

Implementing Software Defined Radio – a 16 QAM  
System using the USRP2 Board

Project Proposal

**Patrick Ellis & Scott Jaris**  
**Dr. In Soo Ahn & Dr. Yufeng Lu**  
**December 14, 2010**

## Project Summary

Software Defined Radio takes what used to be physically ingrained within the radio and provides a variety of hardware and software alternatives that add to it more flexibility and functionality. This project looks to implement a 16QAM communication system using a baseband receiver and transmitter and two USRP2 boards. It will show the benefits of Software Defined Radio (SDR) and will demonstrate the power, flexibility, and advantages that it gives the designer through the ease of modification that it provides. This project does, however, require a great amount of new, unfamiliar software which may prove to be one of its biggest challenges.

## Complete Description

### Background

Software Defined Radio, as defined by the SDR Forum is a “Radio in which some or all of the physical layer functions are software defined.” Together, the above components will allow the realization of a SDR 16QAM communication system. Each component will be discussed in detail and then the overall system will be explained in the process.

### Goals

The progressive goals of the project are listed as follows.

1. Design 16QAM transmitter and receiver modules using GNU Radio companion (no USRP2 board or noise used at this point).
2. Add simulated noise resources to the communication channel within the prior system to introduce fading and multipath effect.
3. Design Decision Feedback Equalizer system to fully recover the transmitted data at the receiver end.
4. Explore the capabilities of GNU radio and boards by small side projects such as an FM receiver in order to more fully encompass the capabilities and span of SDR.

## Equipment List

The entirety of this project will consist of the following software and equipment:

1. Two USRP2 Boards – Offers a Xilinx Spartan 3 2000 FPGA, 14-bit AD and 16-bit DA converters, digital up/down converters, gigabit Ethernet interface, external memory, and SD card.
2. BasicTX and Basic RX Daughterboards – Gives access to all of the signals on daughterboard interface and provide AD/DC outputs with no mixers, filters, or amplifiers.
3. GNU Radio Companion – an open-source software development similar to Simulink in nature whose applications are written in Python and the signal processing paths implemented in C++.
4. Python Interpreter – software to write and debug code written in Python.
5. Ubuntu – Version 8.04 or greater will provide the operating system required for GNU Radio and the USRP2 to function properly.

## USRP2

The USRP2 is a high speed Ethernet-based board that is specifically built by *Ettus Research* for software radio. Drivers are open source and there is a variety of free software to integrate such software toolkits as GNU Radio. The USRP2 is built with SDR in mind as it is made to be highly flexible with designer's specific needs regarding frequency bands, connectors, and different daughterboards.

### Features:

- Xilinx Spartan 3-2000 FPGA
- Gigabit Ethernet interface
- Two 100MS/s, 14 bit, AD converters
  - LTC2284
  - 72.4dB SNR and 85dB SFDR for signals at Nyquist Frequency
- Two 400MS/s, 16 bit, DA converters
  - AD9777
  - 160 MSPS w/o interpolation
  - Up to 400 MSPS with 8x interpolation

- SD card reader
- MicroBlaze Processor Core
- aeMB software implementation of MicroBlaze

### BasicTX and BasicRX Daughterboards

The BasicRX board has 4 subdevices. Subdevice A is a real signal on antenna RXA, subdevice B is a real signal on antenna RXB, subdevice AB is a quadrature subdevice using both antennas (IQ), and subdevice BA is a quadrature subdevice using both antennas (QI). The Basic TX board is the exact same but with “TX” instead of “RX.” The boards do not allow any tuning of elements or gains, but the ability to up-convert signals greater than the Nyquist rate of the DAC can be done with aliasing. External RF Hardware must be used with these boards and the TVRX antennas.

#### Features:

- Serve as RF front end
- Modulates output signal to higher frequency
- Takes away input signal carrier frequency
- BasicRX is a 1-250MHz Receiver
- BasicTX is a 1-250MHz Transmitter

### GNU Radio, Python, and Ubuntu

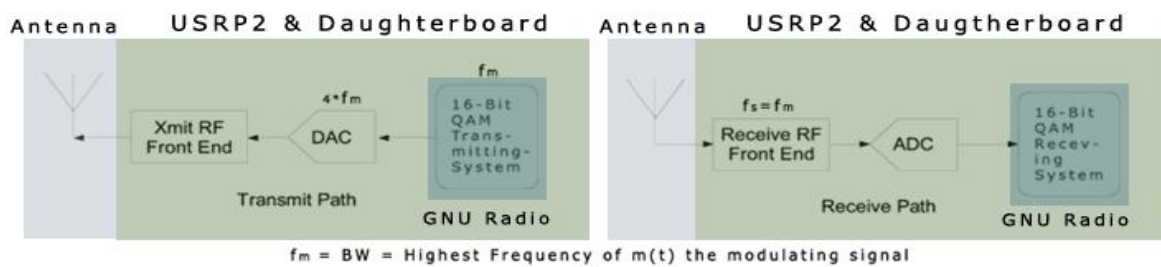
GNU radio is a free software toolset for building and implementing SDR. It is a signal processing package which provides a variety of signal processing blocks as well as the ability to develop ones own blocks. The signal processing blocks are written in C++ and are accessed, called, and implemented by Python; much in the manner an m-file would control a Simulink file. GNU radio, when implemented with the USRP2 must be on a Linux-based system.

### High Level Block Diagram

Figure 1 shows the high level over-all system block diagram that shall be implemented within the final objective. As can be seen, the entire communication system is software-based and only the ADC/DAC and RF are hardware-based. The 16QAM system, or

other systems, can be quickly changed through programming the Spartan3-2000 FPGA device on a USRP2 board. Figure 1 clearly points out the relationships between the USRP2, Daughterboards, and GNU Radio. GNU radio initiates the interfacing between the USRP2, daughterboards, and the signal processing that it performs. The daughterboards perform converting (up or down as needed) and the antenna transmits or receives the desired information.

Figure 1 – High Level Block Diagram



## 16 QAM Communication System

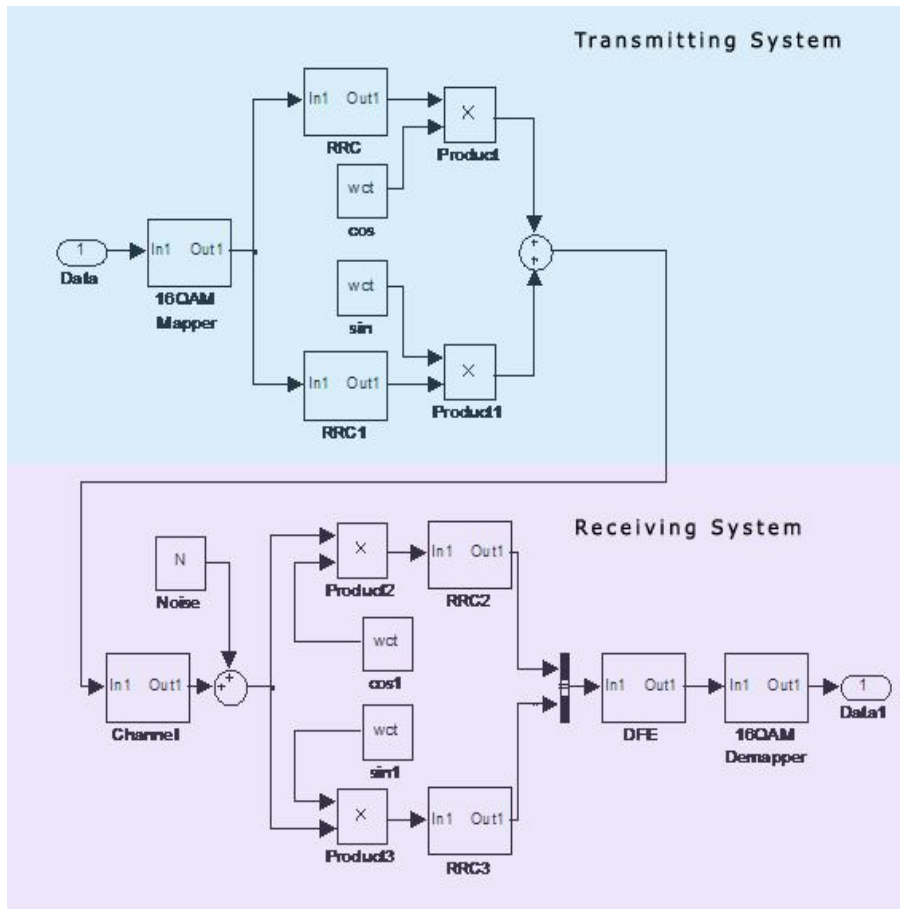
Figure 2 shows the general flowchart of a 16QAM system which shall be implemented on the USRP2 board via GNU Radio Companion. The 16QAM transmitting and receiving systems are shown on the top and bottom of Figure 2 respectively. The RRC blocks represent root raised cosine filters and the DFE is a decision feedback equalizer.

The transmitting side works as follows: The 16QAM mapper takes in the data and selects 4 bits. Two sets of 2 bits are selected and are mapped to one of four values: -3, -1, 1, or 3. One symbol contains the I component and the other the Q component, the values of which are determined from the gray coding scheme. Each symbol stream is sent through the RRC block to decrease bandwidth. They are then used to modulate their respective carrier waveforms and added to form the transmitted signal.

At the receiving end, the transmitted signal is split once again into its I and Q components and the modulating signals are brought back to baseband and filtered by the RRC filters. A decision feedback equalizer (DFE) is then used to recover the signal

and compensate for any noise, fading, or multipath effect that can be added in the channel during transmission. The signal is then decoded through a look-up table (LUT) and the original data is received.

Figure 2 – 16QAM Block Diagram



### System Specifications

This system shall be able to transmit and receive within the frequency range of the Basic TX and RX daughterboard, which is 1 MHz to 250 MHz. The generated and received signals shall use the DAC and ADC respectively; on the USRP2 board. The signal shall then be processed inside the 16QAM system. The carrier frequency  $f_c$  shall be at least 4 times smaller than the USRP2 overall system clock, which is 100 MHz. As a result the  $f_c$  maximum value shall be 25 MHz.

The 16QAM system has the following constraints: The sampling frequency of the receiving system shall be greater than or equal to twice the bandwidth, or twice  $f_c$  to satisfy Nyquist's Theorem. The QAM system shall minimize bit error rate. Once success has been reached on transmitting a variety of signals (random bit stream, picture, audio), flexibility and system performance shall be tested by introducing multipath effects.

As familiarity increases with GNU radio, Python, the USRP2 and its daughterboards, more specifics in regards to particular bandwidths, baud rates, ACD/DAC conversions, and power, more detail-oriented objectives will be able to be made.

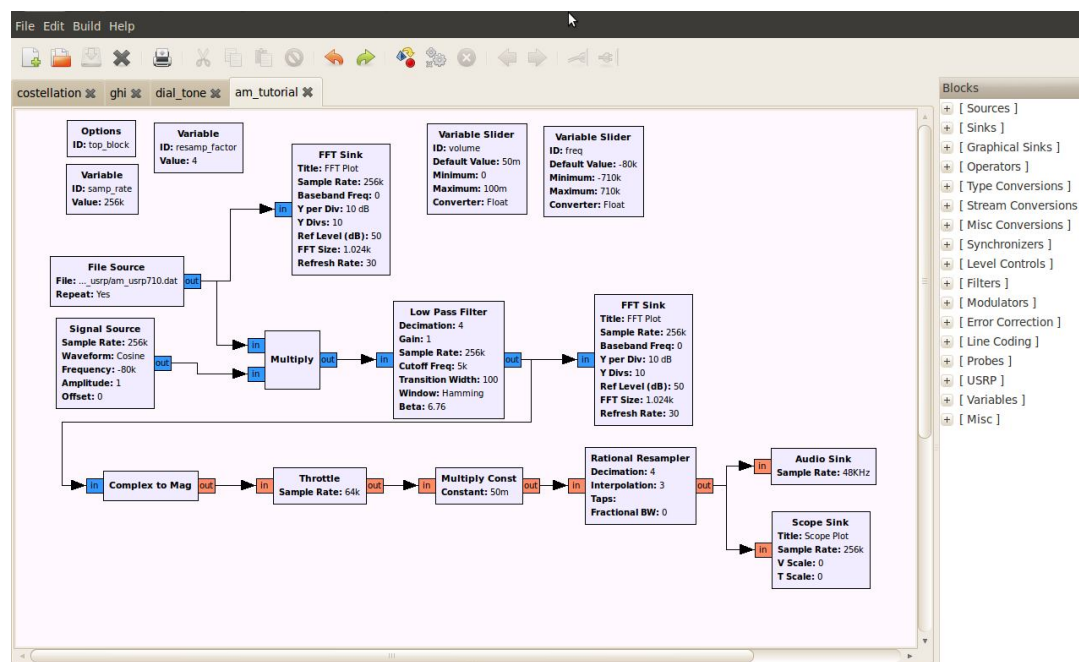
## **Preliminary Work:**

### **Tutorials:**

#### **GNU Radio Companion (GRC):**

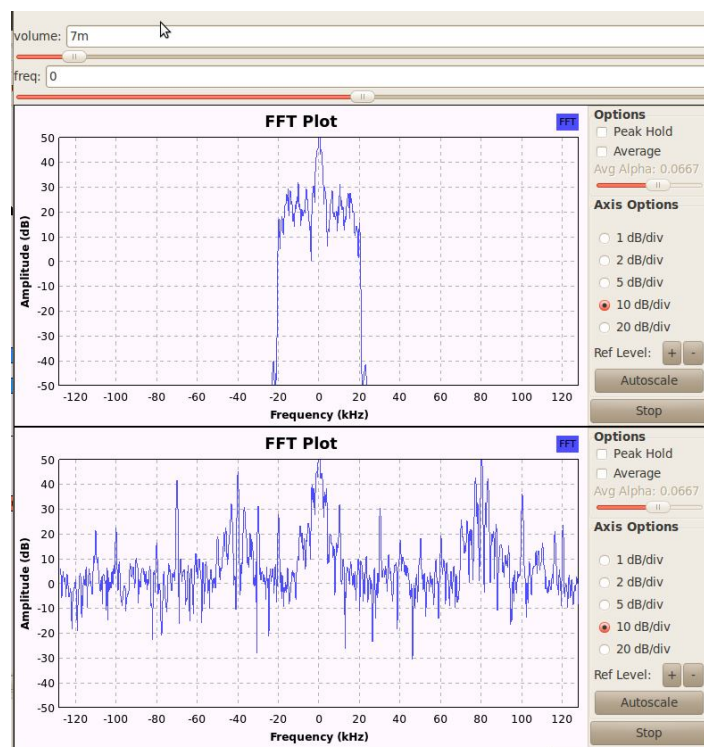
GRC is an open source program that consists of signal processing blocks to allow easy construction of various systems using the GNU Radio libraries. In order to familiarize with this program, several tutorials provided by Sharlene Katz were completed. Figure 3 shows the end product of one tutorial in which an AM receiver was constructed using a file source as the input. Variable sliders are used to tune to the desired frequency and adjust the volume. Various parameters such as sampling rate and filter cutoff frequency can be set as desired. This demonstrates power of SDR.

Figure 3 – Flow Graph for an AM Receiver



The FFT outputs of the file source and the filtered signal for the selected frequency are shown in Figure 4 and provide a real time simulation visual for the user.

Figure 4 – FFT Sink Outputs of AM Receiver Flow Graph

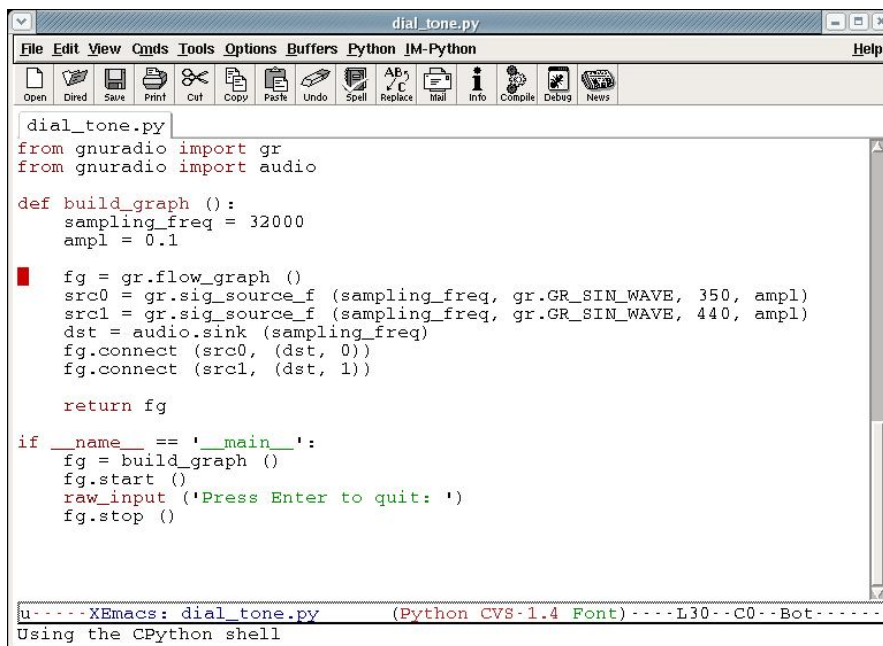




## Python:

Since Python programming language is used to link the GNU Radio signal processing blocks and will be have to be used to create customized blocks, a simple tutorial was completed. The tutorial constructed a system which produced a basic dial tone using two sinusoids of separate frequencies. The code is shown in Figure 5 and is equivalent to a flow graph sending two sinusoids to an audio sink in GRC.

Figure 5 – Python File for Generation of a Dial Tone



```
dial_tone.py
from gnuradio import gr
from gnuradio import audio

def build_graph ():
    sampling_freq = 32000
    ampl = 0.1

    fg = gr.flow_graph ()
    src0 = gr.sig_source_f (sampling_freq, gr.GR_SIN_WAVE, 350, ampl)
    src1 = gr.sig_source_f (sampling_freq, gr.GR_SIN_WAVE, 440, ampl)
    dst = audio.sink (sampling_freq)
    fg.connect (src0, (dst, 0))
    fg.connect (src1, (dst, 1))

    return fg

if __name__ == '__main__':
    fg = build_graph ()
    fg.start ()
    raw_input ('Press Enter to quit: ')
    fg.stop ()

u-----XEmacs: dial_tone.py (Python CVS-1.4 Font)-----L30--C0--Bot-----
Using the CPython shell
```

## 16QAM System Construction & Simulation

In the progress of completing goal number one of simulating the 16QAM system with no noise or USRP2 board, the system in Figure 6 has been created in GRC. The purpose of this system is to show that a stream of random binary data can be successfully sent and received using the 16QAM method. Although this system is not yet functioning correctly due to an unknown error created in the QAM Demod block, the input data and the constellation plot at the output of the QAM Mod block are shown in Figure 7.

Figure 6 – Flow Graph of the 16QAM System without Noise

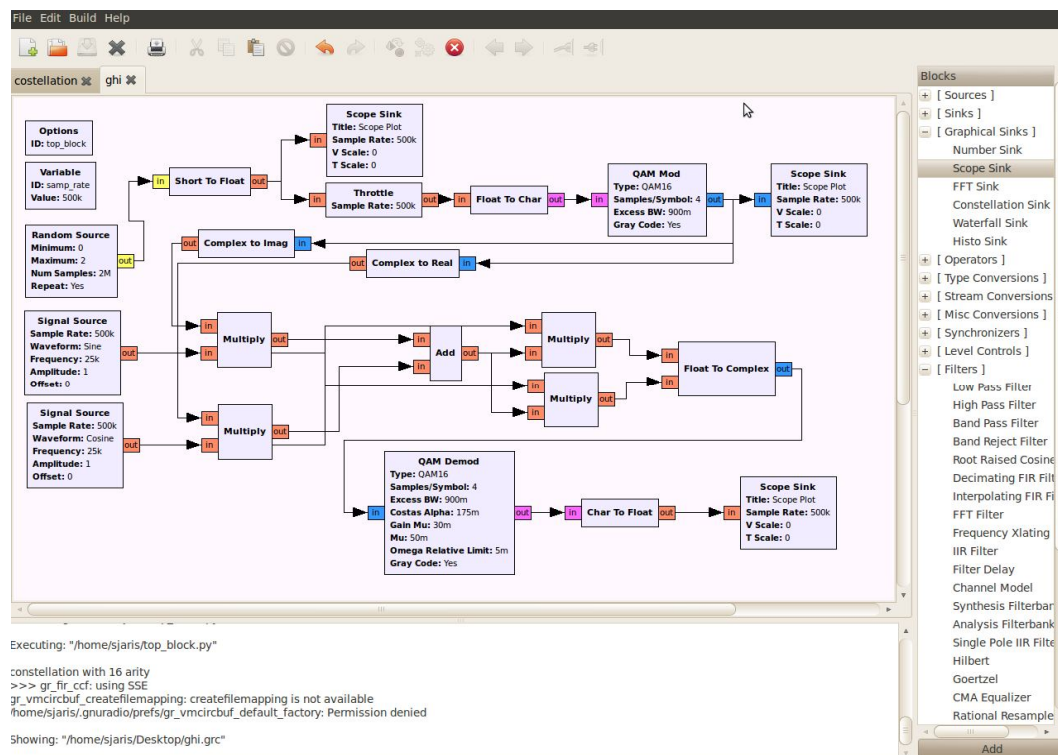
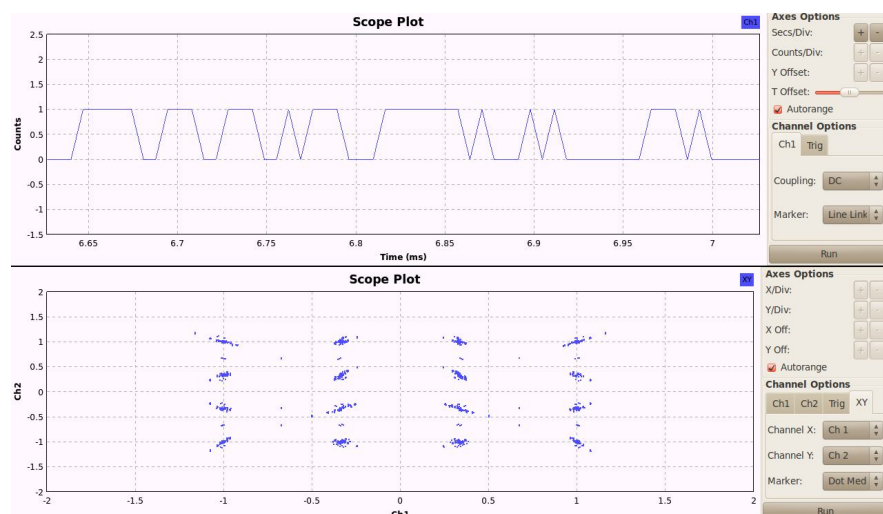


Figure 7 – Random Source Binary Data and 16QAM Constellation



The plots in Figure 7 demonstrate how the transmitting signal can be created with the use of only one main block. The block contains the creation of the I and Q components and the filtering process using RRC filters. The small imperfections in the constellation plot are due to this filtering.

## Preliminary Schedule

Dates	Task
1/20 - 1/26	Create and simulate a 16 QAM system using GNU radio companion
1/27 - 2/2	Continue creating system
2/3 - 2/9	Introduce fading and multipath effect. Create DFE for signal recovery
2/10 - 2/16	Continue creating recovery system
2/17 - 2/23	Continue creating recovery system
2/24 -3/2	Continue creating recovery system
3/3 - 3/9	Explore the USRP2 board by creating simple FM receiver and other systems
3/10 - 3/11	Continue exploring USRP2 capabilities
3/21 - 3/23	Continue exploring USRP2 capabilities
3/24 - 3/30	Continue exploring USRP2 capabilities
3/31 - 4/6	Test the 16 QAM system by sending various data between two USRP2 boards
4/7 - 4/13	System debugging and documentation
4/14 - 4/20	System debugging and documentation
4/21 - 4/27	System debugging and documentation

Note: Not shown on the Preliminary Schedule are our planned efforts over winter break to complete two tasks. One is the identification of the parameters given to us to use the signal processing C++ blocks in GNU radio. No documentation exists to tell us what these parameters represent and an in depth look at the Python, then C++ files along with communication textbooks will hopefully allow us to identify these parameters. Secondly we will be researching the receiver system – primarily the Decision Feedback Equalizer portion in an effort to combat the complexity and learning curve when implementing it in the spring semester.

## Bibliography

- [1] Couch, Leon W. *Digital and Analog Communication Systems*. Upper Saddle River, NJ: Pearson Prentice Hall, 2007. Print.
- [2] "Exploring GNU Radio." *The GNU Operating System*. Web. 11 Nov. 2010.  
<<http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html>>.
- [3] Hanzo, Lajos, Thomas Keller, Soon Xin. Ng, and William Webb. *Quadrature Amplitude Modulation: from Basics to Adaptive Trellis-coded, Turbo-equalised and Space-time Coded OFDM, CDMA and MC-CDMA Systems*. [Piscataway, NJ]: IEEE, 2004. Print.
- [4] Katz, Sharlene. "CSUN/EAFB SDR Project." *California State University, Northridge*. Web. 07 Oct. 2010.  
<[http://www.csun.edu/~skatz/katzpage/sdr\\_project/sdrproject.html](http://www.csun.edu/~skatz/katzpage/sdr_project/sdrproject.html)>.