DEVELOPMENT OF EMULATION-BASED PROJECTS FOR TEACHING WIRELESS SENSOR NETWORKS

T. Andrew Yang **Deepesh Jain**
Division of Computing and Mathematics
University of Houston – Clear Lake
Houston, Texas 77058
yang@uhcl.edu deeptansh@gmail.com

Bo Sun
Dept. of Computer Science
Lamar University
Beaumont, TX USA 77710
bsun@my.lamar.edu

Abstract
Wireless technologies have achieved dramatic growth in recent years. Wireless sensor networks (WSN), in particular, have captured the interest of various sectors. The increasing popularity of WSN has motivated universities to include the subject in their curricula. Effective learning and teaching of WSN, however, require the students to gain hands-on experiences in developing WSN projects, which help the students to understand the strength and limitations of this new technology. In this paper we report our experience of teaching WSN by requiring students to develop emulation-based labs following pre-designed template projects. The projects make it easier for the instructor adopt WSN labs in the teaching, and help to smooth the student's learning curve.

1. INTRODUCTION

A wireless sensor network (WSN) 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 [1]. Normally the unit that consists of the radio module and the on-board processor is called *mote*. The most essential feature of wireless sensor networks 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 [2]. A typical WSN consists of a set of nodes collecting data from the environment; these nodes transmit the collected data to a base station. The *base station* is generally a computer connected to a *gateway*, which is a device that collects data from the sensors. An application running on the base station analyzes the received data, performs appropriate computation, and displays the information on the user screen.

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]. It is typically 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 projects that could be easily adopted by the instructor, and could help the students get started in deploying sensor networks.

Instead of relying on simulation software, this paper focuses on emulation-based projects, by using actual devices (motes, sensor boards, et al.) when designing labs. The reason that most WSN projects are simulation-based is that implementation of WSN projects with actual devices is more difficult and demanding. Actual hardware devices are needed, thus increasing the cost and complexity; in contrast, the simulation approach requires only the simulation software. Emulation becomes particularly complicated when a large number of sensors are deployed. Another limitation is battery power constraint; it may be cumbersome to replace the battery while the WSN is already deployed. The sensors, for example, may be hard to reach.

The rest of the paper first discusses the hardware/software requirements of developing WSN labs; it then presents the template projects that we have developed.

2. REQUIREMENTS FOR DEVELOPING WSN LAB PROJECTS

The hardware components of a sensor network typically include the following:

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

The following are the software requirements:

- Typically a special operating system such as *TinyOS* is used for the motes. *TinyOS* is an open-source development environment for sensor applications.
- The *NesC* language is commonly used to write the programs for motes. Therefore the *NesC editor and compiler* are needed.
- The *program installer* software is required to install the program into the motes.
- A *packet analyzer* is an application running on the base station to analyze the packet, perform computation on raw data, and display information on the screen.

Generally a developer only needs to install the *TinyOS* package to get the above listed software. Although a WSN may maintain a database to store the collected data [4], in this paper we deal with simple projects that do not need a database.

3. DEVELOPMENT OF LAB PROJECTS

Wireless sensor networks may be classified into three classes: data collection, surveillance, and object tracking [5]. In this section we discuss two projects: Project 1 demonstrates the use of WSN for data collection, while project 2 is for surveillance.

3.1. 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 project may be used in graduate or upper level undergraduate classes like Network Protocols, Wireless Sensor Networks, et al.

<u>Learning objectives:</u> Students will learn the following after completing the lab.

- How a wireless sensor network operates in general
- How to compile a *NesC* program
- How to install the compiled program in motes
- How to plug the sensor board onto the mote if it is not soldered
- How to configure and run the application to receive data from the network

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

<u>Requirements:</u> Each student is given the source codes of example applications (available in the *MoteWorks* software package) and the relevant documentations of the software packages. A student is required to complete one of the following tasks:

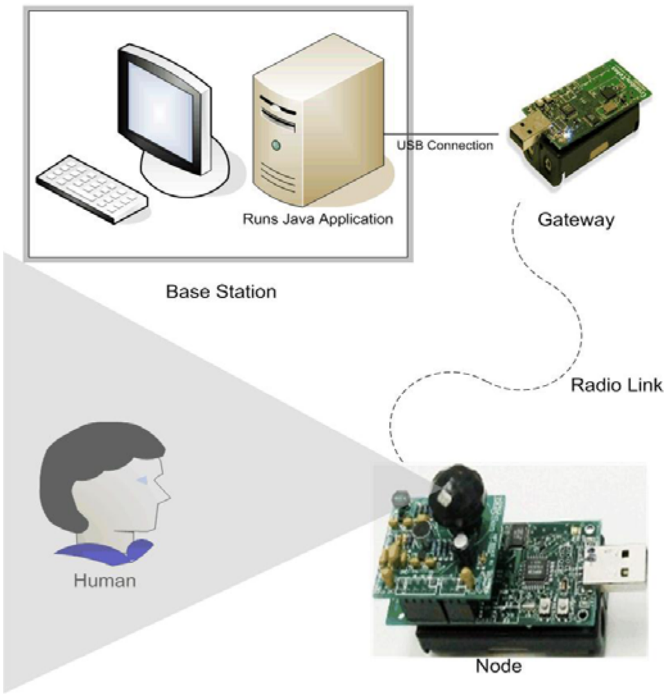


Figure 1. Design of a WSN Detecting Human Presence

a) *Beginning* level - Students 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 on the gateway, which further transmits the data to the base station. Wireless radio communication is enabled in the second case. Figure 1 illustrates a similar setup.

b) *Intermediate* level - 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.

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

<u>How it may be implemented in the lab:</u> 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 application on the base station.

<u>Level of difficulty</u>: The difficulty level is easy (case *a*) or intermediate (case *b*).

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

Students' experiences: As reported by students, working on the 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 had learned the significance of limited battery power, runtime battery replacement problem, and the short radio range issue when moving nodes at run time. Besides, the students reported that they had learned the concepts of new operating system (*TinyOS*) and programming languages (*NesC*).

3.2. Project 2 - human detection sensor network

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

Project Title: Detection of Human Presence using Wireless Sensor Networks

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

Learning objectives: Students will learn the following after completing 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 humans
- 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

Tools Utilized:

- The *TelosB* (TPR2400) manufactured by Crossbow, Inc. is used as the *mote*.
- 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 1, the sensor board is mounted on a TelosB mote to form a node.
- No special gateway device is needed. The *gateway* function is implemented by having a TelosB mote directly connected to the base station.
- The *base station* is a typical desktop computer with Cygwin and Java VM.
- *Cygwin* is used to configure the WSN, including installing the application into motes, and invoking a base station application.
- *Programmer's Notepad* is used to write, compile and debug the *NesC* program, which is installed in the motes to collect data from the environment.
- *Java SDK 1.6.0* is used to develop a Java application, which collects data from the network, processes the data, and sends the output to the user screen.

Requirements: Each student is given the source codes of example applications, available on the EasySen website (http://www.easysen.com/support/SBT80v2), relevant documentations, and the needed software. Students first 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, programmed as a Java application, will analyze the data to detect the presence of humans in the environment. A mote directly connected to the base station is used as the gateway, which 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. When developing the application, students may modify those 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 humans. 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 a 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 is show in Figure 2. 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!" should be shown. Figure 3 is a sample output screen of the application.

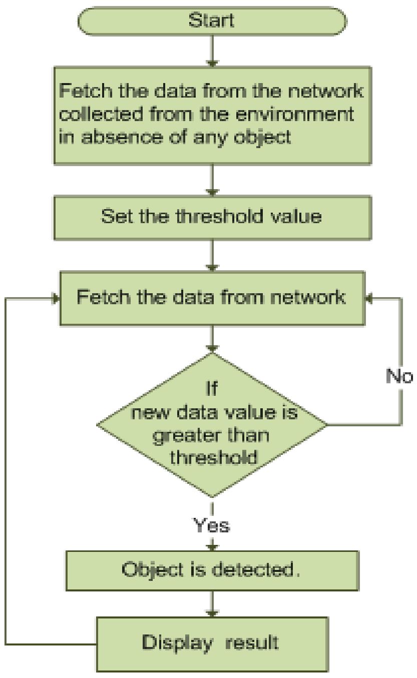
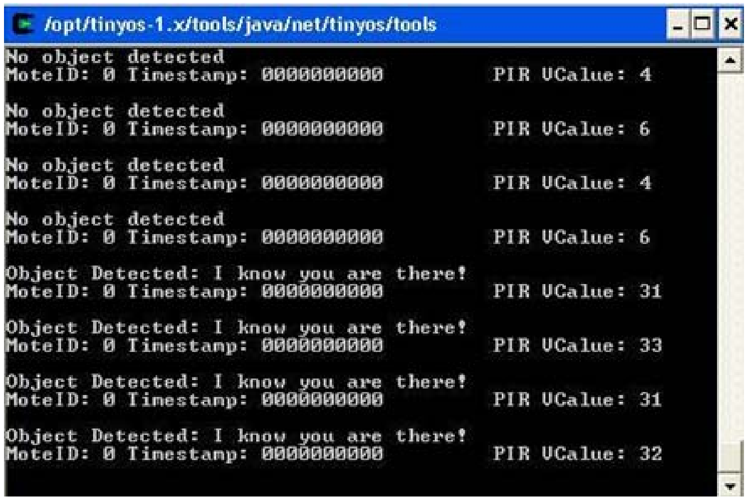

Start
Fetch the data from the network collected from the environment in absence of any object
Set the threshold value
Fetch the data from network
If new data value is greater than threshold
No
Yes
Object is detected.
Display result

Figure 2. Flowchart of the Java Application

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

How it may be implemented in the lab: Students install the software package on a desktop computer. They need to develop and implement the object detection algorithm. Follow the programming sequence as outlined in the EasySen documentation to configure the environment variables, compile the program, and install the program on to the motes.



```
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. Sample Output of the Java Application Detecting Human Presence

Note: http://sce.uhcl.edu/yang/ProcedureProgrammingEasySenSensors.htm consists of a summary of the procedure of programming the EasySen sensors.

Level of difficulty: hard

Grading criteria and methods: Grades mainly depend on the successful implementation of the algorithm and deployment of 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: The project allowed the students to learn the use of WSN for surveillance. They developed understanding of related issues such as how to develop an algorithm according to the requirements, and how the collected data are analyzed.

4. DISCUSSION AND FUTURE WORK

Development of hands-on labs provides in-depth understanding of the subject. In computer science courses, too often simulation is the chosen approach when it comes to 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, therefore providing 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 advanced concepts of wireless sensor networks.

ACKNOWLEDGMENT

The authors are partially supported by the National Science Foundation, and the Texas Higher Education Coordination Board's Advanced Research Program.

REFERENCES

1. Tran, S.P.M. and T.A. Yang. Evaluations of target tracking in wireless sensor networks, in *Proceedings of the 37th SIGCSE technical symposium on Computer science education SIGCSE '06, ACM SIGCSE Bulletin*. 2006.
2. Raghavendra, C.S., K.M. Sivalingam, and T. Znati, *Wireless Sensor Networks*. ERCOFTAC Series. 2004: Springer.
3. Werner-Allen, G., P. Swieskowski, and M. Welsh, "MoteLab: A Wireless Sensor Network Testbed," in *Information Processing in Sensor Networks, 2005. IPSN 2005 Fourth International Symposium*, 2005, pp. 483- 488.
4. Salatas, V., *Object tracking using wireless sensor networks*. MS Thesis, Naval Postgraduate School: California. 2005.
5. Singh, I., *Real-time Object Tracking with Wireless Sensor Networks*. MS Thesis, Luleå University of Technology. 2007.