

# Demo Abstract: Software Radio Implementation of Short-range Wireless Standards for Sensor Networking

Thomas Schmid

Tad Dreier

Mani B. Srivastava

Networked & Embedded Systems Laboratory  
Electrical Engineering Department  
University of California, Los Angeles  
{thomas.schmid, tjdreier, mbs}@ucla.edu

## Categories and Subject Descriptors:

C.2.1 [Computer Systems Organization]: Wireless Communication

**General Terms:** Design, Experimentation

**Keywords:** Software Defined Radio, IEEE 802.15.4

## 1 Introduction

Software Defined Radios can provide significant benefits as backend gateways or base stations for sensor networks, which do not face the stringent resource constraints of in-network nodes. We extended GNU Radio with two physical layer implementations of IEEE 802.15.4 and an FSK modulation, and use the Universal Software Radio Peripheral (USRP) to interoperate with the Chipcon CC1000 and CC2420 radios found on the popular Mica2, MicaZ and Telos B motes. The wideband nature of the USRP makes it feasible for a single SDR base station to simultaneously communicate on multiple independent channels, and provide network bridging across incompatible radio standards. The demo will show a software defined radio base station talking simultaneously to Mica2 and MicaZ nodes and will relay messages coming from the Mica2 to the MicaZ, and vice versa. Additionally, we will present a unified design and modeling environment for Software Defined Radio systems, which aims to bridge the diverse set of design abstractions and software tools used by engineers specializing in communication, networking, and embedded systems. Our environment extends Berkeley's Ptolemy II, a heterogeneous modeling and simulation environment, which provides an overall framework and graphical interface. We provide a Ptolemy interface to GNU Radio for physical layer design, and the Click modular router from MIT, for network level design. This approach allows designers to work at their preferred de-

sign abstraction, while permitting integrated system modeling, access to rich libraries of existing system components, and efficient code generation for implementation on real targets.

## 2 Background

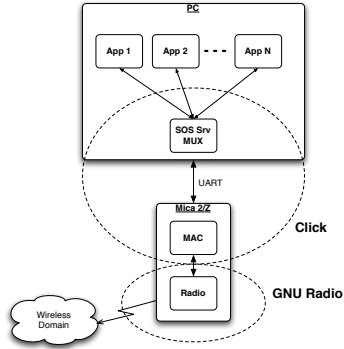
A software defined radio is a system which uses software for modulation and demodulation of communication signals. The goal is to use as few hardware as possible. The ideal software defined radio would have an antenna which gets sampled by an ADC and the rest of the processing is done in software. Unfortunately, fast ADCs are very difficult to build and very expensive. Therefore, we need some more hardware to first down-convert a band of adequate width from a higher frequency into an intermediate frequency where we then can sample it with our ADC. This is possible today with fairly inexpensive hardware and one can easily process bands of at least 32 MHz.

### 2.1 GNU Radio

GNU Radio [1] is a collection of open source software which when combined with minimal hardware allows to construct radios, and thus turns usual hardware into software problems. The main goal of GNU Radio is to allow easy combination of signal and data processing blocks into powerful modulation, demodulation, or more complex signal processing systems. GNU Radio achieves this by providing simple signal processing primitives written in C++. By using SWIG, an interface compiler which allows an easy integration of C/C++ into scripting languages, GNU Radio provides a simple interface to the signal processing blocks from Python. Thus, the power of scripting languages is used to simply connect the signal processing blocks which can run at native speed without any interpretation.

### 2.2 The Universal Software Radio Peripheral (USRP)

Matt Ettus developed the USRP [2] as a low-cost and flexible platform for software defined radios. It consists of one motherboard which holds the ADCs, DACs, and a FPGA to do some simple, but bandwidth consuming processing, and to reduce the data rate such that one can push it over the USB 2.0 connection, with which it is hooked up to a PC. The four high-speed ADCs are 12-bit Analog Devices AD9862 and have 64 MS/s. This allows an effective bandwidth of about



**Figure 1.** This diagram shows which parts in a conventional setup with Mica2/Z basestation and computer are replaced by GNU Radio and Click.

32 MHz. The DACs have 14 bit, 128 MS/s and allow to generate signals of up to around 50 MHz. From these figures we can see that the bottle neck is the USB 2.0 connection to the computer which has at most a data rate of 32 MByte/s, thus resulting in a maximum of 8 MS/s of complex signals (16-bit I and 16-bit Q channel).

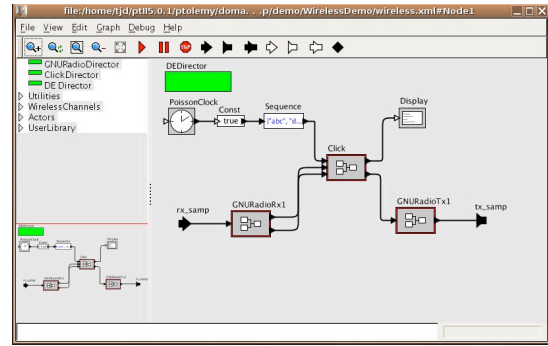
### 2.3 Ptolemy

Ptolemy II [3] is written entirely in Java. It provides several basic framework classes that are common among different models of computation (MoC's), and provide a consistent interface that allows components with varying internal computational semantics to seamlessly interact with one another. In Ptolemy, MoC's are also referred to as domains. In many cases, a domain will provide subclasses for the framework classes, which implement the appropriate domain-specific behavior. Some examples of existing domains include discrete-event (DE), process networks (PN), and finite state machine (FSM).

### 3 Demonstration

The demo consist of two computers, each with one USRP capable of transmitting in the 430MHz and 2.4GHz ISM band. We use standard Mica2 [4] and MicaZ [5] motes both running the SOS operating system [6] for embedded sensor devices. The nodes transmit by default on the 430MHz and the 2.4GHz ISM band and can not communicate with each other directly. Therefore, one computer runs a bridging application written in GNU Radio which translates messages from the 2.4GHz ISM band using IEEE 802.15.4 for the physical layer to the 400MHz ISM band with a FSK modulation, such that they can be received by the CC1000 radio chip used in the Mica2 motes. This will allow the two networks to talk to each other. The other computer monitors both wireless bands and shows the different messages which are sent around on the different channels. A simple ping-pong application will send around a packet from one network to the other and back again. LEDs on the nodes and logging facilities on the second computer, as well as logging output on the bridge, indicate the success of passing messages from one network to the other.

A third computer runs the rapid development environment for software defined radio based on Ptolemy II, GNU Radio,



**Figure 2.** Screenshot of the rapid development environment showing modules for Click and GNU Radio.

and the Click [7] modular router. We will show how one can easily recreate similar code we are using on the bridge and adapt its routing behavior or even exchange the whole physical layer. Depicted on Figure 2 is an example of how the interface looks.

### 4 Conclusion

The current setup is a first step towards investigating software defined radio techniques for wireless sensor networks and low power communication protocols like IEEE 802.15.4. It allows us to identify problems and differences with which one has to deal in order to use software defined radio solutions as they exist today. The rapid development environment facilitates this research by providing a fast way of altering a system and generating a visual representation of the code. Additionally, it allows fast prototyping of routing protocols and it opens the possibility of integrating cognitive radio aspects into wireless sensor networks.

**Acknowledgments:** This is work being done under the DARPA ACERT program.

### 5 References

- [1] Eric Blossom et al. GNU radio. <http://www.gnu.org/software/gnuradio/>.
- [2] Matt Ettus. Universal software radio peripheral. <http://www.ettus.com>.
- [3] J. Davis et al. *Ptolemy II: Heterogeneous Concurrent Modeling and Design in Java*. Electronics Research Laboratory, College of Engineering, University of California, 2001.
- [4] Crossbow Inc. Mpr410cb, mica2. <http://www.xbow.com/products/productsdetails.aspx?sid=72>.
- [5] Crossbow Inc. Mpr240, micaz. <http://www.xbow.com/products/productsdetails.aspx?sid=101>.
- [6] C.C. Han, R.K. Rengaswamy, R. Shea, E. Kohler, and M. Srivastava. Sos: A dynamic operating system for sensor networks. *The Third International Conference on Mobile Systems, Applications, And Services.(2005)*.
- [7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M.F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297, 2000.