ENHANCING TRANSIT SAFETY AND SECURITY WITH WIRELESS DETECTION AND COMMUNICATION TECHNOLOGIES

Final Report

FDOT BD549 RPWO # 45

Prepared for

Florida Department of Transportation
605 Suwannee Street, MS 30
Tallahassee FL 32399

Prepared by

National Center for Transit Research
at the Center for Urban Transportation Research
University of South Florida
4202 E. Fowler Ave. CUT100
Tampa FL 33620-5375

November 2008

DISCLAIMER

The opinions, findings, and conclusions expressed in this publication are those of the author(s) who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the views or policies of the Florida Department of Transportation or the Research and Special Programs Administration. This report does not constitute a standard, specification, or regulation.

The report is prepared in cooperation with the State of Florida Department of Transportation and the U.S. Department of Transportation.

1. Report No. FDOT BD 549 WO45	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Enhancing Transit Safety and Security with Wireless Detection and Communication Technologies		5. Report Date November 2008 6. Performing Organization Code
7. Author(s) Sean Barbeau, Miguel Labrador, Phil Winters, and Nevine Labib Georggi		8. Performing Organization Report No. NCTR-77714-00
9. Performing Organization Name and Address National Center for Transit Research at the Center for Urban Transportation Research, University of South Florida 4202 E. Fowler Ave. CUT100 Tampa FL 33620-5375		10. Work Unit No. (TRAIS) 11. Contract or Grant No. FDOT-BD549 WO #45
12. Sponsoring Agency Name and Address Florida Department of Transportation 605 Suwannee Street, MS 30 Tallahassee FL 32399		13. Type of Report and Period Covered FINAL 14. Sponsoring Agency Code
15. Supplementary Notes Sponsored by a grant from the Florida Department of Transportation and the U.S. Department of Transportation		

16. Abstract

Public transportation systems are among the most open public facilities in the world and susceptible to breaches of security. Reconciling the need for workplace safety and security with budgetary pressures requires new approaches to increase the effectiveness of existing solutions while preserving flexibility and low costs. An inexpensive sensor-based intrusion detection system that remotely monitors and notifies on- and/or off-site personnel of any incidents can significantly multiply the observational effectiveness of a few onsite safety or security personnel monitoring a facility. The advancement in the miniaturization of circuits has produced small computing devices allowing the development of pervasive applications that only a few years ago were not possible. The combination of such devices with wireless networks and micro-electro-mechanical systems technology provides a new platform for research and development of innovative monitoring applications. This project developed a low-cost, scalable, real-time intrusion detection and remote notification system called WSN-IRNS, using wireless sensor networks with the purpose of enhancing the safety and security of transit facilities. WSN-IRNS provides a cost-effective alternative or supplement to traditional wired security systems for protecting vulnerable areas and facilities such as garages, tunnels, and transit yards. The Internet-connected system supports real-time intervention by notifying personnel upon the detection of an intrusion through multimedia messages, which can include captured camera images that are delivered directly to mobile phones. Field tests have successfully demonstrated the proof-of-concept of the system, although adjustments and fine tuning of system parameters will be needed for environment-specific installations.

17. Key Word	18. Distribution Statement		
Safety, Security, Wireless Sensor Networks, Intrusion Detection, MEMS, WSN	Available to the public through the National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield VA 22161, (703) 487-4650, http://www.ntis.gov/, and through the NCTR website at http://www.nctr.usf.edu/		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 53	22. Price

Acknowledgements

This report is prepared by the National Center for Transit Research (NCTR) through the sponsorship of the Florida Department of Transportation (FDOT) and the U.S. Department of Transportation.

Florida Department of Transportation Project Managers:

Mike Johnson and Victor Wiley, FDOT Public Transit Office

Project Team:

Sean Barbeau, Principal Investigator and Research Associate, CUTR
Philip L. Winters, TDM Program Director, CUTR
Nevine Labib Georggi, Research Associate, CUTR
Miguel Labrador, PhD, College of Engineering, USF
Rafael Perez, PhD, College of Engineering, USF

Computer Science and Engineering Students Team:

Alfredo Perez
Clayton Gandy

Executive Summary

Wireless Sensor Networks (WSN) consist of small computing devices with wireless communication capabilities equipped with specialized sensors. This technology can automatically capture and report data in difficult, dangerous, or remote environments. These capabilities have allowed the development of myriad useful applications ranging from the monitoring of endangered animal species to home automation. After presenting the state of the art in WSNs, this report describes the development of an intrusion and remote notification system called WSN-IRNS, based on the technology that monitors sensitive areas in transit facilities such as garages, tunnels, and transit yards. WSN-IRNS was developed as a software system for wireless sensor networks using commercially available hardware. The Internet-connected system supports real-time intervention by notifying personnel upon the detection of an intrusion through multimedia messages, which can include captured camera images that are delivered directly to mobile phones. Field tests have successfully demonstrated the proof-of-concept of the system, although adjustments and fine-tuning of system parameters will be needed for environment-specific installations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I.

INTRODUCTION

The Center for Urban Transportation Research (CUTR) and the Department of Computer Science and Engineering (CSE) at the University of South Florida (USF), with funding from the Florida Department of Transportation (FDOT) and the United States Department of Transportation (USDOT), have established an ongoing partnership over the past several years that focuses on the development of cutting-edge transportation applications through the use of location-aware and mobile technologies. The following sections of this chapter present the research background, goals and objectives, and an illustrated example of the problem this research addresses.

RESEARCH BACKGROUND

Safety and security in transit systems are of major concern because of the open nature of transit and the high density of people and goods transported along thousands of miles of road or track. Maintaining the safety of the public and transit personnel while effectively managing incidents creates challenging demands for transit and emergency management staff. To meet the varying safety and security needs of different transit agencies within their respective resources, flexible solutions are needed that are scalable in size and cost of implementation, as well as deployable in the various environments that make up the transit industry.

PROBLEM STATEMENT

Public transportation systems are among the most open public facilities in the world and susceptible to breaches of security. This challenge is well understood by the industry. According to a Transit Cooperative Research Program (TCRP) report (19), Intrusion detection for Public Transportation Facilities Handbook,

Unwanted intrusion into public transportation facilities is an industry-wide problem that affects safety, security, service reliability, productivity, claims, and customer relations. The problem has been approached with a variety of both high

and low technology solutions producing a wide range of results. Applications address both prevention and detection and include Access Control Systems (ACS) and Intrusion Detection Systems (IDS). The heightened concern for safety and security in public transportation facilities and vehicles has led to increased investment in both ACS and IDS. At the same time, changing economic conditions have resulted in reduced funding and more scrutiny in capital and operating budgets.

This 2003 report identified the need for more research in integration methods for Intrusion Detection Systems (IDS) and Access Control System (ACS) and methods and procedures for video system integration with IDS. This publication also states that the largest long-term impact for an IDS system comes from the additional labor required to repair, service and monitor the system. A relatively inexpensive sensor-based intrusion detection system that remotely monitors and notifies on- and/or off-site personnel of any incidents can significantly multiply the observational effectiveness of a few onsite safety or security personnel monitoring a facility. An intelligent alert system that automatically alerts designated officials based on the nature of the detected incident helps staff of limited size effectively respond to emergencies. Transit agencies can use a system with these capabilities as a pre-emptive mechanism to avoid potential disasters and as a catalyst for immediate response to mitigate the effects of an incident.

STUDY GOALS AND OBJECTIVES

This project focused on the integration of remote wireless sensor networks (WSN) into an existing two-way location-based multimedia communication system for global positioning system (GPS)-enabled mobile phones. This system hereinafter referred to as WSN-IRNS, was developed to monitor unattended remote parking facilities and alert security personnel about intrusions in real-time using multimedia formats. The primary project tasks were:

- Conduct literature review and assess technology options for location-aware wireless sensors and location-enabled cell phones.
- Identify a transit agency and a safety/security application to which the WSN technology will be applied.

- Develop a method to establish and locally monitor a WSN and communicate between the WSN and the Internet.
- Modify the existing two-way location-based communication system to incorporate data from a wireless sensor network.
- Develop a method for alerting appropriate personnel once a sensor event has been detected.

REPORT ORGANIZATION

After this introductory chapter, Chapter 2 provides a literature review on intrusion alert systems and wireless sensor monitoring systems projects. Chapter 3 provides a summary on the state of the art in wireless sensor networks technology. Chapter 4 describes the development process of the WSN-IRNS. Chapter 5 provides the experimental results of system testing while Chapter 6 provides conclusions and recommendations for future research.

CHAPTER II.

LITERATURE REVIEW

There has been significant research on safety and security of transportation systems in previous years. However, most of the applications were developed using long-range lasers. While this can be an effective technique for surveillance, it is rather expensive. WSN-IRNS provides a cost-effective alternative for intrusion detection and reporting in order to support safety and security operations at transit agencies. In (1), an overview of Intrusion Detection Systems for transit operations is provided, giving state-of-the-art practices as of 2003. In addition, a recent publication by the Federal Rail Administration, "State-of-the-Art Technologies for Intrusion and Obstacle Detection for Railroad Operations," provides a complete summary of IDS projects (1). The following sections highlight some relevant applications.

LaserGrab /FenceGrabber Systems

Developed by the Swedish company LaserOptronix, these two systems provide intrusion detection. The LaserGrab illustrated in Figure 1 is composed of laser radar that detects obstacles in distances greater than 100 meters, takes a picture of the detected intrusion, and sends it via wireless communication to a remote location. Tailored to rail operations, the remote locations are trains traveling on the railroad where the system is installed.

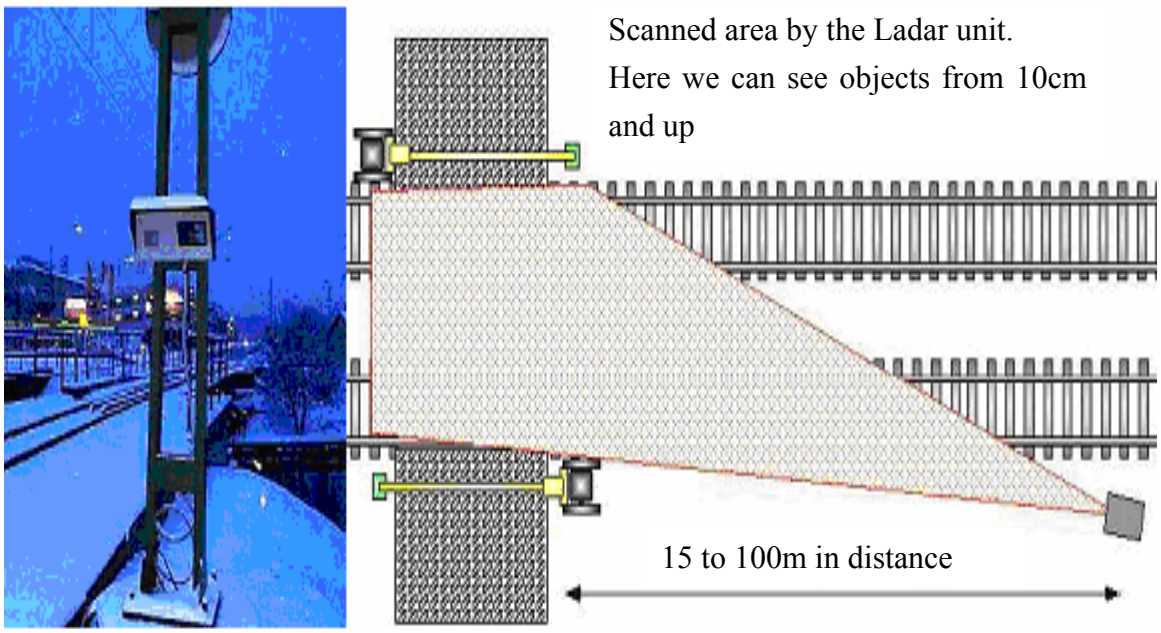


Figure 1 – A LaserGrab System

Source: Federal Railroad Administration. "State-of-the-Art Technologies for Intrusion and Obstacle Detection for Railroad Operations," February 2007. pp. 12

The second system illustrated in Figure 2 is the FenceGrabber composed of laser radars that provide a virtual fence against intruders. The main functionality of this system is to deploy it in areas where physical barriers cannot be installed, such as ponds or swamps. When any of the lasers sensors detect an intrusion, the system generates an alarm. A camera can be connected to the system and the images captured can be relayed with the alarm to a remote station. The maximum range for the FenceGrabber is 200 meters.

Figure 2 – A FenceGrabber System

Source: Federal Railroad Administration. "State-of-the-Art Technologies for Intrusion and Obstacle Detection for Railroad Operations," February 2007. pp. 12

Massachusetts Bay Transportation Authority System

The Massachusetts Bay Transportation Authority (MBTA) system is used for the detection of intruders in the Silver Line Bus Rapid Transit dedicated tunnel. This system uses video equipment to detect trespassers entering the dedicated tunnel. The system is designed to trigger an alarm if a non-MBTA vehicle or object enters the tunnel. The video system can be reprogrammed to detect different classes of situations, and react to different classes of anomalies, such as stalled vehicles and traffic delays. A caption from a video detection illustration is shown in Figure 3.

Figure 3 – A Caption from an MBTA Video System in a Tunnel
Source: Federal Railroad Administration. "State-of-the-Art Technologies for Intrusion and Obstacle Detection for Railroad Operations," February 2007. pp. 17

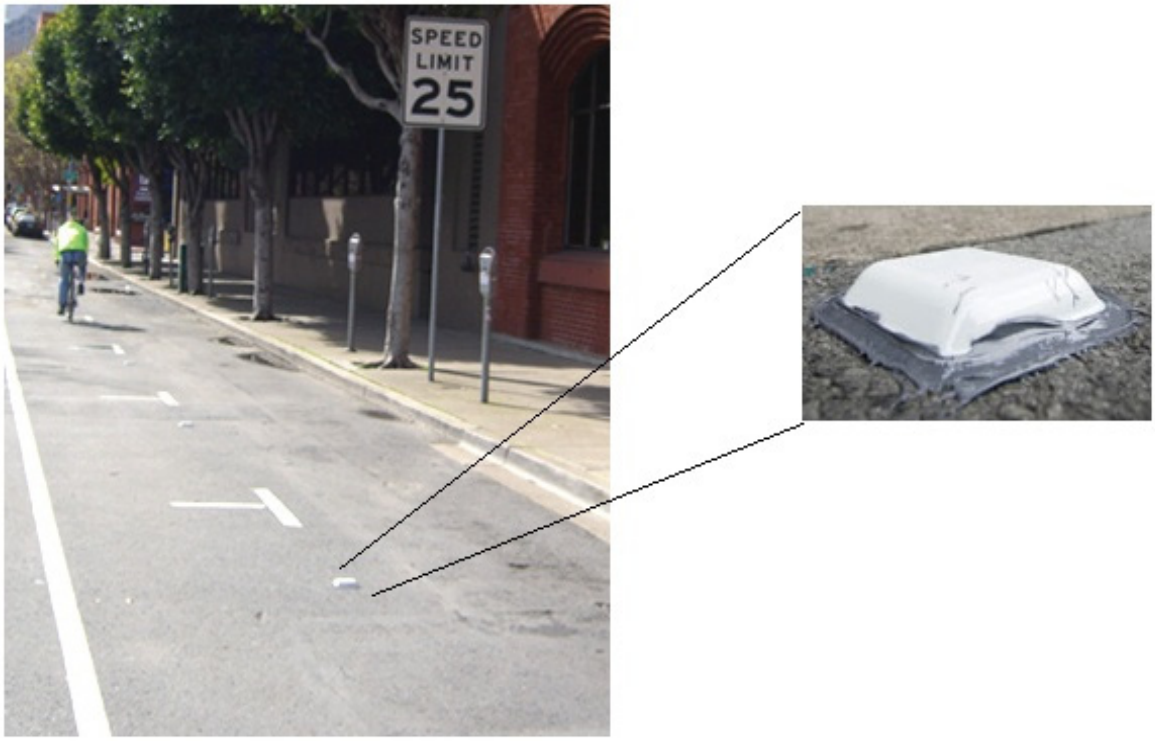
AWARE Pilot Project along a South Florida Corridor

Developed by Nestor Traffic Systems, the Advanced Warnings and Alerts for Railroad Engineers (AWARE) project is an extension of the Nestor's Rail Crossing Guard system to demonstrate video monitoring on grade crossings (3). The complete system provides alerts and displays real time video captured from the crossing to a remote user or a train using the railroad. It provides information regarding hazardous conditions in a railroad grade crossing helping to reduce the risk of train-vehicle collisions. This system is composed of cameras and T1 data communication lines connected to a central control station that broadcasts the information to the trains. This system is tailored for detection of vehicles and its application to pedestrian or other obstacles was not extended as part of that project.

Streetline Vehicle Wireless Sensor System for Parking Spots

Streetline Inc, a company in San Francisco CA has developed a system for monitoring parking spots in congested cities (5). Wireless sensor network devices that are installed in the pavement continuously store information about

the occupation of such spots and house a client-server service. The users can check their smart phones and/or look at public information screens to determine the location of an available parking spot. Figure 4 demonstrates the wireless sensor nodes that Streetline deploys.



.

Figure 4 – Streetline's Wireless Sensor System for Parking Spots
Source: Streetline Systems, Service, and Applications Datasheet, (4)

Wireless Sensors to Protect Parking Structures

Honeywell created a system for monitoring Carbon Monoxide (CO) gas levels in parking garages utilizing WSN (5). This network uses the IEEE 802.15.4 standard and operates for two years without calibration, maintenance, or battery replacement. Honeywell provides this solution to reduce the installation and maintenance costs of the network. The device is shown in Figure 5.

Figure 5 – Honeywell Wireless CO Gas Detector
Source: Honey Vulcain Inc. Web Page at http://www.vulcaininc.com

Intelligent Parking Lot Application Using WSN

A team from the University of Southern California (3) developed a system for monitoring parking lots using WSN, in particular the TelosB wireless sensor motes. It was designed to count the number of vehicles in a parking building garage using magnetic and ultrasonic sensors to provide occupancy information. The objective was to place sensors in strategic areas for each floor of the building to avoid installing a wireless sensor for every parking spot.

CHAPTER III.

SUMMARY OF STATE OF THE ART

The research team conducted an analysis of current technologies in wireless sensor networks technology to evaluate the feasibility of using these devices as intrusion detection systems.

INTRODUCTION TO WIRELESS SENSOR NETWORKS

Wireless Sensor Networks (WSN) are computer networks composed of small, battery-powered computing devices that are deployed in areas of interest for sensing, monitoring, and reporting data about the environment. Normally, wireless sensor networks are deployed in areas difficult to access or areas dangerous to humans. Wireless sensor networks can also take actions based on what they sense. These networks are usually referred to as "wireless sensor and actuator networks."

The components of a wireless sensor networks are wireless nodes/motes, sensors, actuators, and base stations.

Wireless Nodes

Wireless nodes, or motes, are small, battery-powered devices with communication capabilities. Usually these nodes run an Open Source operating system called TinyOS, and are programmed in a language called NesC. Recently, new versions of these nodes can be programmed in Java or in Microsoft.NET. The function of the wireless nodes is to create the wireless network that allows communication between the sensing points and the base stations. The size of these nodes varies from a 25-cent coin to the size of the palm of a hand. Figure 6 shows three motes devices. The devices shown below are the Crossbow Iris node, Crossbow Mote2, and Crossbow IMote2.

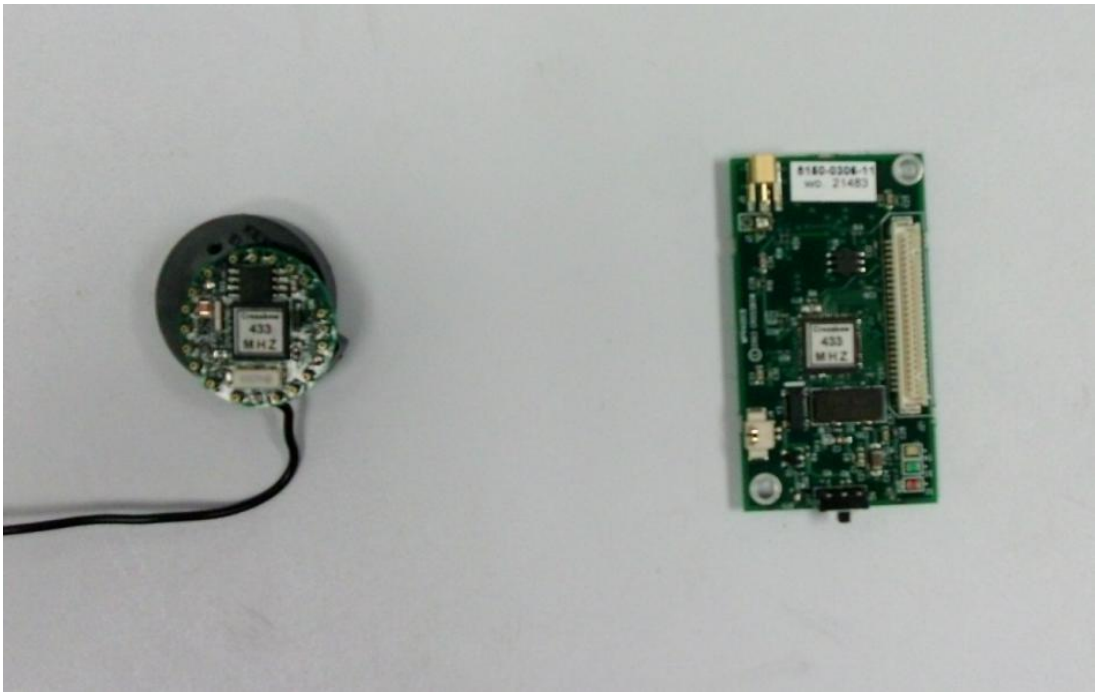


Figure 6 – Wireless Nodes

Sensors

Sensors are the devices attached to the wireless nodes capable of collecting data (i.e., sensing) from the environment. Due to the micro-electro-mechanical systems (MEMS) technology developed over the last few years, several sensors can be integrated in the same board and attached to a wireless node using digital or analog ports. The type of sensor used is dependent on the application. Figure 7 shows three types of sensors: a passive infrared (PIR), magnetometer, and thermometer.

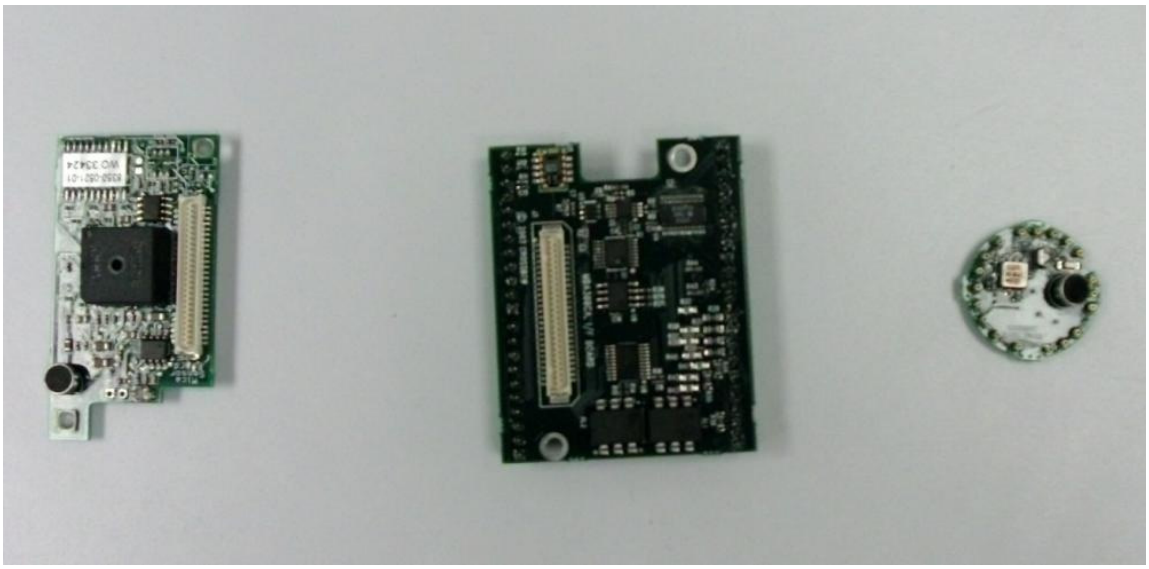


Figure 7 – Example of Available Sensors

Actuators

Actuators are devices that a wireless node uses to take a specific action based on the information it collects. For example, if a wireless node is deployed with a motion detector sensor in a room, an actuator can be attached to the wireless node for turning the light on or off in the room.

Base Stations

Base stations are the gateways that collect the information sent by the wireless nodes. They can work alone or be connected to other networks. In the latter case, the sensed information can be transmitted through other networks, such as the Internet, and displayed in any computer connected to the network. Figure 8 shows a Crossbow's Stargate base station.

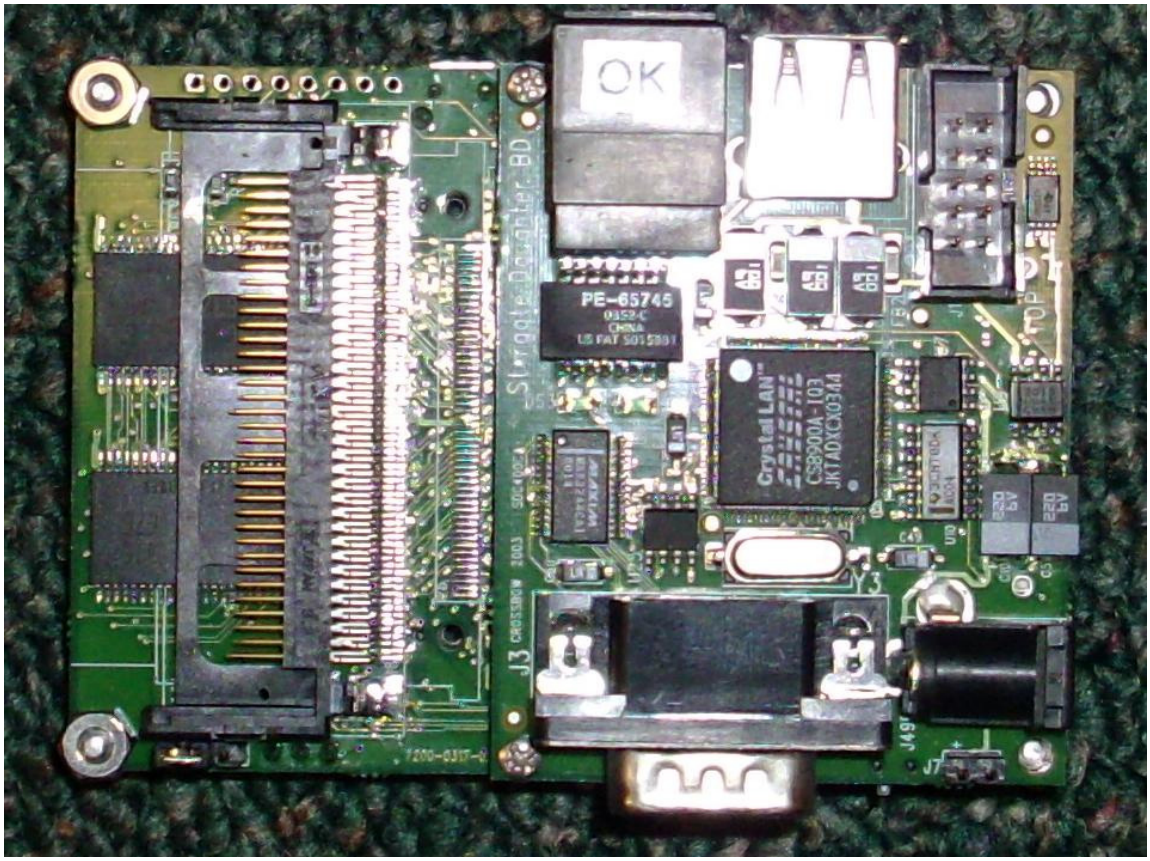


Figure 8 – Crossbow's Stargate Base Station
Source: PlatformX Project with Stargate Website at
http://platformx.sourceforge.net/images/Stargate.jpg

Figure 9 shows a test bed of a wireless network deployment at Harvard University (MoteLab Laboratory (7)). All the nodes are sending information to a node placed in the center of the map that serves as the base station of the wireless sensor network. This WSN is used for testing indoor applications and utilizes a web page for loading programs and schedule applications in the test bed, as well as measuring light, humidity, and temperature.

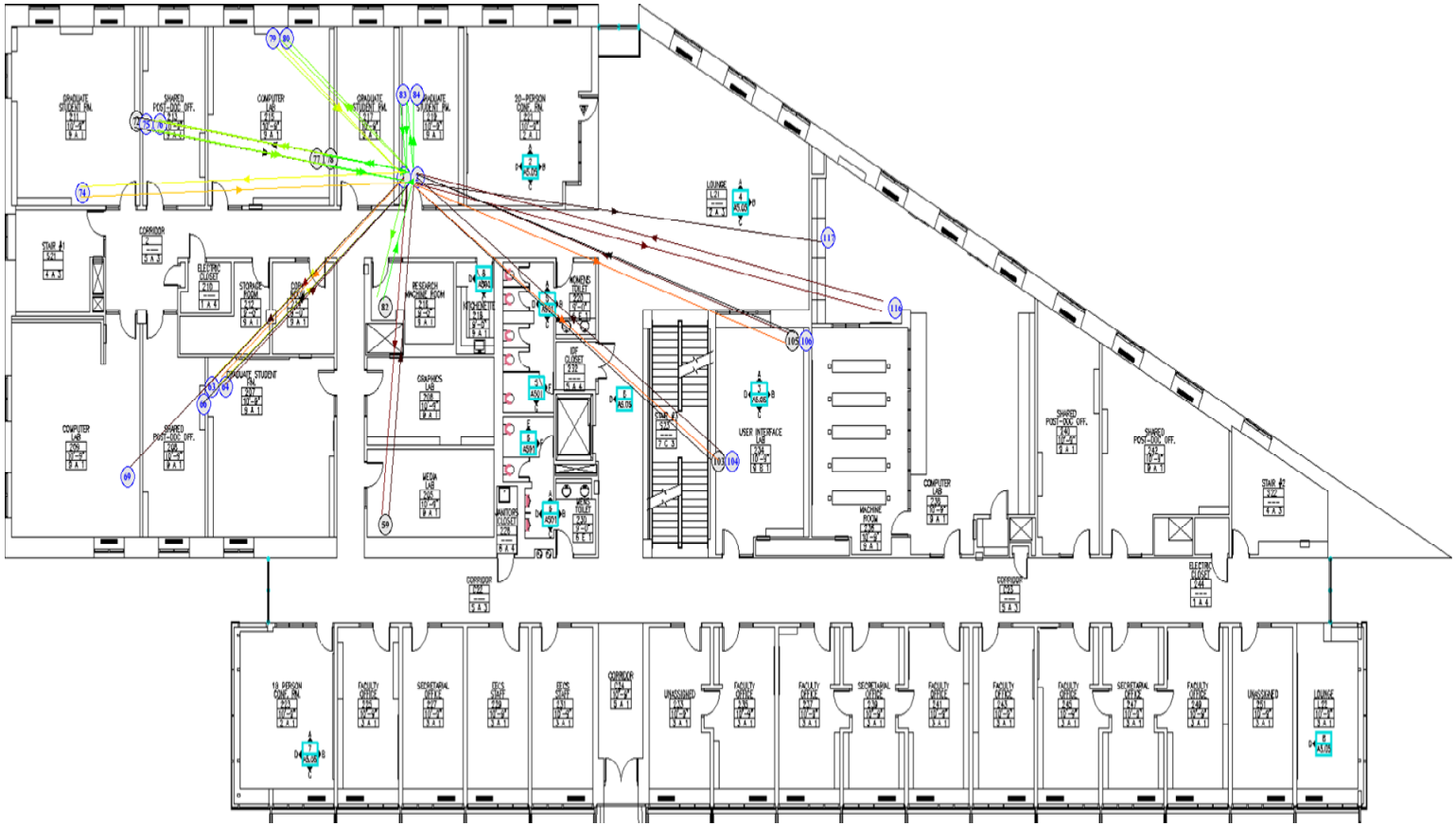


Figure 9 – Example of a Wireless Sensor Network Deployment
Source: Harvard Sensor Network Testbed Website at http://motelab.eecs.harvard.edu/motes-info.php

COMMERCIALLY AVAILABLE DEVICES

Table 1 provides a list of the current communication devices for WSNs. Most of the nodes are programmed using NesC, a variation of the C programming language developed at the University of California at Berkeley. However, sensor nodes with more processing capabilities are being developed and high-level languages such as Java and Microsoft .NET framework have begun to appear as programming interfaces.

Table 1 – Available Wireless Sensor Nodes

NODE	COMPANY	REMARKS
TMote	Sentilla	TI 8MHz CPU, radio is a 250kbps 2.4GHz IEEE 802.15.4 compatible, NesC programmable.
Mica2	Crossbow	ATMEGA128L CPU, with 4K RAM. Radio is 315/433/916 MHz frequency. NesC programmable.
MicaZ	Crossbow	ATMEGA128L CPU, with 4K RAM. Radio is 2.4GHz IEEE 802.15.4 compatible, NesC programmable.
Imote2	Crossbow	Intel PXA271 416MHz processor with MMX capabilities, 32MB RAM, radio is a 250kbps 2.4GHz IEEE 802.15.4. The standard is programmable in NesC, but there is a Microsoft .NET compatible version (Imote2.net)
TelosB	Crossbow	TI 8 MHz CPU, radio is a 250kbps 2.4GHz IEEE 802.15.4 compatible, NesC programmable
IRIS	Crossbow	ATMEGA1281 CPU, with 4K RAM. Radio is 2.4GHz IEEE 802.15.4 compatible, NesC programmable, maximum communication range of 500 mts.
SUN SPOT	Sun Microsystems	180MHz 32 bit ARM920T core - 512K RAM/4M Flash CPU. Radio is 2.4 GHz IEEE 802.15.4 compatible. Java Programmable (Java ME).
Perk	Sentilla	TI MSP430 microcontroller with TI/Chipcon CC2420 low-power wireless radio, Java Programmable (Java ME).

Source: Compiled by Authors from Fact Sheets of a Variety of Sensors

When adding a sensing device to a wireless node, there are many options, depending on the application being developed. Analog and/or digital channels

can be used to attach the sensing devices to the wireless sensor nodes. Analog channels measure a voltage and then convert it (using a table, depending on the analog device) to a number. Examples of these sensors are water level sensors, temperature sensors, etc. Digital channels are read using digital information, utilizing a Universal Asynchronous Receiver/Transmitter (UART), a serial communication channel. Examples of these kinds of devices are GPS receivers and digital cameras. Table 2 shows a sample of available sensors (8).

Table 2- Examples of Sensors

SENSOR NAME	TYPE	COMPANY
Ambient Light Sensor TSL2550	Digital	Texas Advanced Optoelectronic Solutions (TAOS)
Ambient Light Sensor TSL250R	Analog	Texas Advanced Optoelectronic Solutions (TAOS)
Tiny Serial Digital Thermal Sensor TC74	Digital	Microchip
HIH3610 Humidity Sensor	Analog	Honeywell
CO2 engine gas sensor	Digital	AirTest
6004A gas sensor	Digital/Analog	Telaire
AMN1 Product Line Passive Infrared Motion detector sensor	Analog	Matsushita Electronics
Sharp GPD-12 Infrared range finder (distance measurer)	Analog	Sharp
Polaroid 6400 Ultrasonic Ranging Module (distance measurer)	Analog	Polaroid
TruPulse 200, Laser Ranging device (distance measurer)	Digital	Laser Technology

Source – The Sensor Network Museum at http://www.btnode.ethz.ch/Projects/SensorNetworkMuseum

By using a Serial-to-UART circuit or a USB-to-UART, several sensors can be connected to the wireless nodes, allowing the design of applications tailored to customer needs. Other options for connecting sensors exist by using sensor

boards. Crossbow provides several of these sensor boards, especially for the Mica2 wireless node.

Because of the processing, memory, and energy constraints of wireless sensor nodes, multimedia capabilities for wireless sensor networks are limited. For example, given the list of wireless sensor nodes in Table 1, only the Imote2 has enough processing power for connecting a camera directly to the node. For the rest of the nodes, the cameras are connected first to a Digital Signal Processor before they can be connected to the wireless nodes. Although energy spent on processing is important, most of the energy consumed in WSN comes from communication. Therefore, moving multimedia communication along a WSN would easily deplete the batteries of the devices. For example, the size of a typical sensor reading information is about 32 bytes, but a 340 x 280 pixels image is 16*1024 bytes in size, which means that sending a low-resolution image along the wireless sensor network accounts for 512 regular sensor-reading packets. Research by Akyildiz et al. shows a complete survey on multimedia capabilities in WSN including quality of service in the network (9). It was concluded that, depending on the resolution of the cameras, three methods exist to connect them to the nodes, as shown in Table 3. However, most of these capabilities and mechanisms are still in the research stage.

Table 3- Image Devices for Wireless Sensor Nodes

DEVICE	RESOLUTION	SUPPORTED WIRELESS SENSOR NODES
Cyclops Module for CMOS cameras	352×288 pixels, Black and White	Mica2
CMUCam3	352×288 pixels , Color	TelosB
Logitech Quickcam Pro 4000	640 x 480 pixels, Color	Stargate / IMote2

SOFTWARE DEVELOPMENT ENVIRONMENTS

Developing a complete wireless sensor, network-based, intrusion detection system requires writing software in different languages due to the utilization of different classes of devices. Typically, this is done by using a combination of the following three programming languages:

- NesC,
- C/C++, and
- Java.

TinyOS 1.1.10 and NesC

Developed by the University of California at Berkeley, TinyOS is an operating system for running software in wireless sensor nodes (10). The major difference between this operating system and modern operating systems is that at any given time, only one program is allowed to run. Therefore, for multiplexing the utilization of the sensor's processor, TinyOS programs are based on events and small tasks that are executed when an event occurs. Programs for this operating system are developed by writing code in a programming language called NesC, which mixes the writing semantics of C with a Component-based software development paradigm. Component-based development is a technique that allows reutilizing software in an easy manner. TinyOS and NesC are licensed under the open source GNU license.

An application in TinyOS is a set of components that are connected through a single scheduler that multiplexes the hardware resources among the components. Such components are executed when an event occurs. One example is the detection of a movement or the increase of the temperature. This model of execution of a program is known as event-based execution. Because the devices where TinyOS is executed have 4KB of RAM memory, TinyOS does not perform traditional memory management schemes found in a modern operating system for a desktop computer (i.e., virtual memory, pagination, segmentation). Therefore, in TinyOS memory management is static, memory locations are allocated at compilation time, and once the size of the memory of the whole program has been calculated and allocated, it cannot be modified. Figure 10 shows the architecture of an application running on top of TinyOS. In this image, TOS stands for TinyOS, HW for hardware, and motes are the devices where the application is running. An application running on TinyOS is a set of components that are linked through a configuration that states how the components are connected. Each of the components is seen as a black box that provides some set of functionality through interfaces. The component's interfaces make an abstraction of the underlying hardware and they hide the hardware's complexity. For example,

reading a temperature sensor involves the reading of a voltage and the conversion of such voltage to a number. A component that abstracts this process (e.g., a component called thermometer) should provide a method that returns the temperature as an integer in Fahrenheit or Celsius, but not as a voltage.

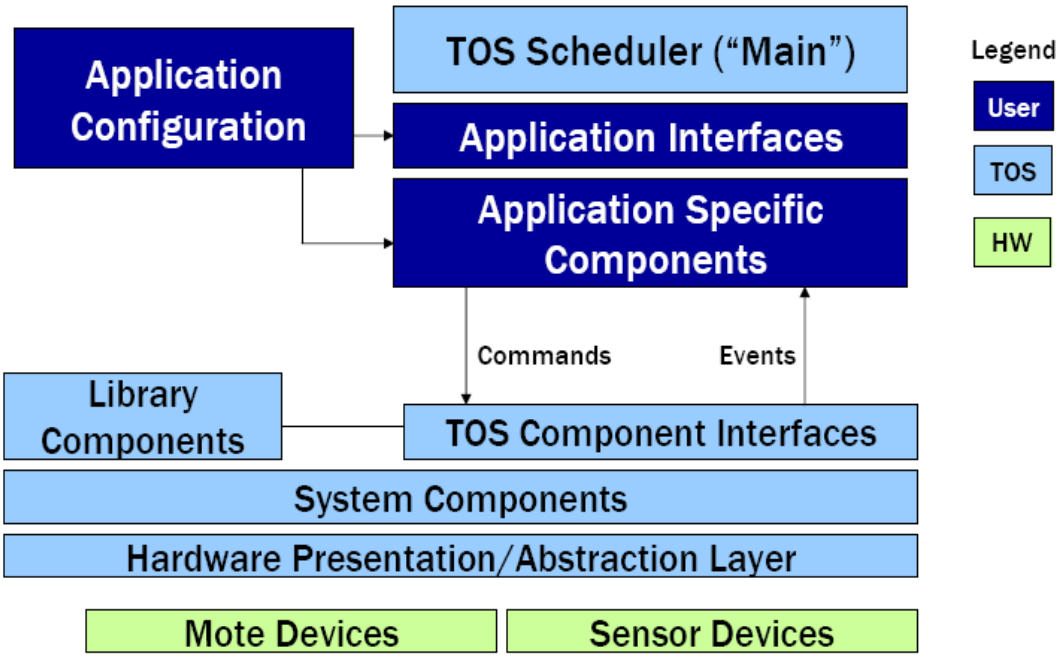


Figure 10 – TinyOS Architecture
Source: WSN kit documentation, TinyOS 101: Basic TinyOS and nesC concepts p. 6

There are three types of functions in TinyOS: Commands, Events, and Tasks.

- A command is a non-blocking function that when invoked, performs some work.
- An event is a function that performs some work when some hardware interruption happens (i.e., detection of movement). It provides an interface that is invoked when the action happens and it can preempt other tasks.
- The task is low priority code that does some computation intensive job.

Since a mote is a simple device with a microcontroller, events and tasks are executed within a time-sharing schedule. How this execution happens is the responsibility of the Scheduler. A two-level scheduler manages the execution of events and tasks. The scheduler is a First-In-First-Out (FIFO) scheduler. Tasks cannot preempt other tasks, but the event functions can preempt tasks, as they have a higher priority in the scheduler.

As mentioned previously, programs for TinyOS are written in a programming language called NesC. The Network Embedded System C (NesC) is a component-based, event driven programming language that has the syntax of the C programming language (the authors of NesC called it a "dialect" of C) but the execution of the program is event based. A program in NesC has two components: modules and configurations. Modules are components that provide and utilize interfaces, developed using a C-like syntax. Configurations are components that put together the modules to provide the wiring of the application.

TinyOS/NesC is open source software maintained by the TinyOS community, and its latest release is the TinyOS version 2.1 (August 2008). The web page of the software community is www.tinyos.net. In the project, NesC and TinyOS are used for the programming of the wireless sensor nodes.

C/C++

C is one of the most widely used programming languages (11). The first version was released in 1972 at Bell Labs by Dennis Ritchie, one of the first developers of the Unix Operating system. C is programmed under a paradigm called procedural, which means that software is developed as orders that act over data

structures. For example, the command Print-F makes the computer write a string to a device, such as a computer monitor. C++ is an extension of C to support a programming paradigm known as Object-Oriented Programming. C/C++ is used to program software for Unix, Linux, and Windows machines.

All Unix and Linux operating system releases include some version of C. C and C++ compilers can be found under GNU open source licenses. The C programming language is standardized by the ISO (International Standard Organization). C/C++ is utilized in the project for programming the base station of the wireless sensor network.

Java

Java is a programming language based on a software paradigm called Object Oriented Programming (OOP) (12). The main idea behind OOP is the easy reutilization of software. Under this paradigm, every functional software unit is abstracted as an object, with methods accessing the internal variables of the object. Developed by Sun Microsystems, Java was first released in 1995. In 2007, Sun Microsystems released Java with the GNU Open Source license.

Java programming language is composed of a series of standard specifications lead primarily by Sun Microsystems, on which industry and academy can participate (12). Each of these specifications addresses a series of programming interfaces for different classes of devices. The major divisions of these specifications are:

Java ME: the Java Micro Edition specifications address programming interfaces for small, portable devices and embedded devices (e.g., cell phones and wireless sensor networks).

Java SE: the Java Standard Edition specifications address programming of desktop computers.

Java EE: the Java Enterprise Edition specifications address the programming of server and distributed applications.

Recently, there has been a move towards using high-level languages like Java and Microsoft.NET to program mote devices. This move is a result of the evolution of technology and the need to minimize the learning curve for programming such devices. Sentilla, with the Perk kit, and Sun Microsystems with the SunSPOT devices, are examples of Java ME compliant devices. On the other hand, Crossbow offers the Imote sensors that can be programmed using Microsoft.NET (12).

For this project, the Java programming language was used for programming the central server where the information from the sensor network arrives and on which a mail/Multimedia Messaging Service (MMS) notification of the intrusion, along with a picture, is sent.

DISCUSSION

WSN technology is quickly evolving. Devices such as the SunSPOTs, the Sentilla Perk, and the Crossbow Imote2 are second-generation state-of-the art devices. The advancement in processor technology (a 900-1.5Ghz processor fabricated in a 45nm process with the size of half of a nail and consuming less than 3 watts), which can now be seen in the Intel Atom processor, will allow the development of more powerful motes in future more highly intelligent and pervasive applications. As a result of this transition, companies are on the production end of the first generation devices causing the reduction of price of the first generation products and deadlines in the end-of-life product cycle.

The utilization of high-level languages such as Java or .NET will displace the utilization of NesC/TinyOS as the default platform for programming WSN. However, this fact depends on the availability of more powerful hardware, since these high-level languages need more resources than NesC/TinyOS.

CHAPTER IV.

WSN-IRNS SYSTEM DESIGN

The WSN-IRNS system developed in this project is a software communication architecture that enables the monitoring, detection, storage, and notification of intrusion alerts. The system is composed of a wireless sensor network using the Crossbow MSP410 motes (12), a base station that collects the information from the WSN, a camera attached to the base station that takes a picture when an intrusion is detected, and a remote server that maintains the database of intrusions and pictures. The remote server is also responsible for the notification of intrusions via e-mail or text message to mobile devices (e.g. cell phones). The notification message may be a simple text message or one that includes as an attachment the picture taken by the camera. Since the base station and the remote server (i.e., notification server) communicate with each other using the TCP/IP protocol, the network that provides the communication between both components can be a local area network or the Internet. For realizing this project, the software was developed in NesC, Java, and C/C++ programming languages. Figure 11 shows the general architecture of the WSN-IRNS system.

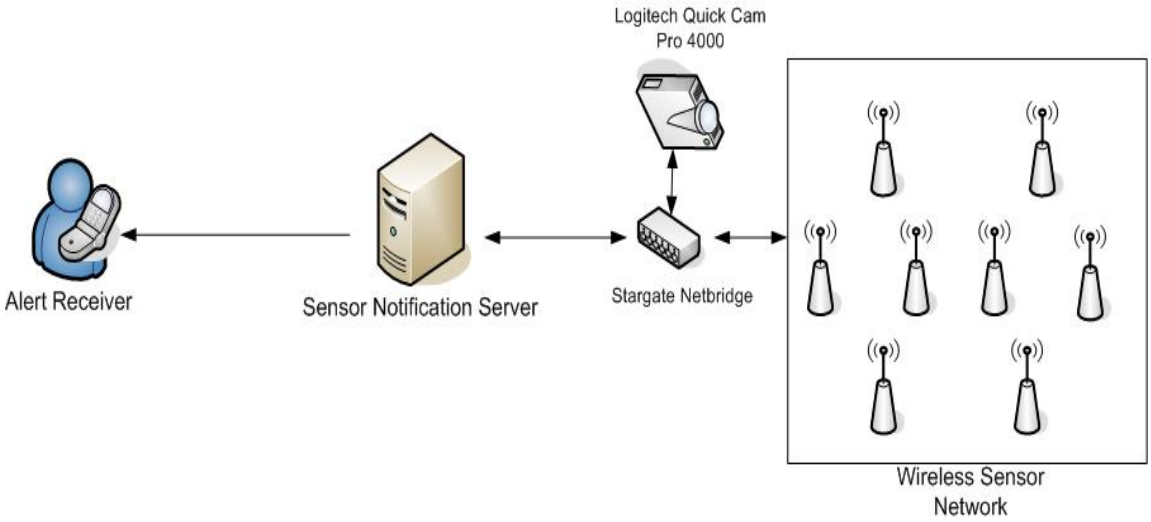


Figure 11 – WSN Intrusion Remote Notification System (WSN-IRNS)

HARDWARE COMPONENTS

The system was developed using the following hardware components:

1 Logitech Quickcam Pro 4000 camera

A USB Camera with up to 1280x960 pixels for image capture is connected via USB to the Stargate base station. It is used for capturing images from the field upon an intrusion notification from the sensors.

1 MIB520 USB Programmer

This device is used for installing/updating the software in the sensor nodes and provides the interface with the motes in the base station.

1 Crossbow Stargate Netbridge

The Stargate (Figure 12) is the base station that serves as an interface between the wireless sensor network and other types of network interfaces, such as Ethernet or 802.11 (Wi-Fi). The Stargate is powered by an Intel IXP420 XScale processor running at 266MHz, with 32MB RAM memory and USB storage. This device utilizes the Linux Debian Operating System, which allows programming in the C/C++ language.

Figure 12 – The Crossbow Stargate Netbridge

1 MSP410 Mote Security Package

This sensor package is composed of one base station interface and eight sensor nodes (Mica2, working at 433Mhz). Each sensor includes both magnetic and passive infrared sensors (PIR). The 2-axis magnetic field sensor detects perturbations in the local magnetic field. The Passive infrared (PIR) sensors are used to detect dynamic changes in the local thermal radiation environment. Each of the MSP410 sensor nodes is equipped with four separate PIR sensors arranged orthogonally, providing full 360-degree coverage (12). Figure 13 shows the MSP 410 wireless sensor node, which is approximately 2" x 2" x 2" (W x H x D).

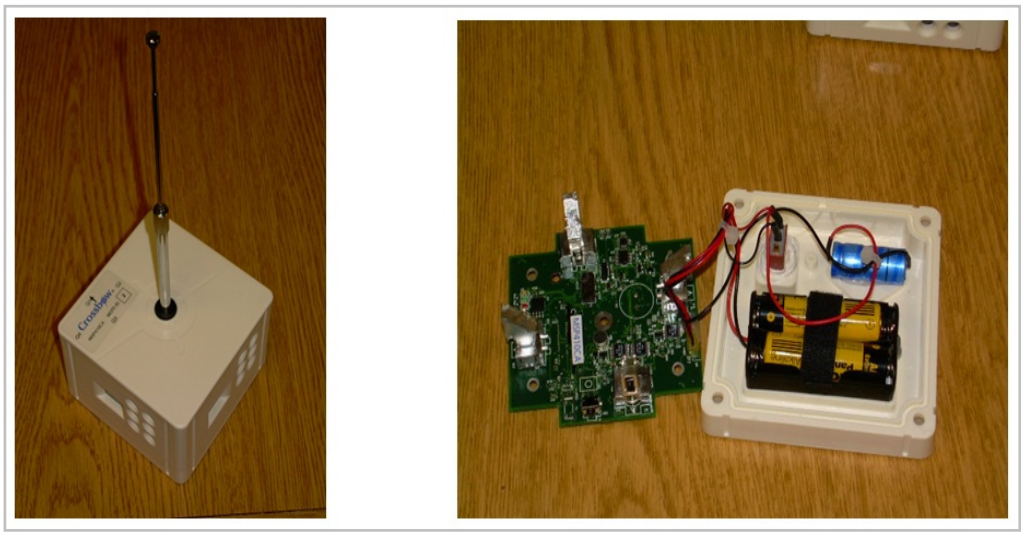


Figure 13 – MSP 410 Motes used in the WSN-IRNS

MSP 410 sensors are deployed around the area to be monitored. The camera and the wireless sensor interface are connected to the Stargate Netbridge, as depicted in Figure 14. There are three USB devices to connect; the pen drive that acts as hard drive, the webcam, and the MIB520. Since the Stargate Netbridge has only two USB ports and three hubs are needed, the Linksys 4-Port Hub was used for the WSN-IRNS.

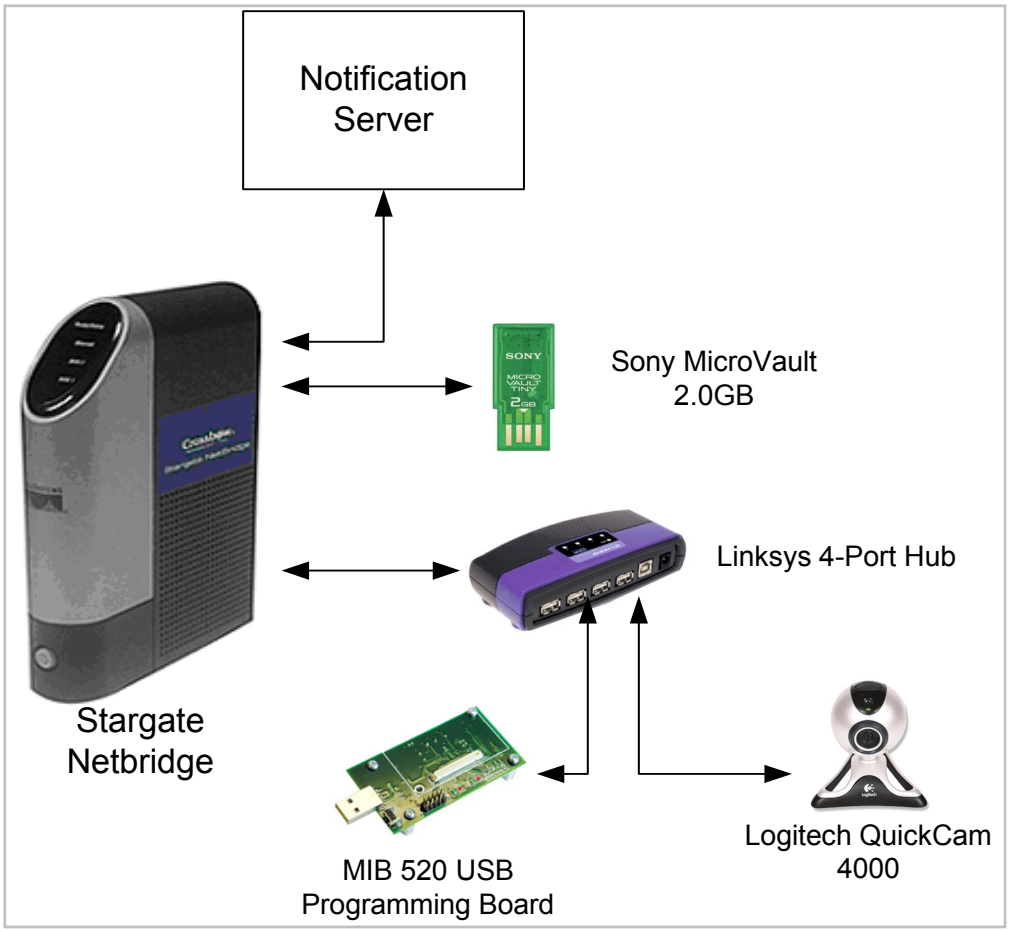


Figure 14 – Interconnection of Devices to the Stargate Netbridge

SOFTWARE COMPONENTS

The development of WSN-IRNS software warranted three different devices: the motes, the base station, and the notification server. The next subsections explain the development of the three software components.

Mote Software Component

The mote software component is programmed in NesC and provides the functionality for detection of the intrusion. Each of the motes runs a small program that notifies the base station upon the detection of the intrusion, and when the battery is nearly out of energy. The program works by reading the values of the accelerometer and the PIRs, and sending a message when the values

of the readings change drastically from the previous sensor readings. The threshold for detection can be changed, but the devices must be reprogrammed.

Since the nodes utilize the XMesh multi-hopping protocol for sending the data back to the Stargate Netbridge, an explanation on how this protocol works is provided in the next section.

XMesh Routing Protocol for Wireless Sensor Networks

WSN are usually deployed in regions with difficult accessibility. Wireless nodes are battery powered so it is important to minimize the energy consumption of the network while performing essential tasks. Efficient routing makes communication possible between all of the nodes and the base station and allows the wireless network to operate for longer periods without human intervention. Developed by Crossbow, the XMesh protocol is an open-architecture, flexible, and powerful wireless routing protocol built on top of the TinyOS operating system.

XMesh (12) is a low-power, multi-hop, routing protocol that offers a solution to routing by creating self-healing, ad-hoc groups of motes nodes that tunnel sensor data to a base node. XMesh packets contain the address of the originating node in addition to the sender and destination. Such a structure allows messages to hop from node to node in a network rather than rely on a point-to-point connection that could rapidly change. If a mote malfunctions for any reason, the remaining motes will automatically re-route around the damaged link, keeping the data flowing along the wireless network. Figure 15 illustrates this concept.

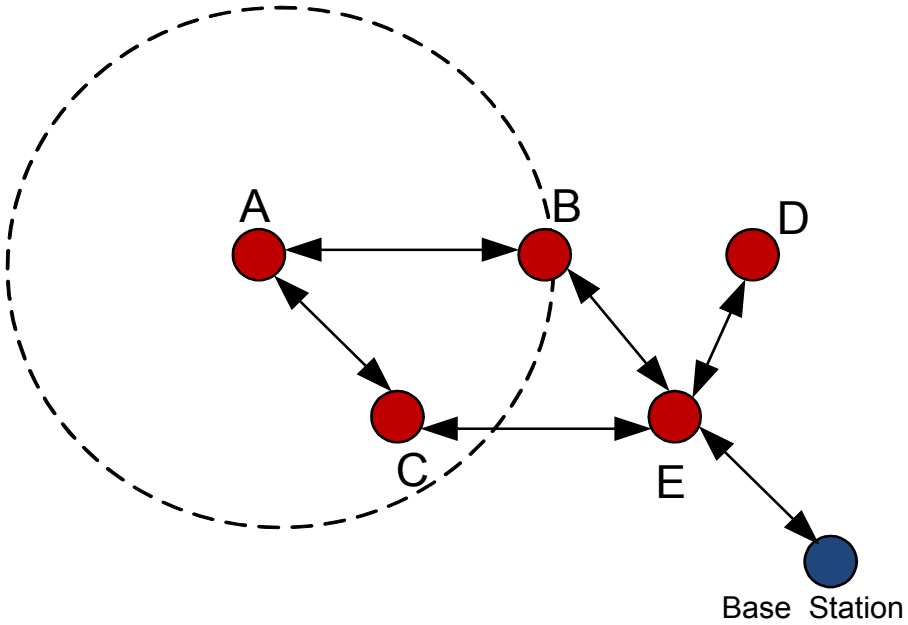


Figure 15 – Wireless Multi-Hop Network

A, B, C, D and E, in Figure 15, are wireless sensor nodes that are communicating with the base station. Since the base station is not inside the communication range of A, any data sent to the base station from A has to be routed using either B or C. This creates two possible paths to reach the base station: (A, B, E) and (A, C, E). Suppose that the routing protocol always use the (A, B, E) path. If for some reason B is not able to work correctly, the routing protocol can switch from B to C, and then use the path (A, C, E) to reach the base station. Since messages pass or jump from node to node, and there are several routing options for data to arrive successfully at the base station, the protocol is multi-hop.

Network Formation

XMesh works in three power modes that are application dependent. The modes are High Power, Low Power, and Extended Low Power. In high power mode, the transceiver (i.e., the radio electronics) of each mote is always on. The average current usage of high power mode is 20-30 milliamps. In low power mode, the transceiver sleeps, but wakes eight times per second to check for messages. The average current usage of low power mode is less than 250 micro-amps. Extended low power mode is used only in certain star topologies in which the end nodes of the network do not route. This means that under the extended low power mode, the network is organized under a hierarchical, tree-based network topology. The

transceiver only wakes up when it transmits messages to its parent. The average current usage of extended low power mode is 50 micro-amps (18).

In XMesh, the mote nodes are pre-assigned unique node ids and a group id for communication. By default, XMesh forms the mesh topology that includes the nodes sharing the same group id, and each node has routing capabilities. On initialization, the primary objectives of an XMesh mote are to determine a parent node and to construct the optimal route upstream to the base node.

XMesh motes calculate network routes by measuring the quality of the radio signal at the receiving side, called the Receive Estimate, to each detected neighbor. First, the sequence numbers for received packets are observed. Then, each mote calculates an exponentially weighted moving average (EWMA) to estimate the respective link cost, similar to TCP/IP's calculation of Round Trip Time over a link. Motes also determine the quality of a transmitted signal (i.e., Send Estimate (SE)) from the route update messages sent by neighbors, as shown in the following equations, where alpha is a value between 0 and 1 that controls the smoothness of the estimate:

*New Estimate = 255*received / (received + missed)*

*Receive Estimate (RE) = (1-alpha) * RE + alpha * New Estimate*

*Link Cost (LC) = (1<<18) / (SE * RE)*

Cost to Base (OC) = LC + Neighbor's Cost

Each mote will next create a neighbor table with up to sixteen entries by default. The base node may have up to forty entries. Additional neighbors exceeding the signal quality of existing entries will replace the lowest quality entry. A parent will be chosen from this table based on two criteria. The parent must have already formally joined the mesh, and the parent must not be a descendent node (i.e., derived from another parent in the last three router update intervals). However, at initialization, a mote will select any node that has already joined the

mesh for its parent to save time. Subsequent full parent selections occur every eight-update intervals.

Router updates occur every thirty-six seconds (times a random number between 0.9 and 1.1) in high power mode, or every 360 seconds (times a random number) in low power mode. The router update messages contain the sending mote's parent id (broadcast if undergoing initialization), its upstream cost to the base node (the base node itself will send zero), the number of hops from it to the base node, and a list of five neighbors. Thus, motes will join the mesh in waves as a function of hop count correlated with router updates. Nodes one hop count from the base station join in one interval and the wave continues in this fashion until eight intervals have elapsed. At this point, the mesh should achieve stability (18). The initialization process will reiterate whenever a mote is damaged or lost.

XMesh message

The structure of an XMesh message is shown in Table 4. This message encapsulates the information sent by a node to the base station. The information read by the magnetometers and the infrared sensors in a mote node is attached to the message in the payload. The fields in italics in Table 4 are the fields being used for the intrusion application. This table corresponds to the values observed from the XMesh protocol while developing the application, since in the original XMesh documentation some fields were inverted.

Table 4 - XMesh Message Structure

Field Meaning	Number of Bytes
Preamble (0x7E)	1
Destination Address	2
Active Message (AM) Type	1
AM Group ID	1
Length	1
Source Address	2
Origin Address	2
Sequence Number	2
Application ID	1
Parent	2
Sensor Board ID	1
Sensor Packet ID	1
Sequence Number	2
Voltage	1
Quad	1
PIR Reading	2
Mag Reading	2
Audio Reading	2
PIR Threshold	2
Mag Threshold	2
Audio Threshold	2
CRC	2
Stop (0x7E)	1

Guaranteeing that the message sent by a node reaches the base station is accomplished by a reliable protocol. Within the Internet, this is done by using TCP, which provides a programmer and user transparent view of the communication process on an end-to-end basis. XMesh can provide reliable communications in the wireless mesh. This is done by providing two

functionalities, one at the link level (on a hop-by-hop basis) and another on end-to-end basis.

The link level reliability is provided by link layer acknowledgements that retransmit lost messages by a sender node if the message is lost or damaged in a transmission to a neighbor node. End-to-end acknowledgements augment link level acknowledgements by sending acknowledgements from the origin node to the base or acknowledgements from the base to the destination node. Enabling end-to-end reliability consumes more energy since every message must be acknowledged by the source or destination node.

Using this multi-hopping protocol, the motes will send only two classes of messages back to the base station. The first one is the intrusion packet, which is sent upon a drastic change in the sensor readings. The second one is the battery packet, sent when the battery reaches any of four predefined levels. Since sending messages is the most expensive operation in a wireless sensor network, the motes have been programmed in this way to minimize the number of messages sent.

Sensors Information Reading Conversion

The PIR values are sent by the motes in bytes 25 and 26. Magnetometer sensor information is sent by the motes in bytes 27 and 28. These values are converted into an integer in the base station by:

PIR_VALUE = 256[byte offset 26] + [byte offset 25]*

MAG_VALUE = 256[byte offset 28] + [byte offset 27]*

For converting the values for the battery readings (byte 23 in the XMesh packet), the following formula is used:

$BV = RV \times ADC\_FS/data$

BV = Battery Voltage.
ADC_FS = 1023.

RV = Voltage Reference for mica2 (1.223 volts).
data = the battery value sampled in the mote and received in the base station.
BV (volts) = 1252.352/data
BV (mv) = 1252352/data

For Mica2, which are the base motes for MSP410, the voltage values vary between 0 and 3V.

Base Station Software Component

The software component of the base station has the functionality of receiving intrusion and battery messages from the motes, taking photos, and sending the messages and the photos to the remote notification server. The software for the base station runs in the Crossbow Stargate Netbridge, a small computer powered by an Intel IXP420 XScale processor running at 266MHz, with 32MB RAM memory and USB storage (19). Crossbow sells these computers running the Debian Linux distribution kernel version 2.6 for ARM processors. The Netbridge is based on the Linksys NSLU2 computer and incorporates the software needed to receive data from the wireless sensor nodes.

The Stargate Netbridge comes with a 2GB USB storage drive that has the Debian Linux Distribution installed. For developing programs for the Stargate, the GNU GCC compiler must be downloaded and installed from the Internet. Then, the Stargate can be programmed using the C programming language, just as a regular computer running Linux. In the previous version of the Stargate, the programs were developed on a normal computer running Linux using a special compiler. Programs were developed, compiled, and then downloaded to the Stargate for execution.

The base station software is composed of four components: the USB port reader, the Photo Capturer, the Remote Communication component, and the Main Control.

USB Port reader: The USB Port reader provides the interaction with the MIB 520 programming board, which is used for receiving intrusion and battery messages. Once the MIB520 is attached to the Stargate Netbridge, it can be read from a computer program running in the Stargate as a device attached to a serial

communication port. The developed USB Port Reader component reads from the serial port, parsing the message and taking from the message the node id, battery level, and magnetometer and PIR readings.

The configuration for reading messages using the MIB520 with Mica 2 (Crossbow sink node for the MSP410) attached to it is as follows:

Serial Port: COM2, in Linux this is found in the file /dev/ttyS1
Baud Rate: 57600 bps
Buffer Size: 36 Characters (Default TinyOS message size)

The USB Port Reader waits until it reads a TinyOS packet.

Photo Capturer: The photo capturer provides the functionality of capturing images using the webcam. This functionality is provided using an open source utility called *streamer*. The *streamer* utility, which is downloaded from the Debian repositories, allows taking pictures in different formats and resolutions. Streamer was utilized because it was compatible with the webcam and it is invoked from the Main control using a Linux system call.

Remote Communication Component: This component provides the communication between the base station and the remote notification server. This component sends the parsed intrusion/battery level messages and the pictures back to the notification server using the TCP protocol.

Main Control: This component utilizes the three previous components. When the program initializes, it calls the USB Port reader and waits until a TinyOS intrusion message is received. Upon reception of the packet, the USB Port reader parses it and places it in a data structure. After this, the Main Control invokes the Photo Capturer if the received packet from the Port Reader is an intrusion message. Once the photo has been taken, the Main Control invokes the Remote Communication Component to send the parsed message to the remote notification server. In case the message is an intrusion, the Main Control also sends the photo to the notification server.

Notification Server Component

The Notification Server component contains the database and the notification functionality of the alerts and photos taken in the wireless sensor network. The software used for developing and running the Notification Component is:

- Java Enterprise Edition as the programming language.
- Sun Java Application Server version 9.1 for the Notification Server Component.
- Postgres SQL version 8.2 as the database system.

The notification process begins when the intrusion/battery message is sent from the Stargate Netbridge to the Notification Server (Figure 16). Depending on the nature of the message, two things can happen. If the message is an intrusion alert, the Notification Server saves the information of the notification and responds back to the Stargate Netbridge with a Ready message, which then allows the Stargate to transfer the photo back to the Notification Server. Once the photo is received, the notification is sent along with the photo, using the Simple Mail Transfer Protocol (SMTP) to the recipients stored in the database. If the message sent by the Stargate is a battery alert, the Notification Server saves this information and sends a notification to the recipients using SMTP. The notification messages sent by the Notification Server are e-mails/MMS. Figure 16 illustrates the process of sending an intrusion notification (upper image) or a battery notification (lower image).

The Notification Server must be connected to the Internet, so that it can send the notifications back to the interested personnel. The base station (Stargate Netbridge) should be able to connect to the Notification Server, which means that such server should be in the same local area network of the Stargate Netbridge, or the Stargate Netbridge should be able to connect to the Internet.

The total time from when the intrusion is detected until the security personnel receive the notification message on a mobile phone can vary depending on certain conditions. Experiments with the prototype WSN-IRNS system produced an end-to-end delay of approximately 40 seconds on average. The components of the system that introduce portions of this time delay include the sensor-to-base station link, the base station to notification server link, and the propagation of the

final notification message from the notification server to the mobile device. First, the message has to be detected and sent from the sensor to the base station. Given that the Medium Access Control protocol is contention-based, the time it takes to acquire each link toward the base station will depend on the amount of traffic in the network. However, in this application a high load with many collisions is not expected. The other time component from the sensors to the base station is the propagation delay, which is a function of the number of links (i.e. size of the network) and their distances. These are deterministic values, and for this application this portion of time delay should be in the order of milliseconds. Other delay components occur between the base station and the notification server. Upon receiving the intrusion notification, the base station captures and stores the picture, assembles the message, and sends the message to the final destination. The time required by this process can vary because it depends on the load on the machines and the transmission through another contention-based network to the notification server. In this application, this delay is also expected to be a small value, in the order of hundreds of milliseconds. Finally, the notification server sends the text/MMS message to the security personnel. The message is sent using the Internet and the cellular network to which the mobile device is subscribed. This part of the end-to-end delay can also vary and depends on the load of the Internet and the cellular network. The other causes of time delay in the system are negligible compared with the potential delay caused by highly dynamic Internet and cellular network conditions. As mentioned above, numerous experiments have determined the entire end-to-end delay for the current WSN-IRNS prototype to be in the order of 40 seconds on average.

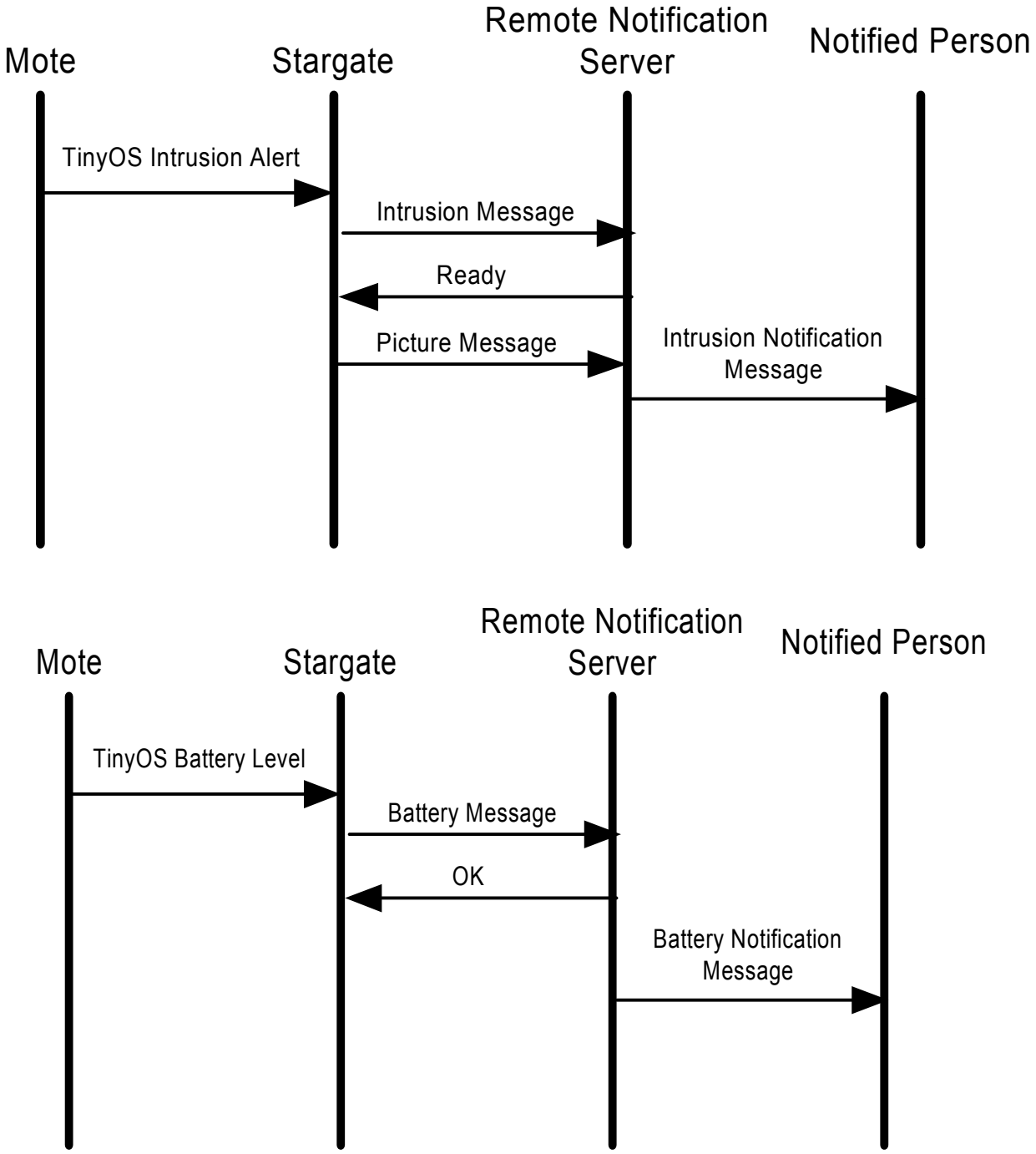


Figure 16 – Intrusion and Battery Notification Sequences

CHAPTER V.

FIELD TESTS

Sensor Tests

The objective of these experiments was to determine the maximum and effective sensing distances for the MSP410 wireless sensor motes. Sensing distance directly affects the density of any deployed perimeter in which an intruder or vehicle must be detected as they pass between any two adjacent nodes.

The MSP410 mote has four orthogonally oriented passive infrared (PIR) sensors. The horizontal field view of each of the four quadrants is composed of nine beams. Detection occurs when a warm object crosses two neighboring beams. The sensor output is the difference between the amounts of infrared radiation striking the two beams. The PIR reading of the MSP410 detection packet is defined as the magnitude over the PIR detection threshold, on a scale from 0 to 1023. The implemented detection algorithm uses a threshold PIR_MAX set to 600.

The MSP410 also has a two-axis linear magnetometer. The magnetometer reading of the MSP410 detection packet is defined as the magnitude over the magnetometer detection threshold. This magnetometer threshold for the implemented detection algorithm sets the threshold at 500.

The first experiment was done to measure the detection range of the PIR sensors. Initially a sensor was placed 30 meters away from the base station. At such distance, there was no communication between the mote and the base station due to the smaller gain of the base station's whip antenna. The problem was solved by placing an intermediate mote 15 meters between the base station and the initial mote. After data was being received in the base station, a test subject began walking until the maximum PIR threshold value was received in the base station. Figure 17 shows the result of the experiments. The data that corresponds to this figure is shown in Table 5.

Table 5 - Passive Infrared Experiment

Distance (meters)	PIR Reading
0	1023
7.62	1023
9.144	969
12.192	840
15.24	711
18.288	621

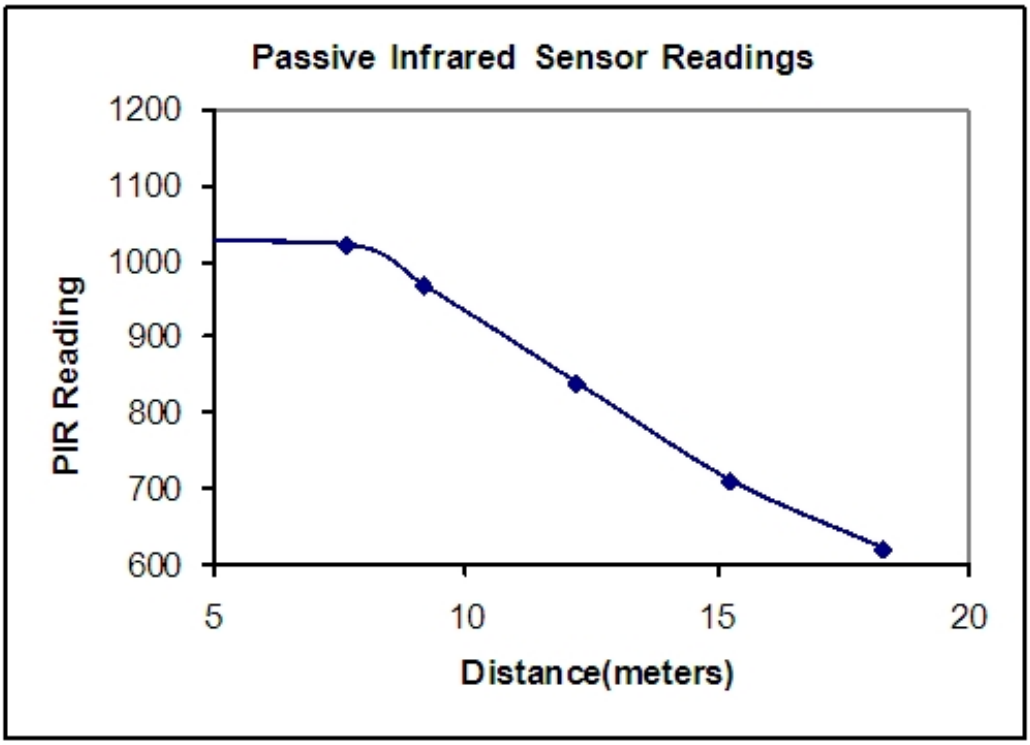


Figure 17 – Passive Infrared Readings as Function of the Distance

Similar experiments were performed with the magnetometer. In this case, the experiment was to take two sensors and place them at either side of a well-transited street. Since these magnetometers work in pairs, the idea of the

experiment was to increase the separation distance among the motes and see the resulting readings. Figure 18 shows the results of these experiments and Table 6 shows the values.

Table 6 – Magnetic Sensor Experiment

Distance (meters)	Mag Reading
0	12563
7.62	12563
9.144	1032
12.192	878
15.24	663
18.288	505

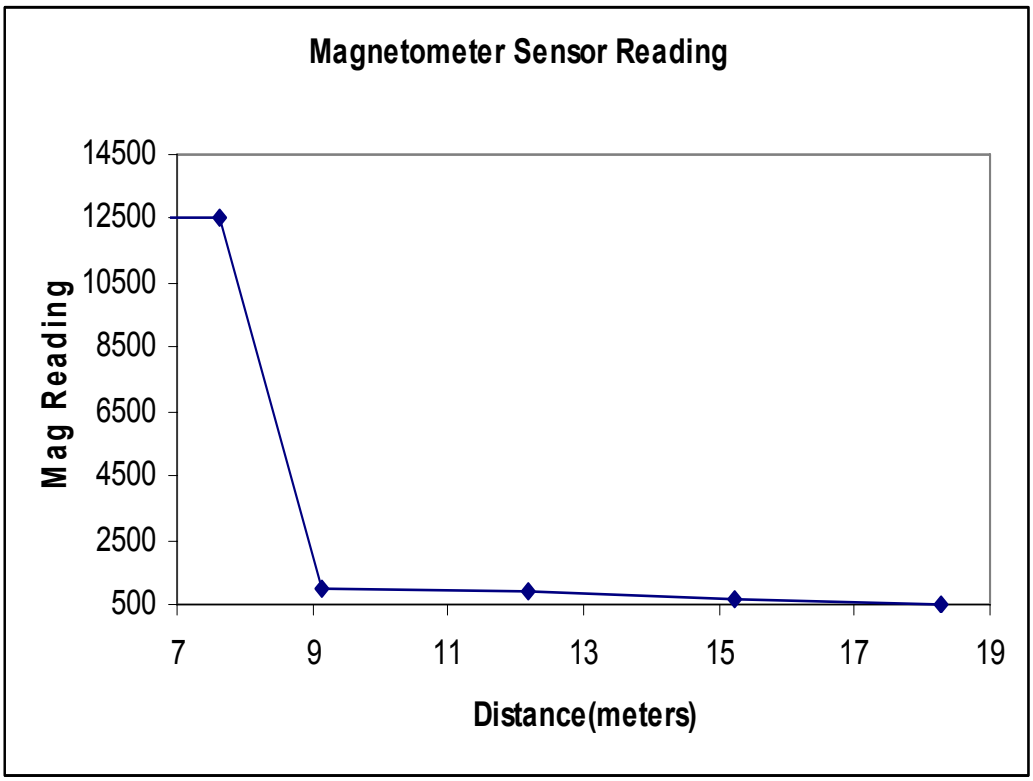


Figure 18 – Magnetometer Readings as Function of Distance Separation

The results of these experiments concluded that to have good readings and detect intrusions by means of the PIR readings (persons) or the magnetometers (vehicles); the sensors cannot be separated by more than 18 meters.

Preliminary System Testing

To test the functionality of the WSN-IRNS prototype, the WSN was set up inside the CUTR building at USF. It checked the intrusion notifications as people passed through the system. A research team member's email address was used as the notification recipient. An example of the notification message received when a mote was sent is shown in Figure 19.

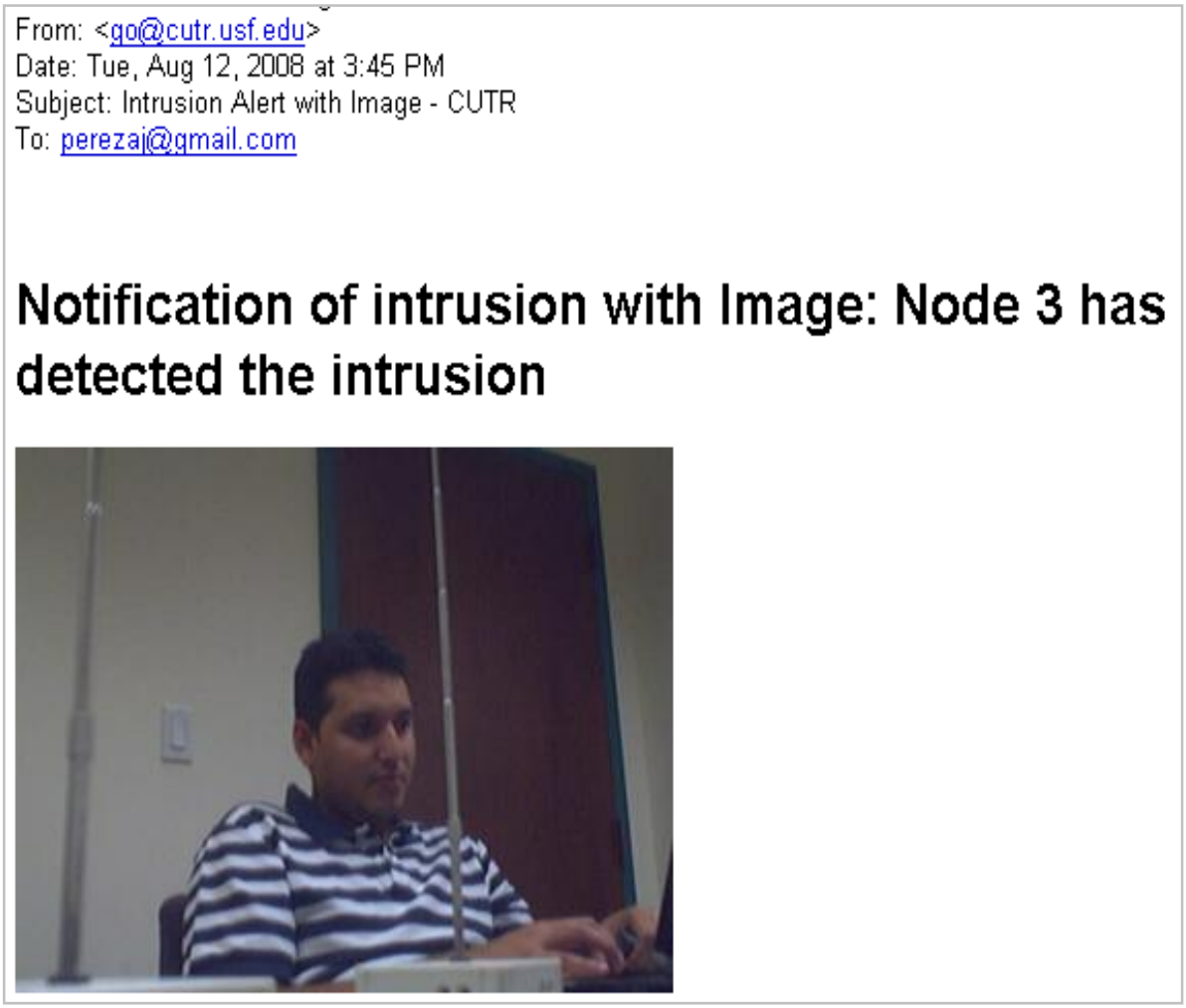


Figure 19 – Example of an Intrusion Message

PROTOTYPE SYSTEM DEPLOYMENT

Starting the System

For WSN-IRNS to be deployed, several components must be set up. The first step is to connect the base station to the Internet since the Notification Server is located in the USF – CUTR computer network.

Initializing the system involves initializing the software in the base station and the Remote Notification Server. The Linux operating system starts the base station and initializes the software there. The steps for setting up the base station are as follows:

1. Connect the base station to a Local Area Network that utilizes DHCP.
2. Turn on the base station and get the base station's IP address.
3. Connect using SSH to the base station.
4. Turn on the wireless sensor nodes and wait 8 – 10 minutes for the wireless sensor network to set up its routing tables.
5. Start the base station software by entering in the SSH command line "./basestation.out –p /dev/ttySS1."
6. The software in the Remote Notification Server is initialized when the operating system of the server boots. As that server is always running, the person that deploys the system needs no interaction with that component.

The system was also tested outside the Pinellas Suncoast Transit Authority (PSTA) facility in St. Petersburg, Florida, the transit agency identified to test the application. These tests, which were performed in an open area outside PSTA's main facility, demonstrated the need for system tuning and customization according to the specific area of deployment. Different environment conditions at deployment sites can affect the detection thresholds that trigger the notifications, thereby causing false-positive and/or false-negative alerts. For example, in the tests performed at PSTA one of the sensors was placed near a metal pipe that caused continuous false-positive alerts triggered by an overly sensitive magnetometer.

The thresholds were later adjusted and successfully tested in an outside parking lot at USF that was similar in traits to the PSTA test location. Several rounds of

threshold value modifications took place in order to enable the system to reliably report vehicle and person intrusion notifications. In the final configuration, the infrared sensors were used to detect motion and trigger the alert notification process, and then the magnetometers were used after motion detection to determine when the motion was caused by a person or vehicle. This trial-and-error process may need to be repeated at each installation site in order to account for environmental obstacles and conditions unique to each location.

During the tests at PSTA, the agency identified the main gate of their parking facility as a potential application for a modified version of WSN-IRNS. PSTA reported the need of a security system that would check for person intrusions in the garage facility's main gate, because during peak hours there are usually problems with the mechanical gate due to the frequency of arriving vehicles. The desired system would report only person intrusions through the main gate while allowing buses to pass freely through the gate without alarm.

PSTA reported that gate intrusion issue is a common problem in most big parking facilities, and they believe those facilities might also be interested in this system. Given the importance and need of the proposed modification, the research team will investigate the modification of WSN-IRNS system to fit these needs in future efforts and demonstrate the new system at PSTA outside of this project. These modifications will include changing the alert-triggering logic and sensor detection thresholds in the WSN software so that an alert is triggered only when the infrared sensor detects movement and the magnetometer does not detect the presence of a vehicle. This alternate configuration differs from the current system where an alert is always issued upon the detection of movement from the infrared sensor, despite whether the intrusion is caused by a vehicle or human.

COSTS OF A COMPLETE DEPLOYMENT

The total cost of deployment is directly related to the cost of the hardware and software associated with the system. The cost of the hardware is directly related to the size of the facility being monitored. A larger facility will require an increased number of devices, thereby increasing costs. An additional cost related to customization of the hardware components is determined by the time spent to install and test the different software components. Approximate prices of WSN-IRNS system components are detailed in Table 7.

Table 7 – Approximate Costs of WSN-IRNS Prototype Components

system components	Costs
Wireless sensor network intrusion detection kit	$5,000.00
Stargate Netbridge	$500.00
Notification server can be as low as the cost of any server.	An available server was used
Software: Licenses of software components were free.	Free
Web camera: Depends on the quality of the camera.	$50.00
USB memory thumb drive	$50.00
USB hub	$50.00
MIB 520 Programming board	$100.00
Total cost of new equipment used to create the system (using an existing computer as the server)	$5750.00

CHAPTER VI.

CONCLUSIONS

This report describes a low-cost WSN-based system that monitors and notifies of intrusions in remote and unattended facilities of a given transit agency such as a parking garage or yard. The WSN-IRNS prototype is equipped with infrared sensors and magnetometers to detect person and vehicle intrusions. Further, upon the detection of an intrusion, the system takes a picture and sends a notification to the appropriate designated security personnel via a multimedia message to a cellular phone or an email to a regular email account.

WSN-IRNS was developed using off-the-shelf commercially available hardware. The prototype was tested and evaluated on the USF campus as well as outside the PSTA facility in St. Petersburg, FL. These tests demonstrated the need for system tuning and customization according to the place of installation. Every environment has different physical characteristics that affect the propagation of the signals and sensitivity of the sensors, which can potentially cause the system to report an excessive amount of false-negatives or false-positives. Therefore, in-situ tuning is of utmost importance to guarantee appropriate performance of the system. For example, during initial field tests at PSTA a nearby metal pipe caused the system to issue multiple false positive alerts as a result of the readings from the magnetometer. After the field tests at PSTA, further threshold tuning was performed in a similar environment at USF and the alert logic was adjusted to only issue notifications when the infrared sensor detected motion, which yielded much better performance with far fewer false positive alerts.

During the life of this project, the WSN intrusion detection kit utilized in this research was discontinued in anticipation of the commercial release of updated technology. Therefore, it is imperative to build a similar system using individual parts available in the market. New sensors need to be found and connected to regular off-the-shelf motes. Although the connection of new sensors to the motes will imply software modifications, these are expected to be minor in scope. The availability of new sensors will require additional investigation efforts.

During testing at PSTA's facilities, the transit agency identified the bus garage main gate as an area where a WSN system such as WSN-IRNS could bring benefits to many transit agencies. Future work, outside of this project contract, will focus on fine-tuning this system to identify person intrusion detection at the agency's bus garage main gate while allowing buses to pass through without triggering alarms. These modifications will include altering the alert-triggering logic and sensors thresholds in the WSN software appropriately so that an alert is triggered only when the infrared sensor detects movement and the magnetometer does not detect a vehicle, instead of the current configuration which always issues an alert when the infrared sensor detects movement. This modified system will allow the facility to increase the efficiency of operations and avoid equipment malfunctions by leaving the physical gate open for arriving and departing buses while protecting the property from intrusion by an individual on foot.

For extremely time-sensitive applications, the end-to-end time delay of approximately 40 seconds observed in prototype testing between sensing an intrusion and the receipt of the notification at the mobile phone could potentially be reduced by relying on IP-based communication between the application server and the mobile phone instead of messaging protocols such as SMTP and MMS. These modifications are likely to significantly reduce the largest time delay in the system. However, this alternate architecture would require a mobile WSN-IRNS software application to be developed and installed on the mobile phone as well as more sophisticated WSN-IRNS application server software that would communicate directly with the mobile phone software. Possible performance tradeoffs resulting from this different system architecture would also have to be examined. This work could be accomplished in future research projects.

REFERENCES

(1) Transit Cooperative Research Program, *Intrusion Detection for Public Transportation Facilities Handbook*, TCRP Report 86, Transportation Security, Volume 4, June 2003.

(2) Federal Railroad Administration, State-of-the-Art Technologies for Intrusion and Obstacle Detection for Railroad Operations, February 2007

(3) Florida Department of Transportation, *Aware Pilot Project along South Florida Rail Corridor,* Final Report 2002

(4) Streetline Incorporated, *Streetline Systems and Service and Applications Datasheet*. http://www.streetlinenetworks.com/site/solutions_parking-management.html (accessed October 2008).

(5) Honeywell-Vulcain Inc., "Vulcain 301W Wireless Gas Detector." http://www.vulcaininc.com/(iz3x40fs3j5ksya31dusy045)/product_item.aspx?item=77&langtype=1033 (accessed October 2008).

(6) Lee, S. D. Yoon, and A. Ghosh, "Intelligent parking lot application using wireless sensor networks," International Symposium on Collaborative Technologies and Systems, CTS 2008, Irvine, California, 19 – 23 May 2008.

(7) Harvard University, *"Harvard Sensor Network Testbed: The Mote Lab Portal."* http://motelab.eecs.harvard.edu/ (accessed September 2008).

(8) ETH Zurich, *"*The Wireless Sensor Museu*m."* http://www.snm.ethz.ch/Main/HomePage (accessed September 2008).

(9) Akyildiz, I.F., T. Melodia, and K.R. Chowdhury, "A Survey on Wireless Multimedia Sensor Networks," Journal Computer Networks, Elsevier Publishing, Vol. 51, No. 4, pp. 951-960, March 2007.

(10) TinyOS Community, "TinyOS Community Forum." http://www.tinyos.net/ (accessed September 2008).

(11) GNU, "GNU GCC Compiler." http://gcc.gnu.org/ (accessed September 2008).

(12) Sun Microsystems, "Sun Java Main Web Page." http://java.sun.com/ (accessed September 2008).

(13) Java Community, "Java Community Process." http://jcp.org/ (accessed September 2008).

(14) Crossbow Corporation, "Crossbow Product Catalog 2007." http://www.xbow.com/Products/wCatalogs.aspx (accessed December 2007).

(15) Crossbow Corporation, "MSP410 Mote Security System Manual," 2005.

(16) Crossbow Corporation, "XMesh User's Manual Revision D," April 2007.

(17) Crossbow Technologies, "Stargate Netbridge User's Manual Revision A," October 2007.

(18) Crossbow Corporation, “XMesh User’s Manual Revision D,” April 2007.

(19) Crossbow Technologies, “Stargate Netbridge User’s Manual Revision A,” October 2007.