

# THE GNU SOFTWARE RADIO TRANSCEIVER PLATFORM<sup>†</sup>

Alex Betts (National Center for Supercomputing Applications (NCSA) / University of Illinois at Urbana-Champaign (UIUC), Champaign, Illinois, USA, abetts@shout.net); Matt Hall (NCSA/UIUC, mahall@ncsa.uiuc.edu); Volodymyr Kindratenko (NCSA/UIUC, kindr@ncsa.uiuc.edu); Meenal Pant (NCSA/UIUC, mpant@ncsa.uiuc.edu); David Pointer (NCSA/UIUC, pointer@ncsa.uiuc.edu); Von Welch (NCSA/UIUC, vwelch@ncsa.uiuc.edu); Paul J. Zawada (NCSA/UIUC, zawada@ncsa.uiuc.edu)

## ABSTRACT

*GNU Radio* is a code base of free software that performs signal processing using a personal computer and freely available Radio Frequency (RF) receiver front-end designs. The *GNU Radio* receiver is an ideal platform for learning and experimenting with Software Defined Radio (SDR) concepts. Recent efforts at NCSA have extended the *GNU Radio* receiver design into a 900MHz narrowband software defined radio transceiver. Our SDR transceiver is a useful tool for development of front-end hardware, algorithms, protocols, performance estimation, and operational visualization. In this paper we describe the extended hardware and software architecture for the SDR transceiver and describe a number of applications we have developed for it. One such application is SDR operational visualization software that serves as an educational tool for introducing the concepts of radio communications to novice users. Another new application is a reconfigurable communication protocol stack that includes network transport protocol layer, security layer, end-user application interface layer and a radio management layer which utilizes the SDR transceiver as the underlying communication fabric. In addition, we describe our efforts to prototype various user authentication mechanisms, such as voice pattern recognition, for unlocking specific application capabilities for specific users.

## 1. INTRODUCTION

The *GNU Radio* software package is a quite capable digital signal processing library freely available and produced by a group of volunteers, principally Eric Blossom and Matt Ettus. [1] The routines that comprise GNU Radio provide all of the signal processing building blocks (mixers, filters, etc.) required to build a radio in software. What is lacking however, is an assortment of hardware for specific radio

applications, especially for transmitting in license-free radio bands. Much of the candidate hardware for new GNU Radio applications tends to be quite expensive, discouraging potential users from experimenting with the package and constructing radios for real-world experimentation. Furthermore, the package itself includes some demonstration programs but lacks an overall software platform for building extensible software defined radios.

The goal of the NCSA project described in this paper was to build a prototyping environment which could be used by researchers to develop SDR concepts on simple and/or embedded hardware. It is important to note that the NCSA effort is not intended to compete with projects like the Joint Tactical Radio System (JTRS). The aim is to introduce SDR concepts for simple applications running on modest hardware. Once comfortable working with this platform, the user can move up to more sophisticated hardware designs to achieve higher performance.

## 2. A 900 MHZ TRANSCEIVER IMPLEMENTATION

In order to start experimenting with transmitter applications based on *GNU Radio* routines and to encourage others to do the same, NCSA embarked on the development of a low-cost board set to provide basic radio functionality at 900 MHz to a PC capable of running radio transceiver applications. This board set is not intended to compete with high-performance PCI bus-based analog-to-digital and digital-to-analog converter (ADC and DAC) platforms on the market today but to serve as a low-cost alternative. While the *GNU Radio* developers are also in the midst of creating a “Universal Software Radio Peripheral (USRPN)” to provide competition in the high-end PCI ADC/DAC market at relatively low cost, that device will still not include any sort of radio “front end” for operation on any specific band of spectrum. The NCSA SDR board set allows developers with modest performance requirements to have

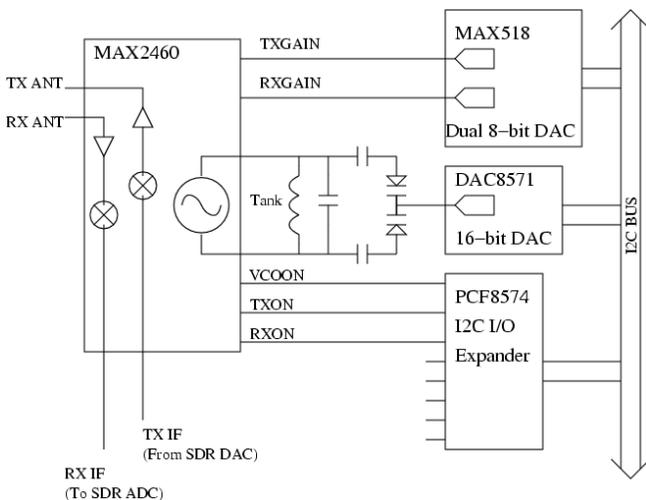
---

<sup>†</sup> This work was carried out under the National Center for Advanced Secure System Research (NCASSR) and funded by the Office of Naval Research (ONR) grant N00014-3-1-0765.

access to a low-cost (yet complete) hardware solution to explore DSP and SDR concepts at 900 MHz.

### 2.1. 900 MHz Conversion / Front End

The first board of the two board set developed by NCSA provides frequency conversion between the 900 MHz operating frequency and a relatively low intermediate frequency (IF) nominally centered at 10.7 MHz. This board operates on both transmit and receive paths, upconverting in the transmit path and downconverting in the receive path using conventional analog mixing (superheterodyne) techniques. As shown in Figure 1, the design based on the MAXIM MAX2460 transceiver integrated circuit, which provides most of the required mixing and local oscillator circuitry in a single 28-pin package. The low-cost MAX2460 converts between a tunable 4 MHz-wide block of spectrum at 900 MHz and an IF block ranging from 8.5 MHz to 12.5 MHz. [2] This allows the NCSA 900 MHz conversion board to handle fairly wide modulation schemes. Additionally, the MAX2460 on-board oscillator allows for a simple varactor diode-based tuning design. The NCSA 900 MHz conversion board incorporates such circuitry to allow operation anywhere within the United States 902-928 MHz industrial/scientific/medical (ISM) band.



**Figure 1 – Block diagram of the NCSA 900 MHz conversion board**

All of the control functionality of the 900 MHz conversion board is integrated into a single I2C two-wire serial bus. All analog tuning/control voltages and digital control bits are interfaced through this bus which can in turn be easily interfaced to a host device through a parallel printer port or specialized USB device. This keeps interfacing requirements fairly simple and straightforward to

implement. An I2C-enabled dual 8-bit digital-to-analog converter (DAC) is used to drive gain control inputs on the MAX2460 and an I2C-enabled 16-bit DAC is used to produce the tuning voltage required by the varactor-based oscillator. The 16-bit DAC yields a tuning increment of about 400 Hz per step. A PCF8574 I2C I/O expander chip provides digital control functionality.

### 2.2. Intermediate Frequency Sound Card Interface

The second board developed by NCSA is a second stage of frequency conversion and filtering to interface the IF of the 900 MHz conversion board to a PC sound card. To keep the cost of NCSA's experimental SDR low, the PC sound card was leveraged as a low cost ADC/DAC device. There is a performance tradeoff since most sound cards limit the RF signal width to 48 kHz, but such operation is still suitable for many applications including voice and low-speed data. (For comparison, commercial land mobile radio systems operate with 12.5 kHz- and 25 kHz-wide channels.) Ideally, the ability to process signals several MHz wide is highly desirable for many SDR applications, but for the goals of this project (i.e. exploration of SDR techniques on a modest DSP platform) the narrowband limitation is not unreasonable.

Rather than building a conventional mixer-based second IF stage to convert the 10.7 MHz IF signal down to the 0-48 kHz baseband frequency for the PC sound card, a different technique recently implemented by another SDR project was used. In his SDR-1000 project, Youngblood used a technique he refers to as "Quadrature Switching." [3] [4] For the receiving direction, the Quadrature Switched Detector (QSD) consists of a four-throw commutating switch that applies the incoming signal to one of four sampling capacitors. (See Figure 2) The switch applies the incoming signal to each of the capacitors for one-quarter period of the carrier of the incoming signal. i.e., If the switch were a rotary switch, it would complete a full revolution at the same frequency of the IF, a nominal 10.7 MHz in this case. The capacitors integrate the incoming signal for one-quarter of the RF cycle and represent how the carrier signal is changing over time. By differentially summing voltages present at the capacitors representing 0 and 180 degrees as well as those representing 90 and 270 degrees, baseband representations of the in-phase ("I") and quadrature ("Q") components of the signal present at the IF frequency can be obtained. The benefit to using quadrature switching instead of conventional mixing techniques is that it offers high-Q filtering with minimal loss. Quadrature switching also reduces the number of parts required for implementation. Furthermore, those parts need not be of high precision to obtain a high level of balance required between the I and Q outputs - something that can be difficult to achieve with inexpensive conventional mixers.

As shown in Figure 3, the NCSA IF board design, as in early Youngblood designs, the commutating switch of the QSD is implemented as a single 1:4 demultiplexer driven by a clock/counter circuit operating at four times the desired IF frequency.[3] Note that one clock step in this case represents one-quarter cycle of the commutating switch, hence the clock needs to run four times the carrier frequency. High quality instrumentation amplifiers are used as differential amplifiers to produce the I and Q signals fed to the input of the sound card.

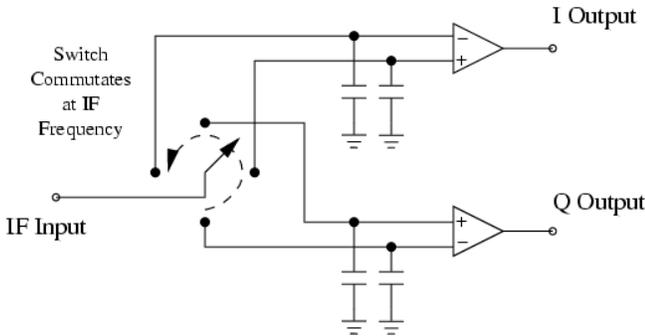


Figure 2 – Quadrature Switched Detector

The Quadrature Switched Exciter (QSE), used to produce transmit signals, is essentially the QSD wired in reverse. I and Q signals from the PC sound card are each connected to a differential line driver, yielding signals to represent 0, 90, 180, and 270 degree “components” of the signal to be transmitted. These signals are applied to capacitors connected to a switching arrangement identical to that used by the QSD. An image of the signal represented by the I and Q signals from the sound card then appears at a carrier frequency equal to the switch commutation frequency.

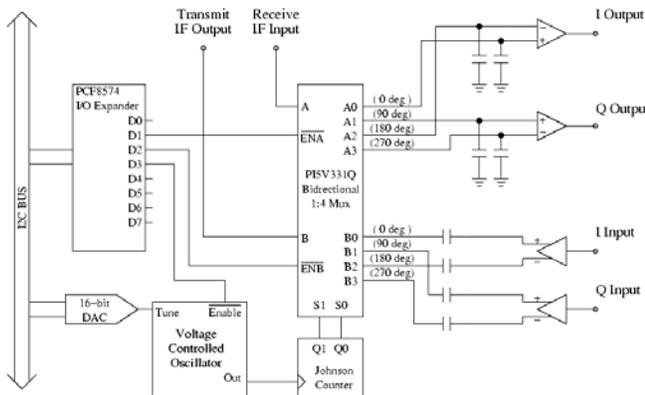


Figure 3 – NCSA IF (Soundcard Interface) Block Diagram

In order to provide additional operational flexibility in the NCSA IF board, the clock oscillator was implemented

with a voltage-controlled oscillator (VCO). This allows the IF to vary from its nominal operating frequency to operate with different types of RF front ends. Additionally, a variable IF allows the SDR to fine-tune to a signal without the need for retuning the front end. The initial design called for the use of a TTL VCO (74LS624) device, but that device could not reliably operate at the 42.8 MHz rate required to attain the 10.7 MHz IF frequency. The final design settled on a Mini-Circuits POS-75P plug-in VCO followed by a simple comparator circuit to provide the TTL levels required by the flip-flop-based clocking circuit. This arrangement allows the NCSA IF board to operate with an IF ranging between roughly 8 and 12 MHz. Again, an I2C-enabled 16-bit digital-to-analog converter provides the VCO tuning voltage; in this case the resulting tuning step was measured to be about 60 Hz.

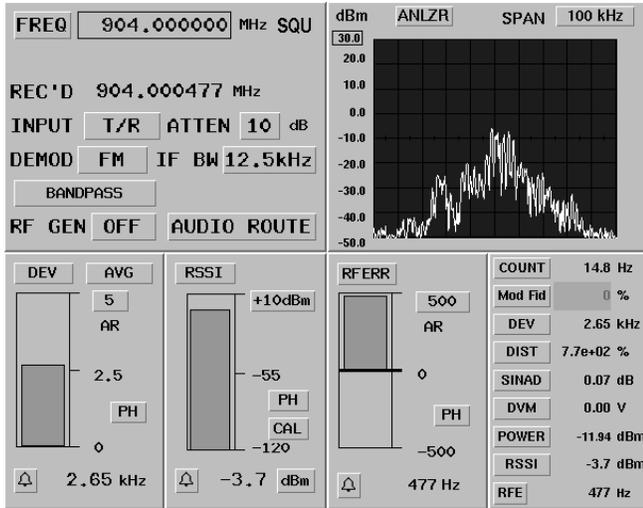
Like the 900 MHz conversion board, control functions on the NCSA IF board are handled by an I2C serial bus but the IF board also hosts a DLP Design DLP-USB245M USB-to-parallel interface. This interface serves as the I2C bus master and allows a PC to directly control the board. A pass-through connection is provided to extend the I2C bus to other boards, such as the NCSA 900 MHz conversion board. NCSA wrote a small suite of control code that controls all of the functions of the two board set through this USB/I2C interfacing arrangement.

For a test application, simple FM transmitter and receiver programs were written to send and receive audio with the NCSA board set. This test code was based on code provided with the GNU Radio package. The test waveform was an FM signal based on 25 kHz channel-spacing with a typical deviation of 3 kHz (a typical land mobile radio waveform). Signals were successfully transmitted and received over short (<30 meters) distances. Receiver sensitivity was measured to be approximately 25 microvolts (12 dB SINAD) and output power of the transmitter ranged between -9 and 0 dBm. The results of transmitter testing are shown in Figure 4.

While the performance of the NCSA SDR board set was satisfactory, there are some areas that need improvement. The biggest issue is stability of the 900-MHz local oscillator. The MAX2460 has no on-board Phase-Locked Loop (PLL) to keep the oscillator frequency stable. This causes the radio operating frequency to shift a few kilohertz over the course of ten minutes. While the circuit recommended by the Maxim data sheet may be suitable for conventional wideband consumer applications (cordless phones, etc.), this drift makes the design unsuitable for our SDR application and will need to be augmented with an external PLL. While adding a PLL will increase the part count, it should not significantly impact complexity or cost as low cost single-chip PLL solutions are readily available.

Even though these two boards were built to be used together as a set, they can also be used individually as

“building blocks” in conjunction with other signal processing systems. The IF board, for example, can be combined with a CATV tuner module with minimal interfacing, producing a wide-coverage narrowband receiver.



**Figure 4 – Test Set Display Showing Test Waveform Sent by NCSA SDR Transmitter**

The 900 MHz conversion board is also used in conjunction with NCSA’s Extensible Sensor Platform (ESP) to provide radio communication capabilities to simple remote sensing nodes running on FPGA-based embedded platforms. It is anticipated that the NCSA 900 MHz design will easily interface to *GNU Radio* group’s USRP when that ADC/DAC platform becomes available.

### 3. SDR VISUALIZATION

The initial SDR visualization (SDR-VIS) uses a pair of software defined FM radios to transmit and receive a gray-scale image. The receiver is instrumented to extract the signal after each stage of radio reception; i.e. after filtering, subtracting the carrier, and demodulation. The extracted signals are then sent via TCP to a visualization front end, where the data is combined with a graphical metaphor of the SDR receiver to provide a visual representation of the working radio. The intent of this visualization is to demonstrate both the principles of FM transmission as well as give a broad overview of how the radio processes the images for transmission. The intent of the SDR-VIZ presentation of these concepts is to give viewers additional insights as to how a radio operates and allow them to “see” inside of a radio in ways that are not possible with hardware-based radio implementations.

*GNU Radio* is a modular system, where a software radio is defined as a set of functional blocks (such as filters,

demodulators, and data sinks), and a set links connecting an output of one filter to an input of another. This yields a straightforward visual representation of an SDR as a 'pipeline', with 'stations' along the way representing the functional blocks.

SDR-VIS works by taking a gray-scale image, encoding each horizontal raster line into an audio signal segment, and strings together the output from successive lines to form an audio signal. The rest of the transmitter frequency modulates the signal, mixes it with a complex carrier, and then sends the signal to the sound card, where the narrowband and 900 MHz conversion hardware converts it to RF.

A fairly unique encoding scheme was used with SDR-VIZ to maximize the benefit of visualizing the various stages of a SDR transmitter/receiver pair. Images are encoded by considering each line of the image as the spectrum of a waveform; i.e. the horizontal position of a pixel corresponds to a certain frequency, and the intensity of that pixel corresponds to the amplitude of a sinusoid at that frequency. As the encoded image is processed by the various stages of the SDR, the visualization of the each stage’s spectra vs. time yields images related to the original image. Comparing the images to each other as well as the original image gives the viewer some intuitive notion of how the respective stages are manipulating the signal as it passes through the radio system.

## 4. SDR COMMUNICATION PROTOCOL

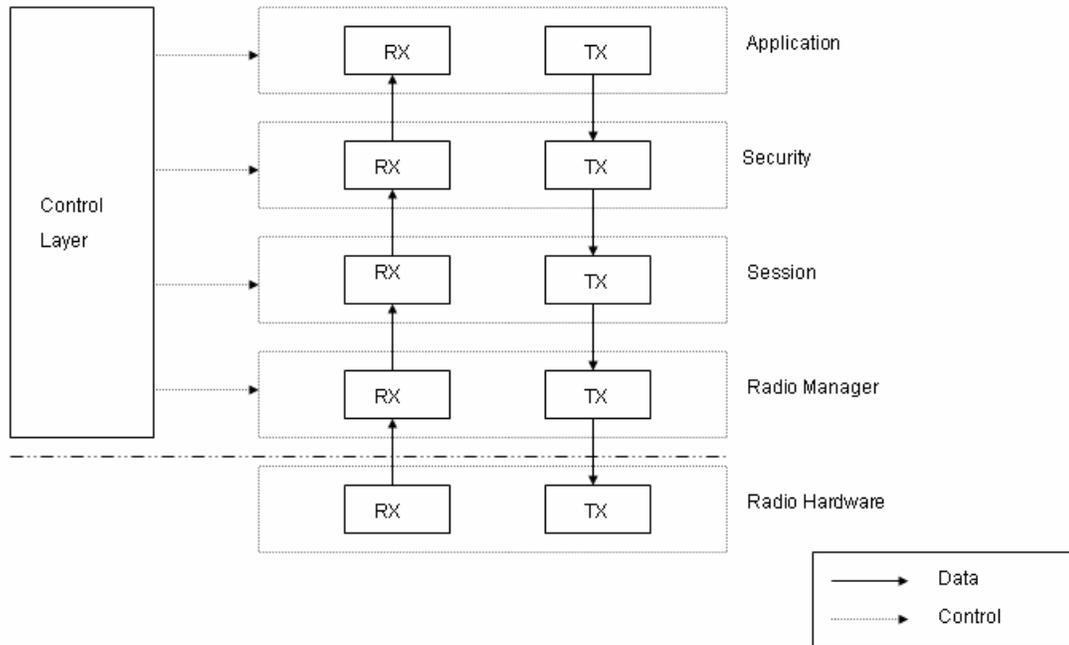
The *GNU Radio* platform was also used to prototype a data communication protocol that will be utilized on other devices, such as NCSA’s Extensible Sensor Platform. The SDR Communication (COM) Protocol stack, developed in the course of this project, has a modular, layered structure. It is made up of two stacks: the Data Stack and the Control Stack. The data stack allows for bidirectional data flow between the installed modules in the data stack. The control stack is currently a single layer stack which controls the data stack operation.

The stacks are implemented so that at run-time, any module in a stack may be replaced, new modules may be inserted, or a module may be removed. This architecture is very similar to the STREAMS architecture developed by AT&T Bell Laboratories for the UNIX System V distribution [5].

### 4.1. SDR Communication Protocol Data Stack

As depicted in Figure 5, our prototype SDR COM data stack consists of five layers:

- The Radio Hardware layer contains the physical components, the RF front-end hardware.



**Figure 5 – SDR COM Stack Control and Data Paths**

- The Radio Manager layer is the GNU Software Radio distribution with NCSA's transmit and receive modifications.
- The Session layer defines the format of the data sent over connections. For example, data packetized using UDP or TCP [6].
- The Security layer is responsible for providing confidentiality and integrity protection of the transmitted data. Different implementations of this layer would implement various algorithms. For example, various combinations of algorithms may be AES and MD5 or Triple DES and MD5 or DES and MD5 [7].
- The Application layer could be any data source or consumer, e.g., a microphone or a speaker.

Each layer in the data stack is implemented as two C++ objects, one for the transmit path and the other for the receive path. These objects are interconnected in a manner similar to UNIX pipes [5]. The pipe objects are implemented as shared memory objects between the reader and the writer. The control stack communicates through these pipe objects with the Application, Session, Security and Radio Manager layers via UNIX signals and messages [5].

On start-up, the stack is adjusted to a "gold channel". This gold channel is set of pre-defined parameters: frequency, modulation type, etc. Use of the gold channel places the SDR into a known communication state which allows for communication with external entities which may then remotely configure the SDR. Such dynamic reconfiguration will allow radios to change modulation scheme, encryption mode, or other operational parameters to accommodate the current radio environment.

## 4.2 SDR Communication Protocol Stack Implementation

This section will cover mainly the software implementation of the data stack. The control stack is currently a single module that communicates with the layers of the data stack through UNIX messages and signals [5]. The architecture of the control portions of the SDR COM Protocol Stack is currently an area of research.

## 5. SECURITY

Given that the SDR is easily reconfigurable and extensible, security to prevent unauthorized access to its capabilities is a highly desirable feature. There are currently two security projects in progress in the NCSA SDR project: voice authorization to allow or disallow SDR capabilities, and

secure download to prevent unauthorized configuration changes to the SDR.

### 5.1 Voice Authorization

The NCSA SDR project has implemented a Voice Authorization mechanism to identify and authenticate end users. Once a user is identified, they are granted a pre-defined set of SDR capabilities. Professor Thomas S. Huang and Ming Liu of the University of Illinois at Urbana-Champaign provided a set of programs based on Gaussian Mixture Models (GMM) [8] for use in this project.

The speech verification code base consists of four applications: gmCopy, gmTrain, gmAdapt, are used to train the Universal Background Model (UBM)[8] and individual speaker models, while gmEval is used to test and verify a speaker.

The speech verification using GMM is a two step process. The first step requires training the UBM and individual speaker models. Voice samples for users are collected and converted into .mfc files using gmCopy. These .mfc samples are then used to train the UBM. The second step is to train individual user models using UBM and gmAdapt. To identify and authenticate any speaker, the unknown voice sample is tested against all speaker models in the database using gmEval. This results in a numerical score. The model resulting in the maximum value among all the scores, gives the identity of the speaker. Thus in this process the speaker is not only identified, but also authenticated against his own individual speaker model.

As shown in Figure 6, a GUI application, Control, will enable any enrolled/registered user to interact with and access the features of the SDR Communications stack. The Policy Database contains permissions for each enrolled/registered user. The permissions settings determine the SDR data stack features available to a particular user. For example, user U may be allowed to listen to the FM broadcast band, but not the military frequency band. The secure repository contains the actual applications that an authenticated user may execute, based on his permissions, and thus install the SDR data stack. For example, if user U wants to tune to a commercial FM radio station at 107.1 MHz, then the relevant application is loaded from the secure repository and executed to give the desired result.

### 6. CONCLUSIONS AND FUTURE WORK

The NCSA SDR project provides a set of simple hardware and software interfaces for experimenters and developers to experiment with SDR techniques on commodity PC hardware. Development will continue on the 900 MHz conversion board to increase stability with the addition of a PLL. Furthermore, work will continue on

integrating the 900 MHz conversion board with embedded platforms with moderate processing capability. The SDR COM protocol stack will also be ported to this environment to provide the same functionality now available in the GNU Radio PC environment.

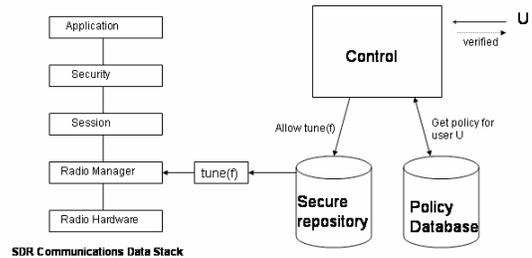


Figure 6 - SDR Voice Authorization Implementation

### 7. REFERENCES

- [1] Free Software Foundation, *GNU Radio home page*, <http://www.gnu.org/software/gnuradio/index.html>
- [2] Maxim Integrated Products, *MAX2420/MAX2421/MAX2422/MAX2460/MAX2463 900 MHz Image-Reject Transceivers*, Jan 2003, Rev 4. (Manufacturer's Data Sheet)
- [3] Youngblood, G., "A Software Defined Radio for the Masses, Part 1," *QEX*, pp. 13-21, Issue 213, Jul/Aug, 2002.
- [4] Youngblood, G., "A Software Defined Radio for the Masses, Part 4," *QEX*, Issue 217, pp. 20-31, Mar/Apr 2003.
- [5] Bach, M.J., *The Design of the UNIX Operating System*, Prentice Hall, New Jersey, 1986.
- [6] Stevens, W. R., *Advanced Programming in the UNIX Environment*, Addison-Wesley, Massachusetts, 1993.
- [7] Hall, E. A., *Internet Core Protocols: The Definitive Guide*, O'Reilly and Associates, Cambridge, MA, February, 2000.
- [8] Reynolds, Douglas A.; Quatieri, Thomas F.; Dunn, Robert B., "Speaker Verification Using

Adapted Gaussian Mixture Models, Digital Signal  
Processing, Volume 10, Issue 1-3,  
January, 2000.