understand.

Junior CE students also have good background in computer interfacing, having interfaced a variety of devices to computers in their lab courses. Thus, the CE students were comfortable interfacing the Tayloe detector to the PC. The circuit design the students used is compatible with PC signal levels, so interfacing was simple.

CE students were responsible for building the hardware and interfacing it to the computer sound card. As they are good engineers, they simulated the circuit in PSpice, but did not account for all the high frequency effects. One of the biggest mistakes they made—a classic mistake—is treating the wires as perfect conductors and not accounting for the induction and capacitance that will affect the RF signals. Thus, the simulation worked, but the circuits they built did not.

The labs projects that the students build in their computer interfacing classes work at relatively low clock rates. Thus, breadboarding and wirewrapping methods for prototyping those circuits (usually) work fine. At high frequency, however, wire length and wire routing are very important. This was a lesson that the students learned well. One of the early attempts at building the circuit resulted in a detector perfectly built to pick up noise from fluorescent lights and nearby signal generators. This is a common problem for first-time radio designers, and provides "learning," albeit not pleasant, experience.[2]

The project team members came to the conclusion that prototyping the circuit was too difficult. Instead, the students used the CS department's FlexRadio SDR-1000, which is a commercial SDR. The SDR-1000 has a box that produces the I and Q signals for a PC sound card. This box was used by both the EE and CS students for their work.

3.4 Project Organization

The CE, EE, and CS teams met weekly with the faculty project advisor (the first author). In these meetings, we discussed both progress and various technical problems and how to solve them.

One of the interesting experiences for the CS students was working in the EE labs. Here, they got to hang probes on circuits and make measurements. While our CS students are comfortable with building computer systems, they rarely, if

[2] The first author remembers slaving for hours debugging an op-amp circuit in undergraduate electronics lab that did nothing but pickup the local high-power commercial AM station (KDKA).

ever, directly debug hardware using logic analyzers, signal generators, etc. They gained an appreciation for how much is involved with getting digital electronics to work.

4. RESULTS

While the undergraduates involved with this research have successfully received AM radio transmissions in real-time, this project continues to grow as the students expand their areas of consideration and research.

4.1 Results to Date

We currently can receive AM signals using the FlexRadio SDR 1000 as the Tayloe detector and demodulate them using signal processing techniques implemented by the group. The code that the students wrote for this purpose has been tested as well. The students tested the system by receiving an AM amateur transmitted by another group member across several amateur radio HF bands. This test allowed them to determine whether our demodulation was correct for AM receiving through a range of carrier frequencies.

4.2 Areas of Ongoing Research

While AM reception works correctly, the team continues to work on FM and SSB reception, as well as transmission of all three modulations. Once SSB is working, will the students will integrate existing software for digital transmission using PSK 31. Because FM and SSB require alternations in the demodulation techniques, the EE students are researching, developing, and testing demodulation routines in MATLAB. Once they have successfully, demodulated FM and SSB in MATLAB, they will help the CS students to create that functionality in the Media Player plug-in.

After that, the students will work on developing both the hardware and software to transmit AM, FM, and SSB signals using the Media Player plug-in. As a transmitter, the system will support all amateur radio bands from 160 meters to 6 meters (Youngblood, 2002).

4.3 Spin-off Projects

At Grove City College, the SDR has become a popular research project, and students other than the six on the original SDR team have become involved with SDR technology.

4.3.1 SDR and public safety

A team of electrical and computer engineers have adopted SDR technology as part of their capstone senior design project. As the SDR team makes progress toward increased receiving capability and transmission, this senior design team will take the improved SDR capability and apply it to public safety. This year-long project will work to develop cheap, PocketPC-based SDR systems for public-safety applications.

Creating these "portable" SDR would allow emergency personnel to communicate on the amateur radio bands during natural disasters. Because most communications systems other than amateur radio go down during hurricanes, floods, and other national emergencies, SDR could allow emergency personnel to gain the information they need in a dependable fashion.

We initially explored SDR for Pocket PCs last year, but the students opted to first get the system running on a regular PC and then to apply the technology to its smaller counterpart. They will continue to work toward the Pocket PC application in the coming year.

4.3.2 SDR and "High-Definition" Radio

We are developing addition plug-ins for commercial radio broadcasts of digital AM and FM, commonly called "high-definition" radio. These new baseband waveforms are intended to give AM radio the quality of current FM stations and to give FM stations the quality of CDs.

These waveforms use a baseband waveform entirely different from current AM and FM waveforms. Using on our library of SDR software, we building the decoders for these waveforms. This is a pure software project using the same downconverting hardware as the other SDR projects.

5. SUMMARY

Software-defined radio provides undergraduate students with the opportunity for multidisciplinary research at Grove City College. The project technology underpins some of the fundamental systems used in most of today's wireless communications systems, and it reinforces the classroom knowledge gained by computer science, electrical engineering, and computer-engineering students.

The current version of theSDR has successfully received AM signals, and it continues to engage undergraduates in active research. Additionally, the project has sparked the interest of additional students who hope to explort SDR technology in other design projects.

One of the great things about the project is that it is fun. Students get to see theory put into practice, and they get to build cutting-edge equipment. Moreover, for those students with amateur-radio licenses (almost all the students on the project have them), they get to mix their hobby with their school work.

The SDR project is a good segueway for the students as they enter the workforce or prepare for graduate school. It shows them a clear application of the concepts they have learned in class, and it gives them an opportunity to work in a multidisciplinary team.

6. ACKNOWLEDGEMENTS

We would like to acknowledge the following students who participated in the SDR research at Grove City College: Adam Jenkins, Brendan McCommas, Stuart McCommas, Matthew Chambers, and David Gottl. Additionally, we would like to thank Grove City College for its financial support. Finally, we are grateful for the ICs provided to our team by National Instruments, and for technical help provided by FlexRadio Systems.

7. REFERENCES

[1] Blossom, E. 2004. Listening to FM radio in software, step by step. *Linux J.* 2004, 125 (Sep. 2004), 1.

[2] Youngblood, Gerald. “A Software-Defined Radio for the Masses, Part 1,” *QEX*, July/Aug 2002, pp. 1-9.

[3] Youngblood, Gerald. “A Software-Defined Radio for the Masses, Part 2,” *QEX*, Sept/Oct 2002, pp. 10-18.

[4] Youngblood, Gerald. “A Software-Defined Radio for the Masses, Part 3,” *QEX*, Nov/Dec 2002, pp. 27-36.

[5] Youngblood, Gerald. “A Software-Defined Radio for the Masses, Part 4,” *QEX*, Mar/Apr 2003, pp. 20-31.