Virtual Lab for Wireless Sensor Networks

Dorel PICOVICI[1], Anghel Vasile CONTIU[2], Adina TOPA[2], John NELSON[3]
[1] *Department of Electronic Mechanical and Aerospace Engineering, Institute of Technology, Carlow, Ireland, Dorel.Picovici@itcarlow.ie*
[2] *Department of Computer Science, Technical University of Cluj-Napoca, Cluj-Napoca, Romania, Anghel.Contiu@ro-metaltrade.com*
[3] *Department of Electronic & Computer Engineering, University of Limerick, Limerick, Ireland, John.Nelson@ul.ie*

***Abstract*—This article details an experimental system developed to enhance the education and research in the area of wireless networks technologies. The system referred, as Virtual Lab (VL) is primarily targeting first time users or users with limited experience in programming and using wireless sensor networks. The VL enables a set of predefined sensor networks to be remotely accessible and controlled for constructive and time-efficient experimentation. In order to facilitate the user's wireless sensor applications, the VL is using three main components: a) a Virtual Lab Motes (VLM), representing the wireless sensor, b) a Virtual Lab Client (VLC), representing the user's tool to interact with the VLM and c) a Virtual Lab Server (VLS) representing the software link between the VLM and VLC. The concept has been proven using the moteiv produced Tmote Sky modules. Initial experimental use clearly demonstrates that the VL approach reduces dramatically the learning curve involved in programming and using the associated wireless sensor nodes. In addition the VL allows the user's focus to be directed towards the experiment and not towards the software programming challenges.**

***Index Terms*—moteiv, Tmote Sky, virtual clients, virtual lab wireless networks, wireless sensors**

I. INTRODUCTION

Recent performances in micro-electro-mechanical systems technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power, multifunctional wireless sensor devices small in size and able to communicate over short distances. These devices, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on collaborative effort of a large number of nodes. Wireless sensor networks can be defined as a collection of such sensor nodes. Research in this area has grown in the past few years given the wide range of applications that can benefit from such technology. Nodes in ad-hoc networks have often limited resources, but sensor nodes are even more constrained [1, 2]. Of all of the resource constraints, limited energy is the most critical one [3]. After deployment, many sensor networks are unattended for long periods and battery recharging or replacement may be infeasible or impossible. Nodes in sensor networks often exhibit trust relationships beyond those that are typically found in ad-hoc networks. Neighboring nodes in sensor networks often witness the same or correlated environmental events. If each node sends a packet to a base station in response, precious energy and bandwidth are consumed. To prune redundant messages, reduce traffic and energy, sensor networks require in-network processing, aggregation, and duplicate elimination.

The wireless sensor networks have reached the level of maturity with an acceptable level of performance for real world environments. However, there are several unsolved challenges such as protocols, real-time data, power management, programming abstractions, security, and privacy [4]. Usually, sensor networks are formed with a large number of sensor nodes. The cost of a single node is very important to justify the overall adoption of a wireless sensor network. If the cost of the network is more expensive than deploying traditional non-wireless sensors, then the sensor network is not cost-justified. It is important to mention that a sensor node may have additional components such as location finding system, mobilizer, or power generator depending on the applications of the sensor networks. As a result, the target sub-dollar cost for sensor node is proving a challenging issue. One of the challenges in developing applications for sensor nodes is the setup and construction of a simple network for functional evaluation and development. Frequently, initial efforts are frustrated in getting basic deployments running, due to long learning curves, local radio interference, etc. This research work developed a controlled web accessible Virtual Lab concept for exploring wireless sensor networks. The wireless sensor motes used for the Virtual Lab application are the Tmote Sky type motes developed by Moteiv Corporation [5].

II. THE VIRTUAL LAB (VL) APPROACH

The main motivation for Virtual Lab (VL) was to develop an application that could be used by all levels of users regardless their computer skills, to handle the challenge of new wireless sensors technology and increases the user's interest in such technology.

The main function is to allow the user to remotely access the TmoteSky motes and provide a transparent way of handling any problems that may occur. The VL has three modules:

1) Virtual Lab Server (VLS) application
2) Virtual Lab Client (VLC) application
3) Virtual Lab Motes (VLM)

Of the above-mentioned modules, the Virtual Lab Server (VLS) application is the main component in the communication chain. Some of its functions are:

a) To maintain the flow of all messages such as client's requests for the motes or the mote data.

b) To facilitate communication with the VLM and

Digital Object Identifier 10.4316/AECE.2008.02007

communication with the VLC.

c) To handle port availability and different settings required by the TinyOS mote operating system environment.

Figure 1 illustrates the components of the Virtual Lab application.

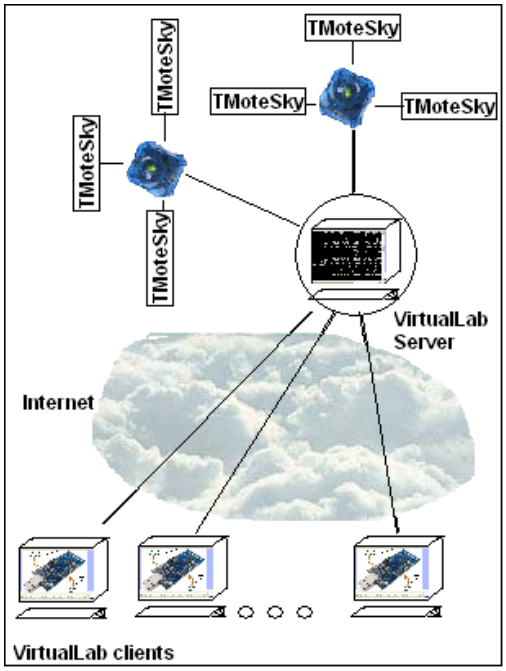

Figure 1. Virtual Lab's major components.

Each client runs the Virtual Lab Client application and is connected through sockets to the server computer running the Virtual Lab Server application. The server is connected to the Virtual Lab Motes using USB ports.

Virtual Lab Client (VLC) Overview

The Virtual Lab Client (VLC) is the end-user's tool for interacting with the remote motes. To facilitate data display from the motes, the VLC is connected between the Virtual Lab Server application, and the user. The VLC application informs the user about VLM availability inside the remote "laboratory" and the state of the installed applications (whether applications are currently running or not, their image number, name, compile date). The VLC's flexibility is provided by features that allow the user to install custom application on the VLM and also offers the same debugging facilities as the case when the user can connect directly to the VLM.

Virtual Lab Server (VLS) Overview

The VLS application is positioned in between VLC and VLM. Its main function is to send information from the VLC to the VLM and vice-versa. The VLS functionality has been also tested with multiple VLCs. The VLS commands for the VLM can be classified as the application level commands (the commands that influence the currently running application) and deluge level commands (the commands that target the list of currently installed application). An important achieved objective is represented by the VLS hiding the complexity of operations from the user by using a simple programming approach. The existing applications that involve communications between a java application and a mote require the user to set some of the environment variables and also to note the ports availability. The VLS automatically solves all these settings without the user's assistance.

Virtual Lab Motes (VLM) Overview

The proposed Virtual Lab (VL) application uses the Tmote Sky motes (as shown in Figure 2 and 3) developed by Moteiv Corporation [5].

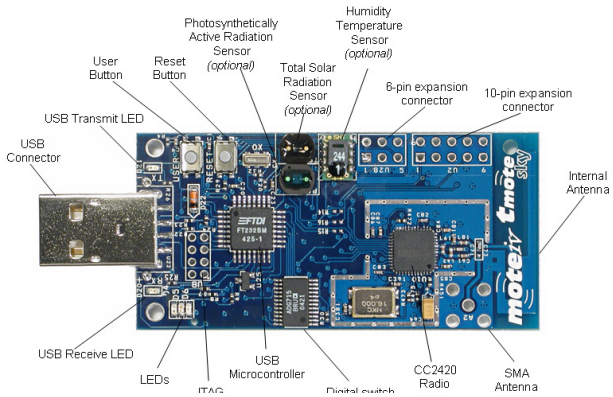

Figure 2. The front-side of T-mote Sky module.

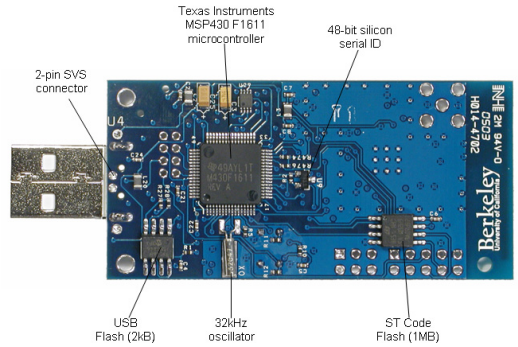

Figure 3. The back-side of T-mote Sky module.

Some of the main characteristics are:

- Programming and data collection via USB;
- 16-pin expansion support and optional SMA antenna connector;
- TinyOS support: mesh networking and communication implementation;
- Complies with FCC Part 15 and Industry Canada regulations;

The Tmote Sky [6] requires a voltage within 2.1 to 3.6 V DC range. A minimum voltage of 2.7V is necessary when programming the microcontroller flash or external flash. For situations when Tmote Sky module is plugged into the USB port for programming or communication, the operational voltage (3V) will be drawn from the host computer. The low power operation of the Tmote Sky module is due to the ultra low power Texas Instruments 16-bit RISC MSP430 F1611 microcontroller featuring 10kB of RAM, 48kB of flash, and 128B of information storage. The MSP430 has an internal digitally controlled oscillator (DCO) that may operate up to 8MHz. The DCO may be turned on from sleep mode in 6μs, however 292ns is typical at room temperature. When the DCO is turned off, the MSP430 operates using an eternal 32768Hz watch crystal. To program the microcontroller a

set of 51 instructions are available [7]. A ST M25P80 40MHz serial code flash is used for external data and code storage. The flash has a storing capacity of 1024kB of data and is decomposed into 16 segments, each 64kB in size. The flash shares SPI communication lines with the CC2420 transceiver.

To communicate with other devices, Tmote Sky uses an IEEE 802.15.4 compliant radio (Chipcon CC2420) with PHY and MAC functions [8]. The CC2420 is controlled by TI MSP430 microcontroller through the SPI port. The radio can be turned off by the microcontroller for low power duty cycled operation. The CC2420 has programmable output power and provides a digital received signal strength indicator (RSSI). Additionally, on each packet reception, the CC2420 samples the first eight chips, calculates the error rate, and produces a link quality indication (LQI) value with each received packet. Tmote Sky's internal antenna is an Inverted-F microstrip design protruding from the end of the board away from the battery pack [9]. The Inverted-F antenna is a wire monopole where the top section is folded down to be parallel with the ground plane. Although it has not a perfect omni directional pattern, the antenna has an operating range of up to 50-meter range indoors and 125-meter outdoors.

III. VIRTUAL LAB CLIENT (VLC) APPLICATION

The main function of the VLC application is to allow a user to remotely use the sensor motes, develop and test applications. In order to fulfill these requirements, the VLC has a graphical user interface (GUI) that assists the user to program and use the wireless sensors. As shown in Figure 4 the VLC's GUI has a main frame and several panels.

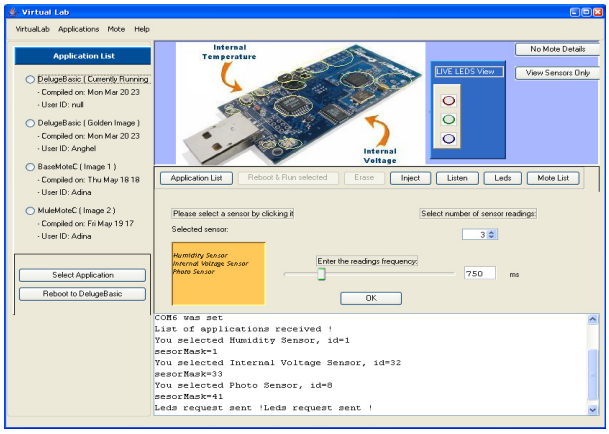

Figure 4. VLC application GUI.

The main frame contains all panels including the menu bar. The main panels (panels that are permanently displayed) are:

a) the left panel
b) the center middle panel
c) the image panel
d) the message panel

The left panel

As suggested by its title, the left panel is the panel from the left side of the graphical user interface. This panel display available motes and allow the selection of the desired one. Once the mote is selected the user can get further details about the state of the running applications. As shown in Figure 5, the bottom part of the panel shows details about the status of the current task (e.g.: executing), or about the list status (refreshing list).

Figure 5. VLC left panel executing status.

The center middle panel

This panel is situated right in the middle of the graphical user interface and it consists of two sub-panels.

The command toolbar

As shown in Figure 6, only the relevant buttons are available at any time.

Figure 6. VLC center middle panel-command toolbar.

For example, if the user has just connected to the server, the command toolbar will have the "Mote List" button available for the user to get the list of motes. After selecting a mote, more buttons will be enabled and more actions will be available. That is an important component of the GUI because it provides the user an easy way to take actions, regardless the actions he has previously taken. The command toolbar will allow the user to take only the actions that are available at a certain point.

The command panel

The panel shown in Figure 7 allows the user to send commands or apply new settings to the currently running application on the mote.

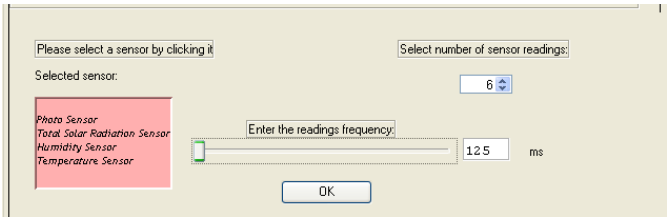

Figure 7. VLC center middle panel-command panel.

The user can activate different sensors, number of readings and the reading frequency.

The image panel

The image panel facilitates the interaction between the user and the mote. As shown in Figure 8, the panel is able to display two pictures of the Tmote Sky device alternatively, one of them containing detailed information (left side of the Figure), and another one with no details (right side of the Figure) for the users that are familiar with the device. When moving the mouse over the mote's detailed image, a tool tip text will show up, specifying the name of the element the mouse pointer is above. Moreover, the user can select the sensors from the mote's image. This is intended to provide an easy way for the user to interact with the application.

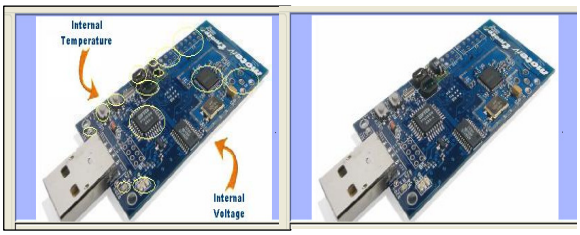

Figure 8. VLC image panel- Tmote-Sky modules.

The user can switch between the two images just by clicking the button on the upper right side of the panel. As shown in Figure 9 by activating this button the user can select: "No mote details" / "View mote details".

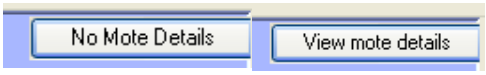

Figure 9. VLC image panel-details selection buttons.

Another important component that is situated on the image panel it, is the LED panel as illustrated in Figure 10.

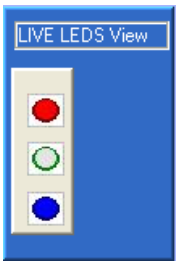

Figure 10. VLC image panel-live Leds view panel.

This component is actually able to provide live, in a visual way the state of the mote's LEDs. Although the LED panel displays real time "on" and "off" state of the LEDs with high accuracy, the on-off time can be influenced by the network delay. However, the display can be of tremendous help for the user in order to debug applications.

The message panel

As shown in Figure 11, this panel displays VLC confirmation messages, VLS confirmation messages, and actual data received from the VLM.

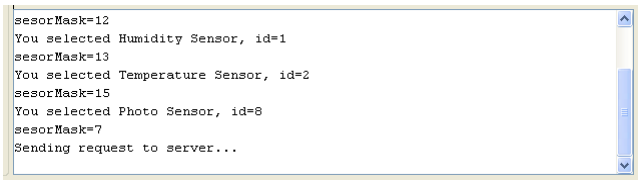

Figure 11. VLC message panel.

Similarly to the LED panel, the message panel is part of the feature that enables the user to debug applications that have the ability to send data to the server. Within this panel live mote data bytes can be displayed, even though the mote is remotely accessed.

Virtual Lab Client GUI functionality

Through VLC's GUI the user has the following options:

- retrieves the list of available motes (mote list button)
- choose a mote from the list of available motes (select button)
- retrieve the list of the installed applications on the selected mote (application list button)
- selects one of the installed applications (select button)
- reboot & run (reboot and run button)
- action confirmation; commands are launched based on user agreement.
- in case of rebooting, the left panel will display the currently running applications.

IV. VIRTUAL LAB (VL) COMMUNICATION

In this section, some technical details of the various communication interfaces implemented are given.

VLM - VLS communication

USB ports facilitate the communication between the VLS and the VLM. Each VLM will be assigned a unique COM port for the entire duration of connection. The LedsObserver and CommOscope components [10] run on the VLM and create connectivity with the VLS and other VLM. In order to achieve this, a nesC [11] application running on the mote must be wired to some specific components like CC2420 (for radio messages), UARTComm for the UART messages that are sent to the VLS or GenericComm for both radio and VLS.

nesC communication interfaces

The VLabOscope application use parameterized interfaces similar to GenericComm component. The GenericComm component can be written:

```
provides{
  interface SendMsg [uint8_t   id];
 }
```

This component provides 256 different instances of the SendMsg interface, one for each uint8_t value. The CommOscope component is responsible for communicating by radio or by UART the sensor readings. It is wired to the GenericComm component and it uses the parameterized interfaces ReceiveMsg[value1] and SendMsg[value2], meaning that it can receive messages that have the handler value equal to the value1 and it can send messages that require a value2 handler to be read as the server application describes.

Active Message Model

The most important element used to send and receive messages is the active message model, also called AM. It is defined in the AM.h file inside the TinyOS file structure [12,13]. Communication in TinyOS follows the AM model, in which each packet on the network specifies a handler ID that will be invoked on recipient nodes. The handler ID can be understood as an integer or "port number" that is carried in the header of the message. When a message is received, the receive event associated with that handler ID is signaled. Different motes can associate different receive events with the same handler ID. In any messaging layer, there are 5 aspects involved in successful communication:

1) Specifying the message data to send
2) Specifying which node is to receive the message
3) Determining when the memory associated with the outgoing message can be reused
4) Buffering the incoming message

5) Processing the message on reception

In TinyOS Active Messages, memory management is very constrained as one can expect from a small-scale embedded environment. The active message structure as shown in Figure 12 contains the fields for the destination address, message type (the AM handler ID), length, payload, etc.

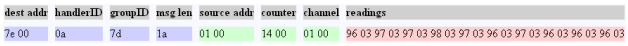

Figure 12. The active message structure.

The maximum payload size is TOSH_DATA_LENGTH and has a default value of 29. The programmer can change the TOSH_DATA_LENGTH but this operation is not recommended since long messages are subject to transmission errors and the entire message has to be retransmitted. The retransmission will be made with a cost reflected in the power consumption on both parts (sender and receiver).

VLS – VLM communication

The communication between VLS and the VLM is achieved by means of message structures that reside as classes on the server side and inside the .h files on the mote side. In order to develop the communication protocol, the VLS application must use the MoteIF objects together with the PhoenixSource objects, MIG automatically generated classes and Listener objects [14]. The MoteIF object will auto register with the MIG automatically generated class (representing the message structure) and to a listener that will call the messageReceived function when a new message arrives. The objects that Virtual Lab's MoteIF registers with are given in Table I:

TABLE I. VIRTUAL LAB'S MOTEIF REGISTERS

MoteIF object	MyMessageListener	Message Type Instance	MIG auto generated class
moteif2	vlabListenerRx	vlabOscopeRx	VLabOscope()
	vlabListenerTx	vlabOscopeTx	VLabOscope()
	ledsListenerRX	ledsRx	LedsObserverMsg()
	ledsListenerTimerRx	ledsTimerRx	LedsObserverMsg()
	ledsListenerTx	ledsTx	LedsObserverMsg()

The instances of MIG auto generated classes must have the handler set in advance such that mote components will be able to handle them. The VLS application sets the handler for these objects accordingly to the handlers expected by the mote's Oscilloscope TmoteSky application. The registered MoteIF's elements are given in Table II.

TABLE II. REGISTERED MOTEIF ELEMENTS

Message Listener object	Message Type Instance	Handler number	Description
vlabListenerRx	vlabOscopeRx	10	Listen to Oscilloscope's messages
vlabListenerTx	vlabOscopeTx	32	Send messages to Oscilloscope component
ledsListenerRX	ledsRx	11	Listen to LedsObserver messages
ledsListenerRx	ledsTimerRx	12	Listen messages involving LedsObserver's timer
ledsListenerTx	ledsTx	13	Send messages to LedsObserver

The server needs to translate client's messages into messages that the mote will be able to understand. This goal is achieved by using the handler (FreshMessageParser object) and its connections in order to deal with the client's request on one side, and the TestMif and MoteIF objects to actually send the data to the mote. When the FreshMessageParser determines the type of request the client is ready to send to the mote, it creates the TestMif object (and the MoteIF object implicitly) if it is not yet created and sends then a message to the mote using the object of the class that was automatically created by the MIG tool, so the mote will understand the message. For example, when the client changes the settings for the OscilloscopeTmoteSky application, it sends the new parameters to the server, the FreshMessageParser object detects the client's intentions, creates a new TestMif object if necessary, and uses the VLabOscope object (generated by MIG) to set the parameters for the mote message. After setting the parameters, the server invokes the "moteIf2.send(MoteIF.TOS _BCAST_ADDR, vlaboscope)" operation and the message is sent to the mote. The mote will answer by sending the data in the new format to the server, which is then forwarded to the client.

VLM to VLM communication

The Tmote Sky motes are able to communicate with each other by using the IEEE 802.15.4-compliant CC2420 Radio. The nesC applications that are downloaded on the motes must implement the necessary interfaces so VLMs can transfer data between each other. The Virtual Lab provides two ways for the user to develop and download nesC applications that use the Tmote Sky's radio. The first method requires the user to download its application (that uses the radio components) on two or more VLMs that are connected to a computer running the VLS application. The second method is by using the deluge message epidemic. This method involves the following scenario: one mote connected to a PC contains the new image to be sent to other motes that are currently not connected to the PC and are powered by their own batteries. The DelugeBasic [15] application running on the mote connected to the PC will advertise and send the new image by radio to the other available motes.

VLS – VLC communication

The server–client communication is achieved using software sockets. The server has the multithreading feature implemented so multiple clients can connect and get access to the motes that are directly connected to the VLS application. Objects are sent between the client and the server in order to ensure that the user's request will get to the server and the mote data will get to the client through the server. The objects used by server and client in order to communicate are:

- ImageObj
- InjectImageObj
- LedsStateObj
- MoteListObj
- OscilloscopData
- OscilloscopRequest

All the classes representing the messages that are exchanged between the client and the server implement the Serializable interface. This interface allows the objects to be serialized and sent through the socket to the destination.

V. CONCLUSIONS AND FUTURE WORK

The Virtual Lab was developed to allow education and research sector to use the wireless sensor networks technology remotely. It is also designed to increase the user's interested in programming, mathematics, physics and everything that the wireless sensor motes and Virtual Lab can provide together. To achieve this, Virtual Lab combines several elements such as: graphical user interface, TCP/IP connection, message exchanging between the client and the server through TCP/IP, multithreading server, message handling, serial connection between the server application and the motes, and COM ports for the serial connection. The server application provides port handling and message object handling in order to avoid communication problems, watch timers for avoiding process stagnation; MIG (message interface generator) for server – mote communication objects, and TinyOS java tool chain for server – mote communication.

Although the user has no direct contact with the wireless motes the Virtual Lab provides the necessary tools to display motes inside the virtual laboratory, to manage the applications installed on the mote, to install new applications, and it provides the necessary nesC components and graphical user interface elements for real time debugging.

Initial experimental use of the VL clearly demonstrates that the learning curve involved in programming and using the wireless sensor nodes can be seriously reduced. In addition to this, the VL approach allows the user to focus more on the experiment to be taken rather than on the software programming.

Although significant amounts of effort have been made during this research work to produce a highly robust Virtual Lab architecture and application, there are many directions for further improvements and enhancement. One such direction is to improve the extensibility of the application level messages and MIG. Further work is required in including MIG as part of Virtual Lab in order to automatically provide the new java classes based on the structures in the ".h" file that is used by the nesC application. This will allow the user to more easily define and send more customized messages to a new developed application. The user should then be able to create nesC message structures, use MIG to create the objects that will be sent as messages to the mote, based on the nesC message structure and develop the associated control panels that will be added to the Virtual Lab's graphical user interface. Finally, a major challenge is to identify and setup a set of isolatable wireless sensor networks supporting various network topologies, with high usability potential. Isolation, and indeed controlling interference, in an RF sense, is highly desirable to allow the user to experiment in a controlled environment.

REFERENCES

[1] Stankovic, J.A., Research Challenges for Wireless Sensor Networks, ACM Press, July 2004

[2] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, Erdal Cayirci, A Survey on Sensor Networks, Communications Magazine, IEEE, August 2002

[3] Curt Schurgers Mani B. Srivastava, Energy efficient routing in wireless sensor networks, International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, October 2004

[4] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures, In Proceedings of First IEEE International Workshop on Sensor Network Protocols and Applications, May 2003.

[5] Moteiv Corporation, San Francisco, CA http://www.moteiv.com

[6] Moteiv Corporation, TmoteSky Technical documentation, http://www.moteiv.com/products-tmotesky.php

[7] Texas Instruments, MSP430x1xx Family User's Guide, http://focus.ti.com/lit/ug/slau049f/ slau049f.pdf

[8] Moteiv Corporation, Tmote Sky datasheet, http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf

[9] Chipcon Corporation, Inverted F antenna datasheet, http://www.chipcon.com/

[10] Anghel Vasile Contiu, A virtual lab approach for wireless sensor motes and wireless sensor networks, MEng Thesis, Technical University of Cluj-Napoca, 2006

[11] David Gay, Matt Welsh, David Culler, The nesC Language: A Holistic Approach to Networked Embedded Systems, ACM Press, May 2003

[12] Philip Levis, TinyOS Programming, available online at http://csl.stanford.edu/~pal/pubs/ tinyos-programming.pdf , February 2006

[13] Philip Levis, David Gay, Vlado Handziski, Jan-Hinrich Hauer, Ben Greenstein, Martin Turon, Jonathan Hui, Kevin Klues, Cory Sharp, Robert Szewczyk, Joe Polastre, Philip Buonadonna, Lama Nachman, Gilman Tolle, David Culler, and Adam Wolisz, TinyOS 2.0, The 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys'05), November 2005

[14] MIG tool description, available at http://www.tinyos.net/tinyos-1.x/doc/nesc/mig.html

[15] Jonathan Hui, Deluge 2.0 –TinyOS Network Programming, available online at http://www.cs.berkeley.edu/~jwhui/research/deluge/deluge-manual.pdf