Development of Emulation Projects for Teaching Wireless Sensor Networks

Deepesh Jain **T. Andrew Yang**
University of Houston – Clear Lake
Houston, Texas

Abstract

In recent years research and development in wireless technologies have achieved an exponential growth. Wireless sensor networks (WSN's), in particular, have captured the attention and interest of various sectors, including research, industry and academia. Because of the increasing popularity and uses of WSN, many universities have included this subject in their curricula. Effective learning and teaching of WSN, however, require the students' gaining hands-on experiences in developing WSN projects, which help the students to get in-depth understanding of the subject. Merely studying the theories is not sufficient to understand the strength and limitation factors of this new technology. In this paper we report our experience of teaching WSN, including how a novice should start learning this subject by developing lab projects. The lab projects discussed in this paper will help the instructor to adopt the labs in his/her teaching; the labs will also help to smooth the learning curve when a student starts to learn the basic concepts of the subject. Our goal is to provide a set of WSN labs that will help to answer preliminary questions such as "How to set up a sensor network", "Hardware/software required", "How to program sensor nodes", "How the nodes communicate", et al.

1. Introduction

Wireless sensor networks are a new class of distributed systems. Most of the sensor networks have certain fundamental features in common. However, the most essential feature is that they are embedded in the real world. Sensors detect the world's physical nature, such as light intensity, temperature, sound, or proximity to objects [1]. We know that it is difficult for a student to understand a new field of technology with no prior experience. The same is true with WSN; for a beginner it is hard to deploy or test such a network. Our goal is to develop a set of WSN lab projects that could be easily adopted by the instructor, and could be used to help the students get started in testing and/or deploying wireless sensor networks. Although both simulation and emulation may be employed when developing the labs, we focus on emulation-based projects in this paper, by using actual devices (motes, sensor boards, gateways, et al).

Wireless sensor network is a new field and researchers are still exploring it. However most of the WSN projects are implemented using simulation, rather than emulation. The reason behind this phenomenon is that implementation of WSN projects with actual devices is more difficult and demanding. Actual hardware devices are needed which increases the project cost and complexity compared to the simulation approach, in which only simulation software is required. Emulation becomes more complicated when a large number of sensor nodes need to be deployed to form a sensor network. Another limitation is battery power constraint; it may be cumbersome to replace the battery of a node while the WSN application is already deployed. The sensors, for example, may be deployed in a hard-to-reach location.

Before delving into the details of lab projects, this paper discusses basic concepts of wireless sensor networks. First, the basic working of wireless sensor network is described, followed by the hardware and software requirements of WSN. The rest of the paper discusses the details of the lab projects, including problem definition, steps to implement the project, et al.

2. Basics of Wireless Sensor Networks

A wireless sensor network is a network made of a set of independent sensor nodes. The sensor nodes are self-contained units consisting of a battery, a radio module, sensors, and a low-speed on-board processor

[2]. Normally the unit that consists of the radio module and the on-board processor is called the *mote*. A complete sensor node is composed of the sensor board(s) and the battery mounted on the mote. Deploying a WSN requires iteratively programming a set of nodes, positioning them in an area large enough to produce an interesting radio topology, and using them to extract debugging and performance data [3]. For classroom projects, it is usually sufficient to deploy three or so sensor nodes in a room that is large enough to accommodate the WSN. A typical WSN consists of a set of nodes collecting data from the environment; these nodes transmit the collected raw data to a base station, which typically has enough computation power compared to the sensor nodes. The base station is generally a desktop computer connected to a gateway, which collects data sent by the sensor nodes. Once data is collected, an application running on the base station analyzes the received data packets, performs appropriate computation, and displays the information on the user screen.

3. Requirements for Developing WSN Lab Projects

This section describes the basic hardware and software requirements for deploying a wireless sensor network. The hardware components of a sensor network typically include the following:

a) Motes: A mote is a low-powered computer with a radio transmitter capable of forming ad hoc communication with other motes. A mote may be connected to one or more sensor boards.
b) Sensor boards: A sensor board is a chip on which one or more sensors are present.
c) Gateway: A gateway is a device which is responsible for injecting queries into the sensor network, gathering responses from the network, and presenting the responses to the user's workstation. The gateway communicates with the WSN through short-range wireless links, and interacts with the user directly or remotely through a wired or mobile communication network.
d) Monitoring workstation: The monitoring workstation is a PC with required compatible software installed, and is used by the user to configure the WSN, to submit queries to the network, or to view the data collected by the network.

The following are the software requirements:

a) Operating system: Typically a special operating system such as TinyOS is used for the motes. TinyOS is a commonly accepted open-source development environment for sensor applications.
b) NesC editor and compiler: NesC language is commonly used to write the programs for motes.
c) Program Installer: This software is required to install the compiled program onto the motes.
d) Packet analyzer: This is either a packaged software or a custom-built application running on base station to analyze the packet, perform computation on raw data, and display information on the screen.

Generally all of the above software are available in the TinyOS. A developer only needs to install the TinyOS package. In addition, separate software packages are also available; each package provides a number of software to perform the tasks. Furthermore, a database can be maintained to store the information collected by the sensor network [4]. In this paper we are dealing with simple projects and thus do not store the collected data.

4. Development of Lab Projects

Wireless sensor networks have significant impact upon the efficiency of military and civilian applications [5], which may be classified into three classes: (a) Data collection, (b) Surveillance, and (c) Object tracking. In this section we discuss two projects, which serve as examples of data collection and surveillance. Project 1 demonstrates the use of WSN to collect data from the environment. Project 2 demonstrates the use of WSN for surveillance.

4.1. Project 1: Data collection from the environment

This project involves the development of a WSN for one of the most common applications, i.e., data collection from the environment.

Project Title: Simple Data Collection using Wireless Sensor Networks

Target Classes: This lab project may be used in upper level undergraduate classes like Network Protocols, Wireless Sensor Networks, et al., or in graduate classes involving Wireless Sensor Networks.

Learning objectives: Students will learn the following after having successfully completed the lab.

- How a wireless sensor network operates in general
- How to compile a program
- How to install the complied program in mote
- How to plug the sensor board onto the mote if it is not soldered
- How to configure and run the application at base station to receive data from the network

Tools Utilized: Desktop computer, motes (Mica-2), sensor boards (MTS 3120), gateway (MIB 510), the MoteWorks WSN development package.

Requirements: Each student is given the example source codes available in the MoteWorks software package and the relevant documentations of the software packages. Depending on the level of difficulty of the project assigned to a student, the student is required to complete one of the following tasks:

a) *beginning* level (e.g., upper-level undergraduate students) - Students are required to perform two tasks. First, collect data from the environment (e.g., temperature, light, et al) using a single node connected to the base station, i.e., without utilizing the radio communication link. Second, develop a two-mote WSN. One of the motes is connected to the gateway, while the other mote is equipped with sensor boards, which collect data and transmit them to the mote at the gateway, which further transmits the data to the base station. Wireless radio communication is enabled in the second case.

b) *intermediate* level (e.g., advanced undergraduate or graduate students) - Students are required to collect data from the network using multi-hops, i.e., data is collected by node1, transmitted to node2, which further transmits the data to the mote attached to the base station.

Problem classification: The application can be classified as a study and programming project, because it involves studying the documentations and source codes, and developing/implementing the network.

How it may be implemented in the lab: Students first install the software package on a desktop computer. They need to configure the environment variables and compile the program, and then install the program into motes. Connect the batteries and the sensor boards to the motes and turn them on. Connect the gateway to the base station and place a mote on the gateway. Compile and execute the base station application.

Level of difficulty: The level of difficulty of the project is easy (for case a) or intermediate (for case b).

Grading criteria and methods: Grades mainly depend on the successful deployment and implementation of the sensor network(s), as well as documentations, presentations, et al.

Students' experiences: As reported by students, working on this lab projects allows the students to learn the fundamentals of WSN with respect to how a WSN works, how the nodes are configured and programmed, and how sensed data are aggregated and sent to the base station. They also have learned the importance of limited battery power and runtime battery replacement problem, short radio range for communication by moving nodes at run time. Besides, the students have reported that they had learned the concepts of new operating system (TinyOS) and programming languages (NesC).

4.2. Project 2: Human detection sensor network

This project helps to deploy a WSN that is capable of detecting the presence of human.

Project Title: Detection of Human Presence using Wireless Sensor Networks

Target Classes: The lab project may be used in upper level undergraduate classes like Network Protocols, Wireless Sensor Networks, et al., or in graduate classes involving Wireless Sensor Networks.

Learning objectives: Students will learn the following after having successfully completed the lab.

- How a wireless sensor network operates in general
- How to analyze the raw data collected from the network
- How to develop an algorithm to detect the presence of human
- What kind of sensor is required to detect the presence of a human or vehicle
- How to program a base station application to apply the human detection algorithm

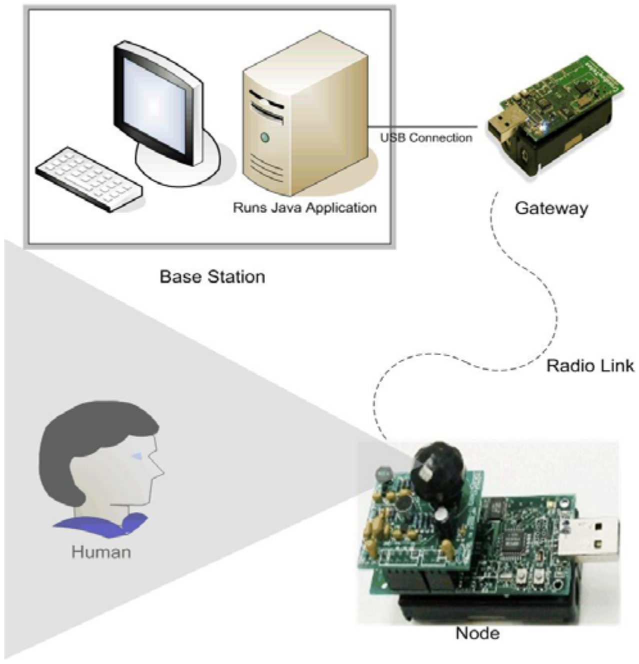


Figure 1. Design of a WSN Detecting Human Presence

Tools Utilized: Listed below are the hardware and software needed for the project.

Hardware required:

- Mote: The *TelosB* (TPR2400) mote, manufactured by Crossbow Inc, is used; it has USB support to connect to other devices or a workstation.
- Sensor Board: The *WiEye* sensor board manufactured by EasySen Inc. is used. It has long-range passive infrared (PIR) sensor with 90-100° wide detection cone, 20-30 feet detection range for human presence. As shown in Figure 4, the sensor board is mounted on a TelosB mote to form a sensor node.
- Gateway: No special gateway is used. The TelosB mote is directly connected to the base station via the USB port; the mote collects data from the sensor network, and sends them to the base station.
- Base Station: This is a typical desktop computer on which Cygwin and Java Virtual Machine are installed

Software required:

- Cygwin is used to configure the WSN. Applications in motes are installed through this command based software. Base station application is also invoked from this software.
- Programmer's Notepad is used to write, compile and debug the NesC program. NesC program is installed in the motes to collect the data from the environment.
- Java SDK 1.6.0 is used to develop a Java application which runs on the base station. This java application collects data from the network, process it and send the output to the base station.

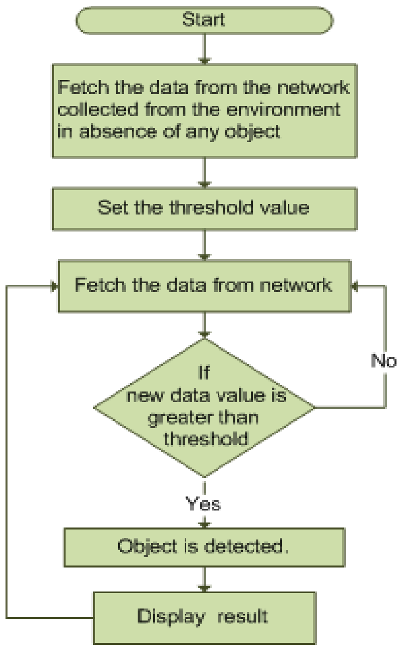
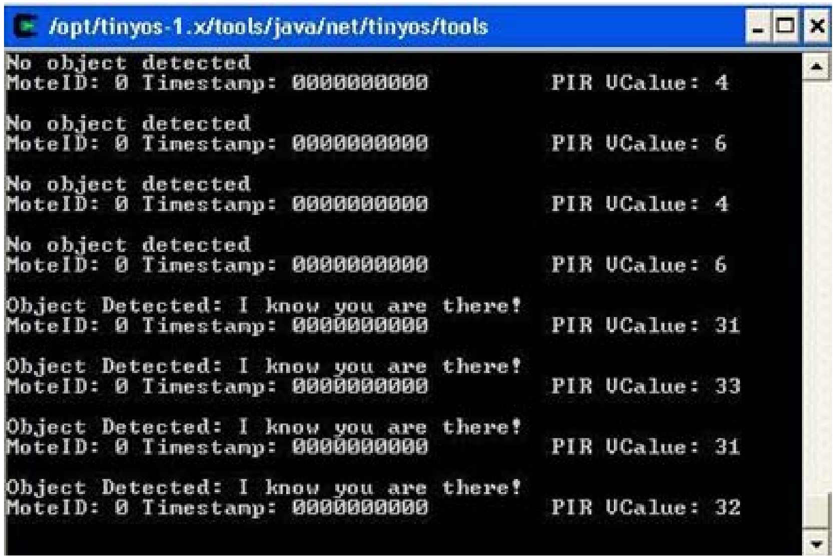

Start
Fetch the data from the network collected from the environment in absence of any object
Set the threshold value
Fetch the data from network
If new data value is greater than threshold
No
Yes
Object is detected.
Display result

Figure 2. Flowchart of the Java Application

Requirements: Each student is given the example source codes available on the EasySen website, the documentation of the motes and the needed software package. Students are required to develop a human detection algorithm and later implement that algorithm to program the base station application. Appropriate sensors should be used to collect raw data from the environment. The base station application will analyze the data to detect the presence of human in the environment. The base station application to recognize a human can be programmed as a Java application. A gateway with a mote installed on it is connected to the base station. The gateway collects data from the network and transmits them to the base station. Each node in the sensor network consists of a mote powered by battery and a WiEye sensor board from EasySen Inc. Figure 1 illustrates the design of the system.

Developing the human detection application – Sample Java application for reading sensor channels of the SBT80 sensor over the TelosB motes are available at the EasySen website [1]. When developing the application, students may modify those sample codes to suit the need of the project. The application runs on the base station to collect data from the sensor network, and analyze the collected data to detect the presence of human. The initial data collected from the environment (without human presence) is used to set the threshold value (representing the condition of no human presence). Later if human comes in the vicinity of the sensor it compares the current value with the threshold value to make the decision of human presence. The algorithm developed using this logic is show in Figure 2.


```
/opt/tinyos-1.x/tools/java/net/tinyos/tools
No object detected
MoteID: 0 Timestamp: 0000000000          PIR VCalue: 4

No object detected
MoteID: 0 Timestamp: 0000000000          PIR VCalue: 6

No object detected
MoteID: 0 Timestamp: 0000000000          PIR VCalue: 4

No object detected
MoteID: 0 Timestamp: 0000000000          PIR VCalue: 6

Object Detected: I know you are there!
MoteID: 0 Timestamp: 0000000000          PIR VCalue: 31

Object Detected: I know you are there!
MoteID: 0 Timestamp: 0000000000          PIR VCalue: 33

Object Detected: I know you are there!
MoteID: 0 Timestamp: 0000000000          PIR VCalue: 31

Object Detected: I know you are there!
MoteID: 0 Timestamp: 0000000000          PIR VCalue: 32
```

Figure 3. Cygwin Screen Showing Sample Output of the Java Application Detecting Human Presence

Anticipated results – If no human is present in the vicinity of the sensor, a message saying "No object detected" should be shown on the screen. If a human is detected, a message saying "Object detected: I know you are there!" should be shown on the screen. Figure 3 is a sample output screen of the application.

Problem classification: The application can be classified as a study and programming project, because it involves studying documentations and source codes, and developing/implementing the algorithm and deploying a network.

How it may be implemented in the lab: Students will install the software package on a desktop computer. They need to develop and implement the abject detection algorithm. Follow the programming sequence as outlined in the EasySen documentation[2] to configure the environment variables, compile the program, and install the program on to the motes. Install the batteries and the sensor boards on the motes and turn them on. Connect the gateway to the base station. Compile and execute the base station application.

Level of difficulty: The level of difficulty of the project is hard.

[1] EasySen SBT80 Multi-Modality Wireless Sensor Board, http://www.easysen.com/support/SBT80v2 (9-15-2007)

[2] The programming sequence is attached in the Appendix of this paper.

Grading criteria and methods: Grades mainly depend on the successful deployment and implementation of the algorithm and the sensor network, as well as documentations, presentations, et al. The algorithm shown in Figure 2 can be used as a scale to compare the algorithms developed by the students.

Students' experiences: As reported by students, working on this lab projects allowed the students to learn the use of WSN for real-world applications. They also developed understanding of related issues such as how to develop and implement the algorithm according to the requirements, how sensed data are aggregated and sent to the base station, how the received data packets are analyzed. Some of the students also proposed interesting ideas and algorithms to extend the existing system to effectively detect human motion.

5. Conclusion and future work

Development of hands-on lab projects provides in-depth understanding of the subject. In computer science courses, too often simulation is the chosen approach when it comes to hands-on lab projects. Due to the cost and difficulty of implementing a wireless sensor network using actual devices, emulation is seldom the approach chosen by instructors when developing class projects. In this paper, we present both basic and intermediate-level sensor network projects, which may be adopted by the interested instructor to bring hands-on emulation experiences into his/her classes, providing students the opportunities to develop realistic wireless sensor network applications. Students learn how to modify an existing WSN application to incorporate new algorithms. We have explained how a student with no prior knowledge of WSN can develop a network from the scratch. We are currently developing algorithms to track the movement of humans and vehicles. Part of our future work is to incorporate the new algorithms into labs, which should be suitable for students interested in learning more advance concepts of wireless sensor networks.

References

1. Raghavendra, C.S., K.M. Sivalingam, and T. Znati, *Wireless Sensor Networks*. ERCOFTAC Series. 2004: Springer.
2. Tran, S. P. M. and T. A. Yang (2006). "Evaluations of target tracking in wireless sensor networks". *Proceedings of the 37th SIGCSE technical symposium on Computer science education SIGCSE '06, ACM SIGCSE Bulletin*, 2006..
3. Werner-Allen, Geoffrey, Patrick Swieskowski, and Matt Welsh. "MoteLab: A Wireless Sensor Network Testbed". In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN'05)*, Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS), April 2005..
4. Salatas, V. (2005). "Object tracking using wireless sensor networks". M.S. Thesis. Naval Postgraduate School, California.
5. Singh, I. (2007). "Real-time Object Tracking with Wireless Sensor Networks". M.S. Thesis. Luleå University of Technology.

Appendix: Procedure of programming the EasySen sensors

The following is the programming sequence as outlined in the data sheet of the SBT80[3] sensors; the same procedure can be used for programming WiEye[4] sensors. The sample application folder *SBT80app* consists of 2 header files (*SBT80ADCmap.h* and *SBT80Msg.h*), a configuration file (*MobileNode.nc*), and a module file (*MobileNodeM.nc*). It samples all 8 sensor channels in a sequence and transmits data to a PC. The file *ListenSBT80v2.java* reads and displays the packets transmitted using the *MobileNode* application above.

1. Program one TmoteSky / TelosB [5] mote with the MobileNode application given here.
   (a) Copy and unzip the folder SBT80app into the /opt/tinyos1.x/apps folder in your TinyOS installation
   (b) Connect TmoteSky mote to the PC USB connector and execute the command make tmote install inside the folder SBT80app to install the application
   (c) Connect the sensor board SBT80v2 to the expansion connector of the TmoteSky mote

2. Program a base station with the generic TOSBase application.
   Connect another TmoteSky mote to the PC USB connector and execute the command make Tmote install inside the folder /opt/tinyos1.x/apps/TOSBase.

3. Display sensor readings on the controlling workstation.
   (a) Copy the java program ListenSBT80v2.java to the following folder in your TinyOS installation: /opt/tinyos1.x/tools/java/net/tinyos/tools.
   (b) Compile by executing the command “javac ListenSBT80v2.java” inside the same folder.
   (c) Set the *MOTECOMM* variable to read the USB port properly according to the TinyOS tutorial.
   (d) Run the java application by executing the command “java net.tinyos.tools.ListenSBT80v2”.

This should produce a display of sensor readings from all 8 channels on the Cygwin window. These steps are given to test the working of SBT80 and same is used to run the working of WiEye. In step 3.C, in order to configure the *MOTECOM* variable, first run the "motelist" command to get the port number on which TelosB is connected.

Once you get the port number, use the export command to configure the *MOTECOM* variable. The command is “*export MOTECOM=serial@COM4:telosb*” or “*export MOTECOM=serial@COM4:telos*”. The number after COM is the port number which is written 4 here.

[3] EasySen. "EasySen SBT80 Datasheet". Retrieved September 15, 2007, from http://www.easysen.com/support/SBT80v2/DatasheetSBT80v2.pdf
[4] EasySen. "EasySen WiEye Datasheet". Retrieved September 15, 2007, from http://www.easysen.com/support/WiEye/DatasheetWiEye.pdf
[5] Crossbow. "Crossbow, TelosB, mote platform". Retrieved December 15, 2007, from http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf.