Fire Detection Using Wireless Sensor Networks

by

Shadi Al-Khateeb

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master

in the

Electrical Engineering

Program

YOUNGSTOWN STATE UNIVERSITY

June, 2014

ETD Program

Digitally signed by ETD Program
DN: cn=ETD Program, o=Department of Graduate Studies and Research, ou=Youngstown State University, email=etdadmin@ysu.edu, c=US
Date: 2014.09.15 10:43:06 -04'00'

Fire Detection Using Wireless Sensor Networks

Shadi Al-Khateeb

I hereby release this thesis to the public. I understand that this thesis will be made available from the OhioLINK ETD Center and the Maag Library Circulation Desk for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

Shadi Al-Khateeb, Student Date

Approvals:

Dr. Frank X. Li, Thesis Advisor Date

Dr. Jalal Jalali, Committee Member Date

Dr. Faramarz Mossayebi, Committee Member Date

Dr. Salvatore A. Sanders, Associate Dean of Graduate Studies Date

Abstract

Fire threatens man, animals, plants and properties. So there should be a very adequate, fast, and easy to install alarm systems to avoid the fire consequences. A wireless sensor network (WSN) with multiple wireless sensor nodes and a base station for fire detection is presented. It is a low-cost, easy to construct, highly efficient, fast, and reliable wireless sensor network. Wireless nodes measure the temperature and light intensity in their surrounding areas. The temperature and light data from the wireless nodes are used to detect the fire event and then the system sounds an alarm. The wireless sensor nodes collect and send data continuously to the base station, which is connected to a computer. Software application has been developed in order to conduct the data mining of the wireless sensor network. The WSN has shown the efficiency of mining real time data, reliability, and its flexibility for expansion.

Acknowledgments

I would like to thank the faculty of the department of Electrical and Computer Engineering at Youngstown State University and my advisory committee. My special thanks to Dr. Frank X. Li for supporting and guiding me through the entire project. I would also like to thank my family, especially my wife.

Table of Contents

List of Figures

List of Tables

Abbreviations

ARPANET	Advanced Research Projects Agency Network
CMU	Carnegie Mellon University
DARPA	Defense Advanced Research Projects Agency
DSN	Distributed Sensor Network
FD	Flame Detection
HD	Heat Detection
I/O	Input/Output
ISM	Industrial, Scientific and Medical
LQRP	Link Quality Routing Protocol
LR-WPAN	Low Rate Wireless Personal Area network
MIT	Massachusetts Institute of Technology
OGM	Open Geospatial Consortium
OS	Operating System
OTA	Over The Air
RFID	Radio Frequency Identification
SPOT	Sun Small Programmable Object Technology
VID	Video Image Detection
VM	Virtual Machine
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Chapter I

Introduction

Over the last decade, the technology of Wireless Sensor Networks (WSNs) has been widely used in many real time applications. These small sensors can sense, process and communicate. Most wireless sensor nodes are capable of measuring temperature, acceleration, light, humidity, level of gases, and chemical materials in the surrounding environment.

Fires can happen anywhere and at any time, Wired sensors could not be implemented everywhere easily, especially in remote areas; also, it is hard and very expensive to install and maintain the wired sensors. Additionally, if a sensor wire breaks, the communication will be lost completely. The WSN with a mesh network could be easily installed and maintained, and the network would be still working even if one of the wireless sensor nodes is damaged.

The amount of data collected from WSNs is huge in a real life situation. Therefore, mining this data efficiently is very important. In this thesis the wireless sensor network data mining is discussed. An application has been developed and tested in order to study the data mining to detect fires using WSNs.

1.1 Background

Wireless sensor networks have been used in many real life applications, such as, quality monitoring, habitat monitoring, entertainment applications, automatic applications, tracking applications and environment monitoring. These scattered wireless sensors must be organized in a certain network topology to overcome the communication problem, the transmission and reception of data between these nodes have to be controlled by using routing protocol and certain routing algorithms. The possible failures of wireless sensors have to be handled, as well as the power management of the wireless sensors is very important. All of these issues and research challenges have been widely studied by the research community [1].

As fire is very dangerous we must develop an efficient, robust, and reliable detecting system. Wireless sensors are low cost, easy to install, low power, communicating using radio waves. They can be installed anywhere, even in the forests as there is no need for wires. As a fire can happen anytime, it is necessary to have a real time detection system, and this can be implemented using WSNs and a software application that analyzes the huge amount of real time data coming from the scattered wireless sensor nodes.

1.2 History of WSNs

The research on WSNs started as a distributed sensor networks (DSN) program at the Defense Advanced Research Projects Agency (DARPA) around 1980. By that time there were about 200 hosts at universities and research institutes working together with the Advanced Research Projects Agency Network (ARPANET) [2]. it was assumed that

DSNs have many spatially distributed low-cost sensing nodes that collaborate with each other but operate autonomously, in this case the information used to be routed to the most appropriate node that needs this kind of information . At that time, this was actually an ambitious program. The processing was mainly done using computers. In 1978, the Distributed Sensor Nets workshop (Proceedings of the Distributed Sensor Nets Workshop, Department of Computer Science, Carnegie Mellon University, Pittsburgh PA) identified the technology components for a DSN, which include: sensors (acoustic), communication and processing modules, and distributed software.

After the creation of the communication-oriented operating system (Accent) by the Carnegie Mellon University researchers, the access of the distributed resources that are required for a fault-tolerant DSN became clear and flexible [3]. To have a demonstrative application of DSN, the Massachusetts Institute of Technology (MIT) developed a helicopter tracking system using a distributed array of acoustic microphones [4]. As the sensors were in their infancy in the first development stage, the size of the sensors was large, which limited the number of applications. Additionally, DSNs were not tightly associated with wireless connectivity. Many new civilian applications of sensor networks such as environment monitoring, vehicular sensor network, and body sensor network emerged as a result of the advances in the network research, networking techniques and networked information processing by. DARPA was the active researcher in the new wave of sensor network research area by launching an initiative research program called SensIT, which added new capabilities to the present sensor networks. Examples of these capabilities include: ad hoc networking, dynamic querying and tasking, reprogramming and multitasking [5]. Furthermore, the IEEE has defined the

IEEE 802.15.4 standard for low data rate wireless personal area networks [6]. Based on IEEE 802.15.4, ZigBee Alliance published the ZigBee standard, which specifies a suite of high level communication protocols that can be used by WSNs [7]. Currently, WSN has been viewed as one of the most important technologies for the 21st century [8].

1.3 Sensor networks in fire detection

"Fire or combustion is a sequence of exothermic chemical reactions between a fuel and an oxidant accompanied by the by-products of combustion being: heat, smoke & electromagnetic radiation (light)'' [9]. From the definition it is clear that fire detection can be easily done by sensing the fire outcomes, such as, heat, smoke, and light.

1.4 Fire Detection Techniques:

There are many techniques that can be used to detect fire; each one has its own advantages and disadvantages. Some of them are suitable for warehouses while others are suitable for large buildings [10].

1.4.1 Smoke Detection:

Smoke detection is the most common type of fire detection that is used in warehouses. Smoke detection is provided in the form of spot-type smoke detectors that are usually fixed on the ceiling or walls. As the smoke is produced during the earliest stages of fire development, smoke detection can provide early alarm of fire in small or low-ceilinged buildings. The smoke from a small fire may not be strong enough to activate smoke detectors that are attached to the ceiling in very large facilities with high

ceilings, because the smoke will not be concentrated at the early stage of the fire. Additionally, this type of detectors on small or large building needs to be cleaned periodically to prevent false alarms.

Video image detection (VID) systems use security cameras to detect smoke and flames. The images captured by these cameras are processed to determine whether smoke or flames can be identified. Using VID in large buildings is more efficient as the VID systems can cover a larger volume of spaces than the spot and beam detections. However, in warehouses, the large number of barriers like walls can create obstructions in coverage. For each small room or entrance, a camera is needed. In addition, VID systems require adequate light to detect smoke.

1.4.2 Heat Detection:

Heat detection (HD) can be used in warehouses. As the heat detectors require a high level of heat to activate, this method will not provide an early alarm as in the smoke method. Heat detection is done by measuring the temperature, or temperature rate-of-rise, and activating when a prescribed threshold is reached. If the temperature threshold is set low, the alarm may be triggered earlier. Therefore, the percentage of false alarms could be high. On the other hand, if the temperature threshold is set too high, the alarm will be too late in the case of fire. Heat detection can be provided in addition to spot type detectors mounted on the ceiling or in storage racks. An example of this detection is the automatic sprinkler systems provided with water flow. Another example is a linear heat detection that can be accomplished through a cable or tube-based detector that measures temperature along its length. Linear heat detection devices, which can be mounted and

run in storage racks at different levels, typically require low-maintenance, and are resistant to adverse environmental conditions.

1.4.3 Flame Detection:

Flame detection (FD) uses optical detectors to measure the radiant energy emitted by flames. The devices can be adapted to detect specific types of wavelengths indicative of fire. This type of detection reduces inconvenient false alarms, because it detects the wavelengths corresponding to only the fire's wavelength. This type of detection is able to cover a large area, but the devices must have a direct view of a fire, which means that barriers in the space can be an issue. This type of detectors requires that the sensor devices be cleaned periodically.

1.5 Organization

This thesis is divided into five chapters. Chapter 2 is the introduction of The Sun Small Programmable Object Technology (SPOT), SPOT hardware and software components, and applications. Chapter 3 provides a detailed description of wireless sensor networks (WSNs), layers architecture, radio communication, network topologies, constraints in designing WSNs, thesis project, and routing protocol. Chapter 4 provides the fire detection algorithm, the experimental procedure and results. Chapter 5 is the future work and conclusion. Appendix A contains the java code for free-range SPOT and the host application.

Chapter 2

The Sun Small Programmable Object Technology (SPOT)

2.1 Sun Spots Project

The SPOTs project was initiated in Sun Labs, the research arm of Sun Microsystems. For now, it includes contributions from a community of researchers, educators, students and hobbyists. This project continues to add more software programs and applications to have a comprehensive, Java-based software platform for these devices. It aims to enhance the use of Java, which means that the Sun SPOT will supply the Java programmers with new methods to express their ideas and work, and it can attract new programmers to the environment of Java. The project also aims to accelerate the creation of the "Internet of things" [11].

2.2 Sun SPOT device

The Sun Small Programmable Object Technology (SPOT) is a small, battery powered, wireless, experimental platform. It is designed to be easy to program and use so that any Java programmer can think beyond the keyboard, mouse and screen and thus be inspired to build new applications using Java [12].

The Sun SPOT wireless sensor network components shown in Figure 1, used as a base-station or free-range unit. The base station is connected to a host computer via USB cable and it can communicate with the free-range Sun SPOTs using radio connection. The base-station and the free-range SPOT have the same mainboard, but the free-range SPOT has its own sensor application board and a rechargeable battery. The base-station spot has no rechargeable battery, because it is always connected to a host computer. The

main board has the basic components: main processor, memory, power management circuitry, and network radio transceiver.

A) Free-Range Sun SPOT B) Base-Station Sun SPOT

Figure 1: Free-Range and Base-Station Sun SPOTs.

2.3 Sun SPOTs Applications

As the spots can be virtually installed at any place, the applications are unlimited. Here are some examples:

- Environmental monitoring, including: monitoring plants moment by moment , such as using autonomous measuring systems to monitor all the necessary parameters for creating the optimal environment in the greenhouse [19], and understanding wildlife habitats.
- Structural monitoring, including: monitoring bridges and buildings.
- Medical monitoring, including: monitoring patient vital signs [18], swallow-able sensors, monitoring emergency rooms.
- Home monitoring, including: security systems and home medical monitoring [11].

- Robotics, including: building swarms of cooperating robots that communicate via their radio link [11].
- Personal Applications, including: light monitoring to turn on auto lights during evening.
- Toys and Art, including: controlling kits.
- Commercial Application, including: monitoring water quality.

2.4 Sun SPOT Components

Table 1: Sun Spot Components

Component	Description
Main Board	It is the same in the base-station and free-range spot [20]. It contains: o 400 MHz 32 bit AT91SAM9G20 core - 512K RAM/4M Flash o 2.4 GHz IEEE 802.15.4 radio with integrated antenna o USB interface o 3.7V rechargeable 770 mAh lithium-ion battery o 36 uA deep sleep mode.
Sensor board	

	It is just in the free-range Sun SPOT. It contains: o 2G/6G 3-axis accelerometer o Temperature sensor o Light sensor o 8 tri-color LEDs o 6 analog inputs o 2 momentary switches o 5 general purpose I/O pins and 4 high current output pins
Squawk Virtual Machine	It is a Java virtual machine mainly written in Java, the interpreter is written in C and garbage collector translated from Java to C [11]. It is: o Fully capable J2ME CLDC 1.1 Java VM with OS functionality o VM executes directly out of flash memory o Device drivers written in Java o Automatic battery management
Developer Tools	They are the tools that are used by the programmer to write the

	sun SPOT application code [11].Standard IDEs like NetBeans, which is used to create the Java code. It is capable to integrate with J2SE applications. The Sun SPOT which is wired via USB to a computer acts as a base-station.

2.5 Distinctions of Sun SPOTs

The Sun SPOT is different from other wireless sensors in the following ways [11]:

- **The system is Java-based.**
    - This makes it easier for programming and integration with networked applications.
    - Developers can use the standard Java development tools such as NetBeans.
    - Java is running "on-the-metal." It handles many of the services traditionally supplied by an OS.
- **Isolates.**
    - Multiple applications can run at the same time without requiring multiple virtual machines and significant amounts of memory.
    - The Sun SPOTs are implemented as objects. This allows the developer to start, stop, or migrate the running applications from one device to another.
- **Battery management**

- Power is a scarce resource on mobile devices. The SPOT has a very low power (36 μA) sleep mode.
- Ultimately it is preferable that this management be analogous to how Java handles memory management through garbage collection and thus frees the developer to concentrate on the logic of their application rather than chasing down memory leaks.

- **The device is quite powerful**
  - It is a full 32-bit processor with lots of easily accessible I/O for experimentation.

2.6 SPOT Wireless Communication

The wireless radio communication part of the device is controlled by a CC2420 chip that is compatible with IEEE 802.15.4 standard. It operates in a frequency range of 2.4GHz to 2.4835GHz ISM (Industrial, Scientific and Medical) unlicensed bands that comply with FCC CFR47 regulations. This band is divided into 16 different frequency channels and 8 different power levels; each channel has its own central frequency (Table 2). The CC2420 chip can be adjusted programmatically to 22 different power levels [13]. More power levels mean more radio coverage area. The Sun SPOT can be programmed for different applications [14] and these applications can be deployed to the SPOT remotely by using Over The Air (OTA) command or using USB connection.

Table 2: Sun Spot channels ant their central frequency.

Channel	Central Frequency		Channel	Central Frequency
11	2405MHz		19	2445MHz
12	2410MHz		20	2450MHz
13	2415MHz		21	2455MHz
14	2420MHz		22	2460MHz
15	2425MHz		23	2465MHz
16	2430MHz		24	2470MHz
17	2435MHz		25	2475MHz
18	2440MHz		26	2480MHz

The output power can be adjusted by the PA_LEVEL register, which is a 6 bit field. PA_Level and output power are shown in the table below:

Table 3: PA_Level and their output power.

PA_Level	Output Power		Channel	Output Power
31	0dBm		15	-7dBm
27	-1dBm		11	-10dBm
23	-3dBm		7	-15dBm
19	-5dBm		3	-25dBm

2.7 For how long will the battery power the Sun SPOT?

The Free-range SPOT is powered by an external battery with 770mAh capacity and an output voltage of 3.7 V; also it can be powered using USB power from a host computer or externally connected power supply. When the SPOT is powered by USB or external power supply, the battery will be in charging mode. SPOT can be in different power modes. The modes are deep sleep, shallow sleep, and awake. In the deep sleep mode, the processor and the application board are powered down. Also, there is no radio connection in this mode. The SPOT can stay on for days using battery without recharging. The shallow sleep mode makes the device active for any response even if there are no active threads running on the processor [20]. The active mode is when the CPU is busy and processing the running threads. The estimate life of the SPOT battery is shown in table 4[13].

Table 4: Sun SPOT states and their battery life estimate.

Sun SPOT State	Battery Life Estimate
Deep sleep	909 Days
Shallow sleep, no radio	23 Hours
Shallow sleep, radio on	15 Hours
CPU busy, no radio	8.5 Hours
CPU busy, radio on	7 Hours
Shallow sleep, 8 LEDs on, no radio	3 Hours

2.8 Sensor Board:

It is also called application board and it is only on the free-range SPOTs [13]. Figure 2 shows the sensor board.

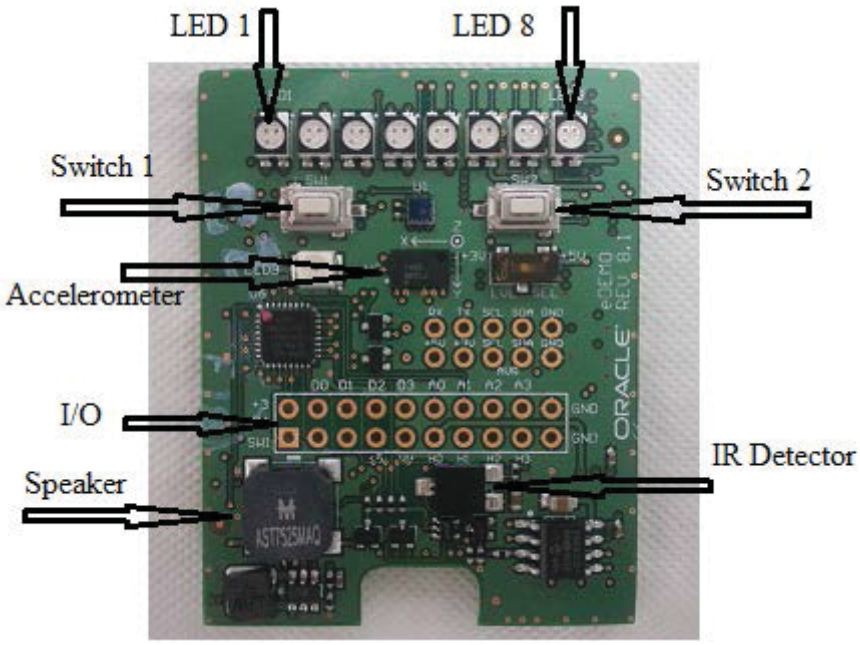


Figure 2: Free-Range Sun SPOT application board.

The demo sensor board has 8 RGB LEDs in a row, with LED1 on the left and LED8 on the right. Each LED has a red, green, and a blue emitter as a part of the LED. Each individual color can have intensity from 0 to 255, with 0 means off and 255 means the brightest.

Chapter 3

Wireless Sensor Network (WSN)

A wireless sensor network (WSN) in its simplest form can be defined as a network of devices called Nodes that can sense the environment and communicate the collected sensor data from the monitored field through wireless links. The wireless sensor data can be forwarded, possibly via multiple hops relaying, to a sink that can use it locally; it can also be connected to other networks (e.g., the Internet) through a gateway [21] as shown in Figure 3.

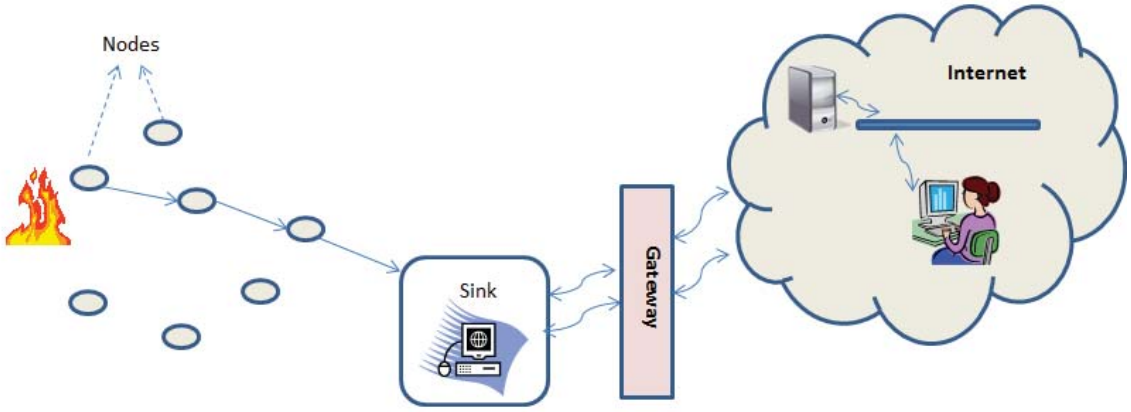


Figure 3: Wireless sensor network (WSN).

3.1 WPAN and LR-WPAN Standards

To transmit data over relatively short distances, the Wireless personal area networks (WPANs) are used. Unlike wireless local area networks (WLANs), the WPANs network connections involve little or no infrastructure. This feature allows small, power-efficient, inexpensive solutions to be implemented for a wide range of devices [15].

There are many commercial WPAN technologies available, such as: Bluetooth, IrDA, ZigBee, etc. These technologies can be used to connect personal devices within a small distance of communication, and they are based on the IEEE 802.15 standard [16].

The Sun SPOT is based on the IEEE 802.15.4 standard or Low Rate Wireless Personal Area network (LR-WPAN). This standard has been developed to define the physical and data link layers of any wireless sensor node categorized under LR-WPAN. The key difference between LR-WPAN and other WPAN is that the LR-WPAN is specifically intended for wireless devices that are low-cost manufacturing property, short-range operation, low-speed data transfer, extremely low-power consumption, and have simple and flexible installation with little or no infrastructure [17].

3.2 Network Layers Architecture

The sensor network layer is divided into different layers and each layer has its functionality. The layers can be classified into three types; Physical layer, Medium Access Control (MAC) layer, and Upper Layer(s). The LR-WPAN device architecture is shown in Figure 4.

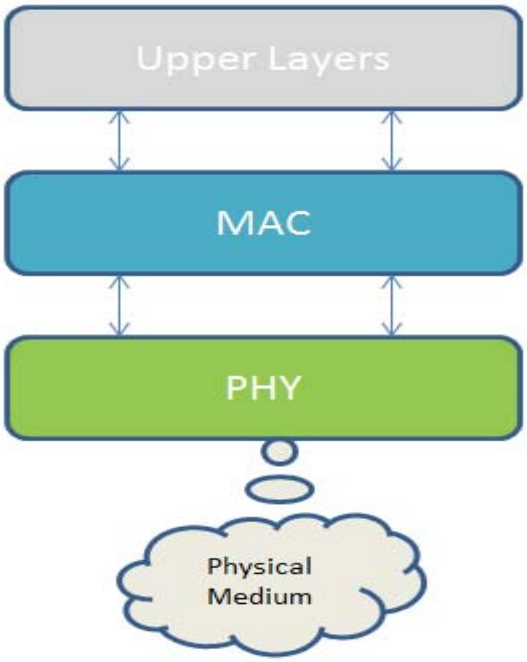


Figure 4: Sensor network layers architecture.

The IEEE 802.15.4 standard focuses on PHY and MAC layers; Physical layer is responsible for lower level radio frequency functionality and management of LR-WPAN devices. It encapsulates the data with additional information like packet sequence number, source and destination addresses.

The medium access control (MAC) layer provides two services: the MAC data service and the MAC management service. The MAC data service enables the transmission and reception of MAC protocol data units (MPDUs) across the PHY data service. The features of the MAC sublayer are beacon management, channel access, guaranteed time slot (GTS) management, frame validation, acknowledged frame delivery, association, and disassociation. In addition, the MAC sublayer provides hooks for implementing applications that are appropriate for the security mechanisms [17].

Upper layers are composed of a network layer and an application layer. Application layers are responsible for deliberating the work function of the device, whereas the network layers are important for network topology configurations, data routing, naming, and addressing.

3.3 Radiostream and Radiogram protocols

The upper layers of the Sun SPOT have radiostream and radiogram protocols. The radiostream protocol is used in peer-to-peer buffered stream-based communication where the two SPOTs can open radio communication with a destination IEEE extended MAC address and a desired port number that ranges from 0 to 255.The IEEE extended MAC address is a 64-bit address, expressed as four sets of four-digits hexadecimal numbers: nnnn.nnnn.nnnn.nnnn. In Sun SPOTs the first eight digits are always 0014.4F01. The last eight digits should be printed on a sticker visible through the translucent plastic on the radio antenna fin. When the radio communication is opened, the two SPOTs can send and receive data by streaming out or streaming in the data. This protocol can work as single-hop or multi-hops [13].

The radiogram protocol is a datagram protocol that allows the exchange of data packets between two devices. It can be operated as a point-to-point or client-server model. In the point-to-point communication model, a radio communication on both ends has to be opened, with each corresponding MAC address as the destination address and the same port number on both ends. In the client-server communication model, all clients should open a radio communication with the server MAC address and the port number, and the server has to specify the port number without any MAC address. Therefore, the

server can listen to any number of communications, not just one. The radiogram protocol also supports broadcasting. The broadcast packets can be received and processed by all wireless sensor nodes that are on the same radio channel and the same PAN identifier as the sender. In the broadcast, the hop count can be set during the process of opening the radio communication. So, the broadcast packet can be controlled in the terms of how far it can be reached.

In both protocols, there are acknowledging packets that are exchanged among the nodes to inform senders when the packets are received. The radio channel and PAN identifier can be changed by accessing the radio property of the SPOT. This can be done either programmatically or by changing the properties of SPOT using the Sun SPOT Manager program [13].

Communication between the free range sensor and the base station can occur in both push and pull modes and in regular periods or on demand. The values are communicated over the meshed wireless sensor network. All configurations can be defined by the user during run-time. In addition, a user can employ a sensor value reader device (essentially another Sun SPOT) in order to get data from a specific sensor by physically moving into the communication range of that sensor and querying the proper data.

3.4 Network Topology

The network topology is the schematic description of the arrangement of a network, including its nodes and connecting lines. There are two ways of defining the network geometry: the physical topology and the logical (or signal) topology.

The physical topology of a network is the actual geometric layout of workstations. There are several common physical topologies like bus, star, ring, mesh, and tree as shown in Figure 5.

Logical (or signal) topology refers to the nature of the paths the data follow from node to node. Usually, the logical topology is the same as the physical topology. But in some cases the networks can be configured physically in one topology, and operate in another. For example, networks may be physically laid out in a star configuration, but they operate logically as bus or ring networks.

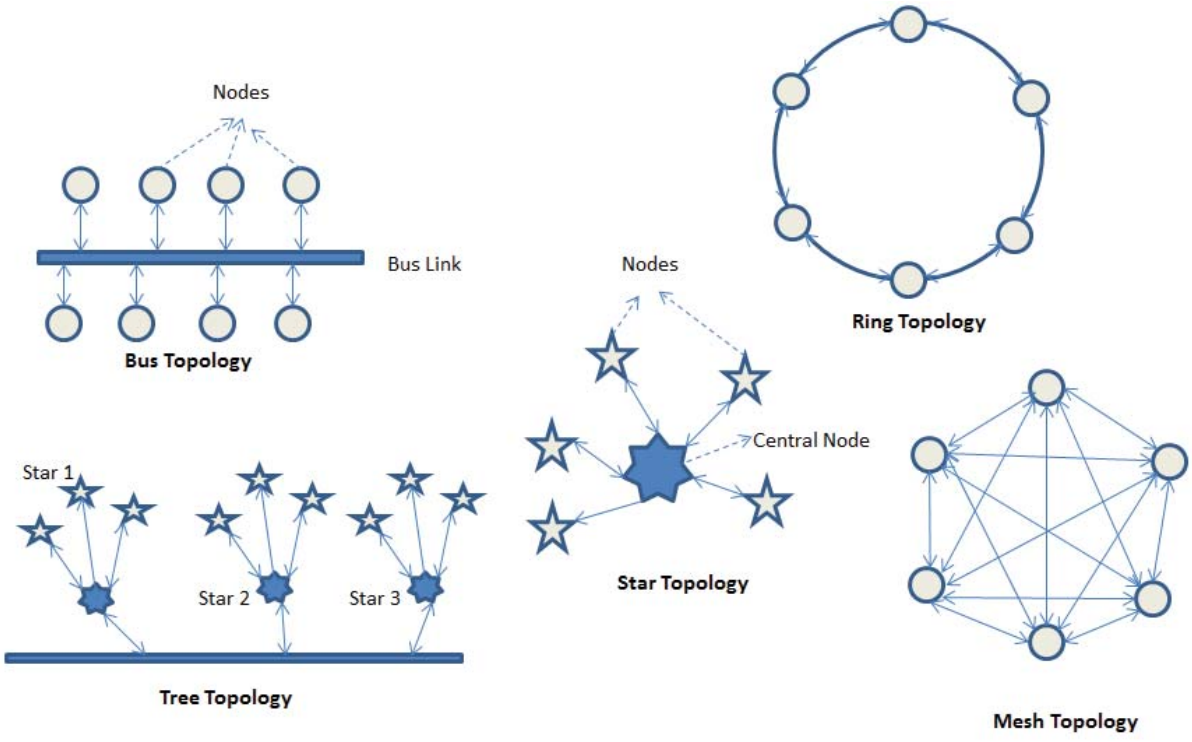


Figure 5: Network Topologies.

3.4.1 Bus Topology

A bus network topology is an arrangement in a local area network (LAN) in which each node is connected to a main cable or link (bus). A bus network is simple and reliable. If one node fails to operate, all the other nodes can communicate with each other as the bus is still working. Bus networks are easy to expand, that is, additional nodes can be added anywhere along the bus. The disadvantages of the bus network topology include: the length of the bus is limited by cable loss; a bus network may not work well if the nodes are located at scattered points that do not lie near a common line, and only one node can transmit at a time. The bus is a bottleneck of the network, which means that if it is disconnected the whole network will be affected [22].

3.4.2 Star Topology

In the star topology all the nodes will be directly connected to a common central node. The star network topology is good when the nodes are at scattered points. It is easy to add or remove nodes and if any single node failed to operate, the whole network will not be affected. The main disadvantage includes: if the central node goes down, the entire network will face degraded performance or complete failure [23].

3.4.3 Ring Topology

In the ring topology the nodes will be connected in a closed loop configuration. Adjacent pairs of nodes are directly connected. Messages from one node to another travel from originator to destination via the set of intermediate nodes; the communication can be unidirectional or bidirectional [25]. It is good for peer-to-peer communication as it

needs no main node, which also gives high performance. The main disadvantages include: if one node fails, the whole network will be affected; the entire network is disturbed while appending or removing a node and it is difficult to troubleshoot.

3.4.4 Mesh Topology

In the mesh topology there will be two connection arrangements, full mesh topology and partial mesh topology. In the full mesh topology, each node is connected directly to each of the others so if there are n nodes, then the number of connections is n(n-1)/2. In the partial mesh topology, some nodes are connected to all the others, but some of the nodes are connected only to those nodes with which they exchange the most data [24]. A mesh network is reliable and offers redundancy. If one node is no longer operating, all the rest can still communicate with each other, directly or through one or more intermediate nodes. Mesh networks work well when the nodes are located at scattered points that do not lie near a common line. The main disadvantage is installation, which means that it is difficult to install because the number of nodes increases and it becomes more complex.

3.4.5 Tree Topology

A tree network is a combination of two or more star networks that are connected together. The central nodes of the star networks are connected to a main bus. So we can define a tree network as a bus network of star networks [2]. The tree network topology is ideal when the workstations are located in groups, with each group occupying a relatively small physical region. It is easy to add or remove workstations from each star network.

Entire star networks can be added to, or removed from, the bus. The main disadvantage of tree topology is that when the Head-end node fails to operate, the entire network will shut down.

3.5 Distinctions of Wireless Sensor Networks over traditional communication networks

There are some characteristics that distinguish WSNs apart from other communication networks [26]:

The WSNs are data centric; this means that the communication should be targeted to the nodes in a given location or with defined data content. A WSN is deployed to perform a specific task like habitat or environment monitoring. Communication links between the nodes are not stable due to the node errors and failures. Unreliable and simple modulations, depletion of batteries, mobility of nodes, and environmental interferences can affect the communication between nodes. So, the functionality must be maintained regardless of the built-in dynamic nature and the failures of the nodes. The number of nodes comprising WSNs may be huge and the density of nodes can be high. The computation, communication, and memory resources in WSN nodes are limited as the WSN node is small in physical size and battery-powered. Thus, energy-saving and load-balancing must be taken into account in the design and implementation of WSN platforms, protocols, and applications. As WSNs are tightly related to the real world, strict timing constraints for sensing, processing, and communication should be available in the WSNs. Also the need for security in WSNs is evident, especially in health care and military applications.

3.6 Experiment Topology

I used two free range Sun SPOTs and base-station as shown on Figure 7.

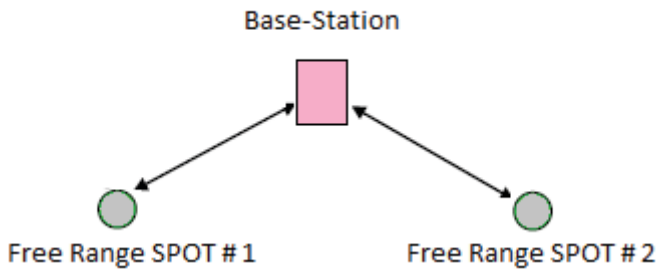


Figure 6: Project network topology

The future expansion of this network can be done as in the Figure 8

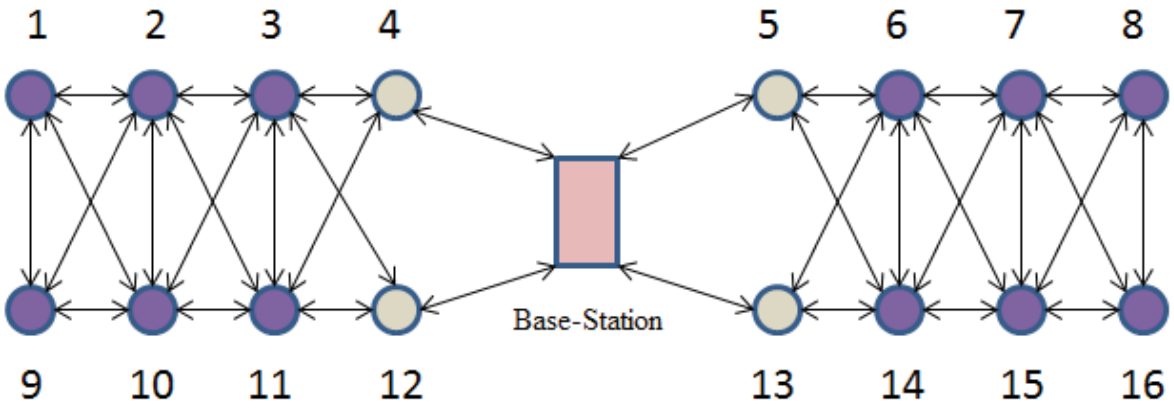


Figure 7: Future expansion of the wireless sensor network topology.

In this experiment, mesh topology is used. This means that each node has more than one path to communicate with the base-station and then to give the alarm of existing fire in the node region. For example, node 8 in Figure 7 can communicate with the base-station through many routs and the best route is determined by the routing protocol. If node 8 failed to reach the base-station using the shortest path 8 - > 7 - > 6 - > 5 - > base-

station, it can use another path like 8 - > 16 - > 15 - > 14 - > 13 - > base-station and this alternate path or any other path can be determined using a routing protocol.

3.7 Routing Protocol

A routing protocol specifies how the nodes communicate with each other and disseminate the information that enables them to select the routes between any two nodes in a network. Routing algorithms determine the specific choice of route. Usually since, the coverage area of the entire network is bigger than the single sensor coverage area, a multi-hop traffic can happen in the network. This multi-hop should be managed in an organized way.

The sun SPOT sensors use the Link Quality Routing Protocol (LQRP) Manager to manage the traffic routing, which is built-in software that takes care of the link quality routing protocol. This software enables the SPOT to act as a mesh router. When a running application starts to open a radio connection, either radiostream or radiogram connection, the LQRP Manager will start to send and receive LQRP packets. The operation of LQRP is based on an algorithm that works to determine the best route. In this algorithm, requests for a route to a particular targeted SPOT are broadcasted by a requester and then re-broadcasted by any Sun SPOT that receives them. Each Sun SPOT that can reach the requested target sends a reply back to the requester. The selected route will be the one with the best link quality [13].

In this thesis, two wireless sensor nodes are not too far from the base-station, and the base-station covers both sensors. The routing protocol will always determine that the destination address is one hop from the source. Therefore, when a sensor node sends a

packet to the base-station, it will directly send the packet to the base station's address without sending the packet to any other intermediate sensor nodes.

Chapter 4

Experimental Procedure

This chapter demonstrates how to detect fire using WSN.

4.1 Fire Detection Algorithm

The fire detection depends on two conditions; a sudden change in the average RGB value for the light wavelength, and a sudden change in the temperature. If the two conditions are satisfied the system will give an alarm of type 1. If the first condition is satisfied without the second one, the system will give an alarm of type 2. If the second condition is satisfied without the first one, the system will give an alarm of type 3.

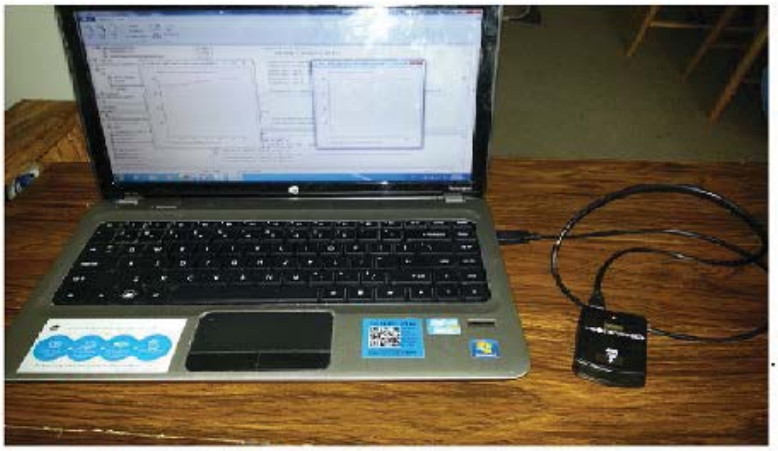
4.2 Experiment Components

The components that were used in the fire detection experiment are:

- Base Station connected to host computer.
- Two Sun SPOT sensors.
- Fire source.

A) Sun SPOT sensor in region A

B) Sun SPOT sensor in region B



C) Base-Station connected to host computer

Figure 8: Fire detection using WSN.

The base-station and the two free-range sensors are configured in a mesh topology. The base station periodically listens to the two sensors.

4.3 Experiment Results

The Java program that is deployed to each sensor measures the temperature and light values every 10 seconds and sends these values via radiogram communication to the base station which is attached to a host computer. The Java application program that runs

on the computer, always takes the average of the last 5 readings of the light and the temperature of each sensor, then it compares the new temperature value with the pre-calculated temperature average for each sensor, and then if the difference between the new temperature value and the average is greater than assigned temperature threshold value, there would be a potential of fire. The same scenario is applied to the light; if the difference between the new light value and the pre-calculated light average value is greater than or equal assigned light threshold value, there would be a potential of fire flames.

If the two conditions are both satisfied: the difference between the temperature value and the temperature average value is greater than or equal the temperature threshold value and difference between the light value and the light average value is greater than or equal the light threshold value, there would be a great possibility of fire in the sensor area and then the alarm of type 1 will be fired as shown in Figure 9.

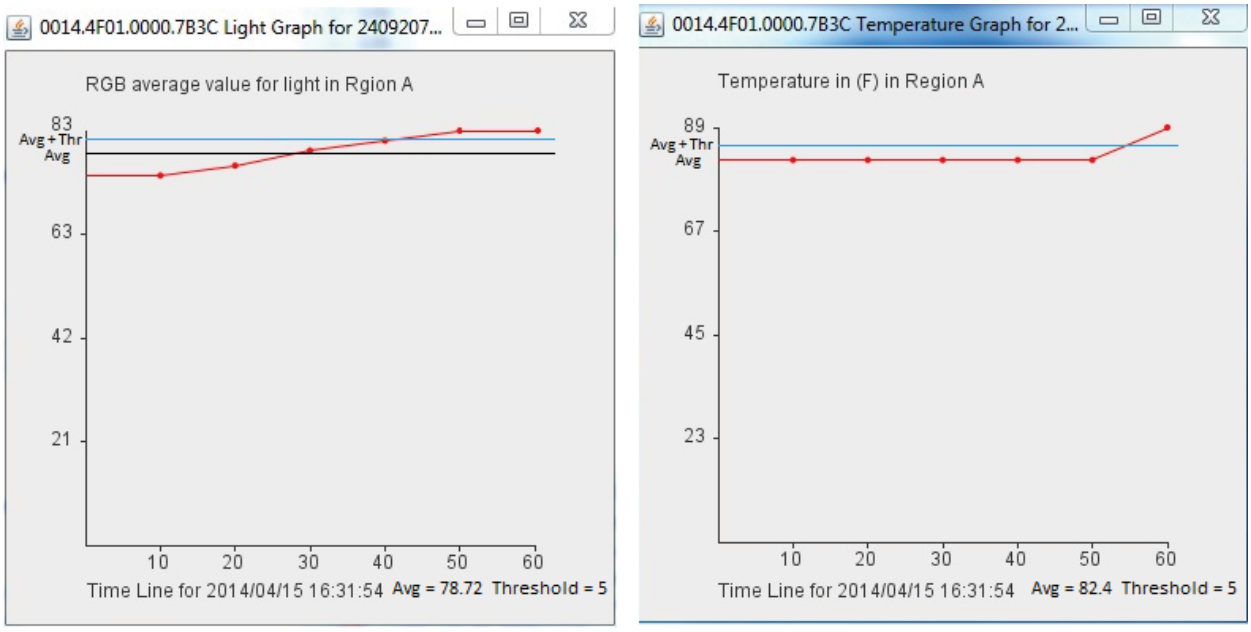


Figure 9: Light and temperature graphs for Type 1 alarm.

If one condition is satisfied, then the possibility of fire in this case will be less than the previous case when the two conditions are satisfied, especially when the light condition is satisfied without the temperature condition because the sudden change in light value may be as a result of turning on a lamp. But it will give alarm of type 2 as shown in Figure 10.

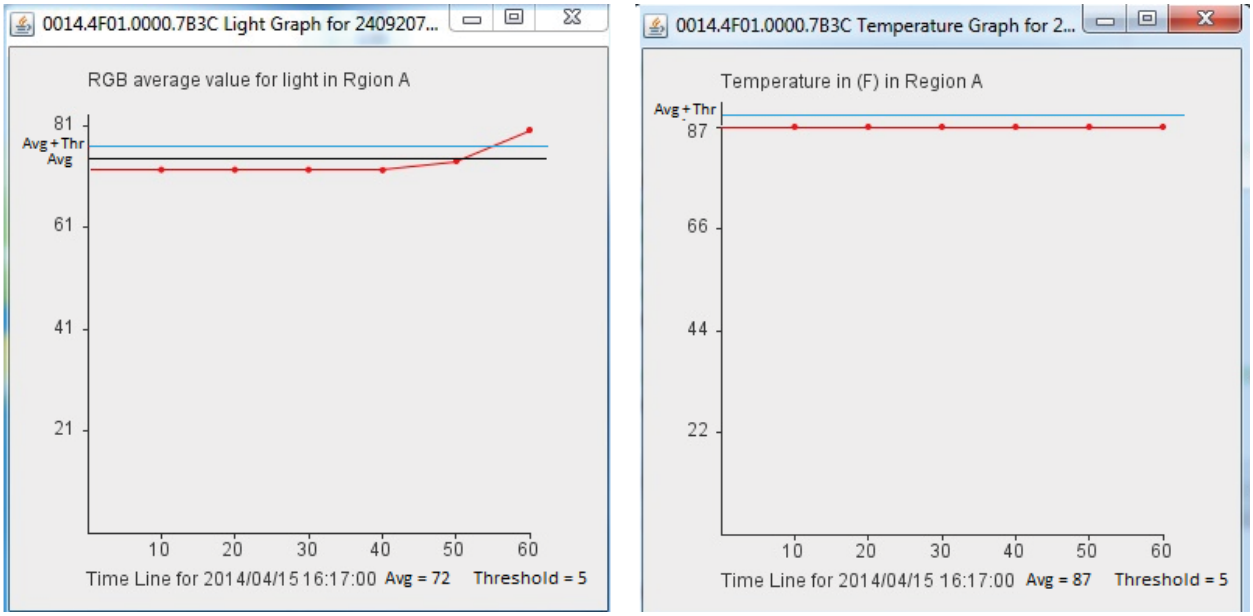


Figure 10: Light and temperature graphs for type 2 alarm.

If the temperature condition is satisfied without light condition, it will give alarm of type 3 as shown in Figure 11 because there is a possibility for a fire and the light is hidden to the Sun Spot light sensor by something in that area.

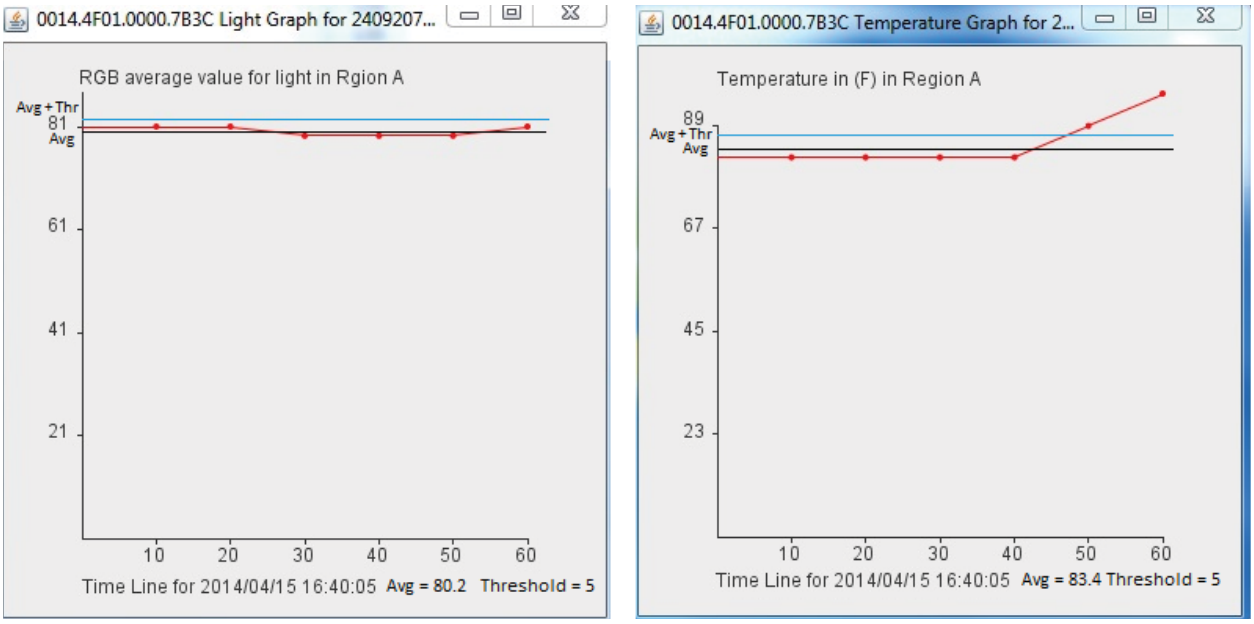


Figure 11: Light and temperature graphs for type 3 alarm.

It is clear that type 1 has the highest priority, so immediate action must be taken according to this alarm. Type 2 has less priority than type 1, so we can wait to make sure the temperature will increase. If the temperature doesn't increase, this means that there is no fire and the change in light will be from a light source. Type 3 has less priority than type 1 and higher priority than type 2 because the increase in temperature without any changing in light is a strong indicator of fire.

Chapter 5

Future Work and Conclusion

5.1 Future Work:

A predicting system can be developed using statistical models to predict the fire event according to historical data from the WSNs. Future WSNs will still be constrained by simple power supplies from both passive (e.g., solar and vibration) and active (e.g., battery power and external power source) sources of energy. Thus, energy efficiency and reliable connectivity will continue to receive significant attention in the research and design of WSNs. It is possible to use external sensors for smoke and humidity and these external sensors can be interfaced with the Sun SPOT using I/O pins. Continuous improvement in the abilities of Sun SPOT, such as the number and type of sensors, power consumption, and wireless communication will greatly help the world.

5.2 Conclusion:

The results of this research indicate that the SPOTs can be used to develop a reliable, wirelessly connected network that is capable to detect the existence of dangerous events like fire hazards.

References:

[1] Kapoor, N., Bhatia, N., Kumar, S., and Kaur, S."Wireless Sensor Networks: A Profound Technology" International Journal of Computer Science and Technology, Vol. 2, Issue 2, June 2011:211-215.

[2] Chong, Chee-Y., and Kumar, S., P." Sensor Networks: Evolution, Opportunities, and Challenges" Proceedings of the IEEE, Vol. 91, NO. 8, August 2003:1247-1256.

[3] Rashid, R., and Robertson, G." Accent: A communication oriented network operating system kernel" Proc. of the 8th Symposium on Operating System Principles, 1981:64–75.

[4] Myers, C., Oppenheim, A., Davis, R. and Dove, W."Knowledge-based speech analysis
and enhancement" Proc. of the International Conference on Acoustics, Speech and Signal Processing, 1984.

[5] Kumar, S. and Shepherd, D."Sensit: Sensor information technology for the warfighter"
Proc. of the 4th International Conference on Information Fusion (FUSION'01), 2001: 3–9.

[6] IEEE 802.15 WPAN Task Group 4. Retrieved from http://www.ieee802.org/15/pub/TG4.html.

[7] ZigBee Alliance. Retrieved from http://www.zigbee.org.

[8] Georgoulas, D., and Blow, K." Wireless Sensor Network Management and Functionality: An Overview" Wireless Sensor Network, 2009:257-267.

[9] Firewize Maintenance innovation for life. Retrieved from http://firewize.com/smoke-gas-flame-fire-detectors-principle-operation

[10] National Fire Protection Association Journal. Retrieved from http://www.nfpa.org/newsandpublications/nfpa-journal/2012/march-april-2012/features/a-harder-look-at-detection

[11] Sun SPOT Frequently Asked Questions. Retrieved from https://java.net/projects/spots/sources/svn/content/trunk/www/FAQ.html%3Fraw%3Dtrue+&cd=1&hl=en&ct=clnk&gl=us

[12] Randall, B. S."SPOTWorld and the Sun SPOT." 6th International Symposium on Information Processing in Sensor Networks, Cambridge, MA, USA, 2007:565 - 566.

[13] Sun SPOT Programmer's Manual, Sun Labs, November 2010, Release v6.0(Yellow). Retrieved from www.sunspotworld.com/docs/.../SunSPOT-Programmers-Manual.pdf

[14] Arseneau, E.,Goldman, R., Poursohi, A., Randall, B.S., and Daniels, J." Simplifying the Development of Sensor Applications," Sun Microsystems, 2006.

[15] The Institute of Electrical and Electronics Engineers, "Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area," IEEE Standard 2011.

[16] Jang, C.S., Lee, D.G., Han, J.W., and Park, J.H."Hybrid Security Protocol for Wireless Body Area Networks," Wireless Communications and Mobile Computing, 2011.

[17] IEEE Standard for Information Technology. 802.15.4, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," IEEE 802.15.4, 2006:1-26.

[18] Giancarlo Fortino, G., and Galzarano, S." Programming Wireless Body Sensor Network

Applications through Agents," University of Calabria (UNICAL).

[19] Janos, S., Martinovic, G., and Matijevics, I." WSN Implementation in the Greenhouse

Environment Using Mobile Measuring Station," Volume 1, Number 1, June 2010.

[20] Sun Labs, "Sun SPOT - Main Board Technical Datasheet," Oracle America, Inc., Technical Datasheet, 2010.

[21] Mittal, D., and Kaur, S." Enhanced Location-Aware Routing Protocol for Wireless Sensor Network," International Journal of Science and Modern Engineering (IJISME), Volume-1, Issue-12, November 2013:38-43

[22] Pandya, K." Network Structure or Topology," International Journal of Advance Research in Computer Science and Management Studies, 2013:22-27.

[23] Roberts, Lawrence,G., Wessler, and Barry D. "Computer network development to achieve resource sharing", Proceedings of the May 5–7, 1970, spring joint computer conference, New York, NY, USA, 1970:543–549.

[24] Mahmud S., Khan S, and AL-RAWESHIDY, H." Meshed High Data Rate Personal Area Networks," IEEE Communications Surveys and Tutorials,1st Quarter 2008:58-69.

[25] Meador, B." A Survey of Computer Network Topology and Analysis Examples," Washington University in St. Louis, 2008.

[26] Sindhanaiselvan, K., and Mekala, T." A Survey on Sensor Cloud: Architecture and Applications," International Journal of P2P Network Trends and Technology (IJPTT), Volume 6, March 2014.

Appendix A

1) The desktop Java code:

Copyright (c) 2008-2009 Sun Microsystems, Inc.

```
package org.sunspotworld.demo;

//import com.sun.media.sound.Toolkit;
import com.sun.spot.io.j2me.radiogram.*;

import com.sun.spot.peripheral.ota.OTACommandServer;
import java.text.DateFormat;
import java.util.Date;
import javax.microedition.io.*;
import java.io.FileWriter;
import java.util.ArrayList;
import  sun.audio.*;    //import the sun.audio package
import  java.io.*;
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
```

```
import java.util.Calendar;
/**
 * This application is the 'on Desktop' portion of the SendDataDemo.
 * This host application collects sensor samples sent by the 'on SPOT'
 * portion running on neighboring SPOTs and just prints them out.
 *
 * @author: Vipul Gupta
 * modified: Ron Goldman
 */
public class SendDataDemoHostApplication {
    // Broadcast port on which we listen for sensor samples
    private static final int HOST_PORT = 67;
    ArrayList<Integer> lightArray1=new ArrayList<Integer>();
    ArrayList<Integer> tempArray1=new ArrayList<Integer>();
    ArrayList<Integer> lightArray2=new ArrayList<Integer>();
    ArrayList<Integer> tempArray2=new ArrayList<Integer>();
    ArrayList<Integer> tempArr=new ArrayList<Integer>();
    ArrayList<Integer> tempArr2=new ArrayList<Integer>();
     int HOST_PORT2=66;
     int lightAverage1;
     int tempAverage1;
      int lightAverage2;
     int tempAverage2;
```

```
 int lightThreshold =5;
 int tempThreshold=5;
private void run() throws Exception {
  // while (true) {
   FileWriter fw=new FileWriter("E:\\at.txt");
   int counter=0;
   do{
      RadiogramConnection rCon;
      Datagram dg;
      DateFormat fmt = DateFormat.getTimeInstance();
      try {
        HOST_PORT2 =66 + counter%2;
      // Open up a server-side broadcast radiogram connection
      // to listen for sensor readings being sent by different SPOTs
        rCon = (RadiogramConnection) Connector.open("radiogram://:" +
HOST_PORT2);
        dg = rCon.newDatagram(rCon.getMaximumLength());
         }
       catch (Exception e) {
              System.err.println("setUp caught " + e.getMessage());
                throw e;
                    }
```

```
// Main data collection loop
    try {
      // Read sensor sample received over the radio
      rCon.receive(dg);
      String addr = dg.getAddress();  // read sender's Id
      long time = dg.readLong();      // read time of the reading
      int val = dg.readInt();         // read the sensor value
      double tempValue=dg.readDouble();
      int tempVal = (int)tempValue;
      counter++;
      if(HOST_PORT2 == 66)
      {
       if(lightArray1.size() < 5)
          lightArray1.add(val);
       else
       {
          int sum=0;
          for(int i=0;i<5;i++)
             sum =sum + lightArray1.get(i);
          tempArr=lightArray1;
          lightAverage1= sum/5;
          if(tempVal > tempAverage1+tempThreshold || val >
lightAverage1+lightThreshold )
```

```
                runAlarm(addr,time,tempArr,val,tempArr2,tempVal);
            lightArray1.set(0, lightArray1.get(1));
            lightArray1.set(1, lightArray1.get(2));
            lightArray1.set(2, lightArray1.get(3));
            lightArray1.set(3, lightArray1.get(4));
            lightArray1.set(4, val);
        }
        if(tempArray1.size() < 5)
            tempArray1.add(tempVal);
        else
        {
            int sum=0;
            for(int i=0;i<5;i++)
                sum =sum + tempArray1.get(i);
            tempArr2=tempArray1;
            tempAverage1= sum/5;
            if(tempVal > tempAverage1+tempThreshold || val >
lightAverage1+lightThreshold )
                runAlarm(addr,time,tempArr,val,tempArr2,tempVal);
            tempArray1.set(0, tempArray1.get(1));
            tempArray1.set(1, tempArray1.get(2));
            tempArray1.set(2, tempArray1.get(3));
            tempArray1.set(3, tempArray1.get(4));
```

```
                tempArray1.set(4, tempVal);
                if(tempVal > tempAverage1+tempThreshold || val >
lightAverage1+lightThreshold )
                    runAlarm(addr,time,tempArr,val,tempArr2,tempVal);
                /////
             }
            }
            else
            {
                if(lightArray2.size() < 5)
                lightArray2.add(val);
             else
             {
                int sum=0;
                for(int i=0;i<5;i++)
                    sum =sum + lightArray2.get(i);
                tempArr=lightArray2;
                lightAverage2= sum/5;
                lightArray2.set(4, lightArray2.get(3));
                lightArray2.set(3, lightArray2.get(2));
                lightArray2.set(2, lightArray2.get(1));
                lightArray2.set(1, lightArray2.get(0));
                lightArray2.set(0, val);
```

```
            //if(val > lightAverage2+lightThreshold )
             //   runAlarm(addr,tempArr,val);
          }
            if(tempArray2.size() < 5)
            tempArray2.add(tempVal);
          else
          {
            int sum=0;
            for(int i=0;i<5;i++)
               sum =sum + tempArray2.get(i);
            tempArr2=tempArray2;
            tempAverage2= sum/5;
            tempArray2.set(4, tempArray2.get(3));
            tempArray2.set(3, tempArray2.get(2));
            tempArray2.set(2, tempArray2.get(1));
            tempArray2.set(1, tempArray2.get(0));
            tempArray2.set(0, tempVal);
            if(tempVal > tempAverage2+tempThreshold || val >
lightAverage2+lightThreshold )
               runAlarm(addr,time,tempArr,val,tempArr2,tempVal);
          }
          }
```

```
            System.out.println(HOST_PORT2 + "**** " +fmt.format(new Date(time)) + "
from: " + addr + "    Light value = " + val+ " Temperature value = "+tempValue);
            String s=HOST_PORT2 + "**** " +fmt.format(new Date(time)) + "  from: " +
addr + "    Light value = " + val+ " Temperature value = "+tempValue;
            fw.write(s);
            fw.write("\r\n");
            rCon.close();
          } catch (Exception e) {
            System.err.println("Caught " + e +  " while reading sensor samples.");
            throw e;
          }
      }while(counter > 0);// this means ifinite loop but to be able to write values on screen
whilr reading
      // if I use while (true) it will not write on the screen
      fw.close();
   }
   /**
    * Start up the host application.
    *
    * @param args any command line arguments
    */
```

```
   public void runAlarm(String addr,long time,ArrayList<Integer> LightAr,int
val,ArrayList<Integer> TempAr,int tempVal)
   {
       System.out.println("&&& @@@@@@@@@  ********* "+addr);
      // for(int i=0;i<10;i++)
       java.awt.Toolkit.getDefaultToolkit().beep();
      JFrame f = new JFrame(addr+" Light Graph for "+time);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.add(new DrawValues("RGB average value for light in Rgion A",addr, LightAr, val));
f.setSize(400,400);
f.setLocation(200,200);
f.setVisible(true);
JFrame f2 = new JFrame(addr+" Temperature Graph for "+time);
f2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f2.add(new DrawValues("Temperature in (F) in Region A ",addr, TempAr, tempVal));
f2.setSize(400,400);
f2.setLocation(800,200);
f2.setVisible(true);
try{
InputStream in = new FileInputStream("E:\\ss.mp3");
// Create an AudioStream object from the input stream.
AudioStream as = new AudioStream(in);
// Use the static class member "player" from class AudioPlayer to play
```

```
// clip.
AudioPlayer.player.start(as);
for(int k=0;k<999999999;k++);
// Similarly, to stop the audio.
AudioPlayer.player.stop(as);
        }
catch(Exception e2){System.out.print(e2.getMessage());}
    }
    public static void main(String[] args) throws Exception {
        // register the application's name with the OTA Command server & start OTA
running
        OTACommandServer.start("SendDataDemo");
        SendDataDemoHostApplication app = new SendDataDemoHostApplication();
        app.run();
    }
}
class DrawValues extends JPanel {
int[] data;
final int PAD = 50;
int val;
String graphH;
public DrawValues( String graphHeader,String addr,ArrayList<Integer> ar,int val)
    {
```

```
    graphH=graphHeader;
    data =new int[ar.size()+1];
    int counter=-1;
    for(int m=ar.size()-1;m>=0;m--)
    {
       counter++;
       data[counter]=ar.get(m);
       }
    data[ar.size()]=val;
}
@Override
protected void paintComponent(Graphics g) {
  super.paintComponent(g);
   Graphics2D g2 = (Graphics2D)g;
g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,RenderingHints.VALUE_
ANTIALIAS_ON);
   int w = getWidth();
   int h = getHeight();
// Draw ordinate.
   g2.draw(new Line2D.Double(PAD, PAD, PAD, h-PAD));
// Draw abcissa.
   g2.draw(new Line2D.Double(PAD, h-PAD, w-PAD, h-PAD));
```

```
// Draw simple lines in x axis
   int numOfSections =data.length;
   for(int j=1;j<=numOfSections;j++)
    {
      g2.draw(new Line2D.Double(PAD+j*(w-2*PAD)/numOfSections , h-PAD+3,
PAD+j*(w-2*PAD)/numOfSections, h-PAD));
      int stt=j*10;
      String ss=""+stt;
      g2.drawString(ss, PAD+j*(w-2*PAD)/numOfSections-8 , h-PAD+3+12);
    }
// for Y axes
    for(int k=0;k<4;k++)
     {
       g2.draw(new Line2D.Double(PAD-3,+PAD+k*(h-2*PAD)/4,PAD,+PAD+k*(h-
2*PAD)/4));
       int x= getMax()- k*getMax()/4;
       String st=""+x;
       g2.drawString(st, PAD-22,+PAD+3+k*(h-2*PAD)/4);
     }
    g2.drawString(graphH, PAD,PAD-25);
    DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
    Calendar cal = Calendar.getInstance();
    System.out.println(dateFormat.format(cal.getTime()));
```

```
    g2.drawString("Time Line for "+dateFormat.format(cal.getTime()), PAD, h-
PAD+3+12+20);
    double xInc = (double)(w - 2*PAD)/(data.length);
    double scale = (double)(h - 2*PAD)/getMax();
// Mark data points.
    g2.setPaint(Color.red);
    ArrayList<Double> xs=new ArrayList<Double>();
    ArrayList<Double> ys=new ArrayList<Double>();
    xs.add(PAD+0.0);
    ys.add(h - PAD - scale*data[0]);
    for(int i = 0; i < data.length; i++)
     {
      double x = PAD + (i+1)*xInc;
      double y = h - PAD - scale*data[i];
      xs.add(x);
      ys.add(y);
      g2.fill(new Ellipse2D.Double(x-2, y-2, 4, 4));
     }
    for(int j=0;j<data.length;j++)
    g2.draw(new Line2D.Double(xs.get(j),ys.get(j),xs.get(j+1),ys.get(j+1)));
}
private int getMax()
{
```

```
  int max = -Integer.MAX_VALUE;
  for(int i = 0; i < data.length; i++) {
   if(data[i] > max)
       max = data[i];
                              }
  return max;
}
}
```

2) The Spot Code:

```
public class SensorSampler extends MIDlet {
   private static final int HOST_PORT = 67;
   private static final int SAMPLE_PERIOD = 10 * 1000;  // in milliseconds
   protected void startApp() throws MIDletStateChangeException {
      RadiogramConnection rCon = null;
      Datagram dg = null;
      String ourAddress = System.getProperty("IEEE_ADDRESS");
      ILightSensor lightSensor = (ILightSensor)Resources.lookup(ILightSensor.class);
      ITriColorLED led = (ITriColorLED)Resources.lookup(ITriColorLED.class,
"LED7");
      ITemperatureInput tempSensor = (ITemperatureInput)
Resources.lookup(ITemperatureInput.class);
      System.out.println("Starting sensor sampler application on " + ourAddress + " ...");
```

```
        // Listen for downloads/commands over USB connection
        new com.sun.spot.service.BootloaderListenerService().getInstance().start();
    try {
        // Open up a broadcast connection to the host port
        // where the 'on Desktop' portion of this demo is listening
        rCon = (RadiogramConnection) Connector.open("radiogram://broadcast:" +
HOST_PORT);
        dg = rCon.newDatagram(50);  // only sending 12 bytes of data
    } catch (Exception e) {
        System.err.println("Caught " + e + " in connection initialization.");
        notifyDestroyed();
    }
    while (true) {
        try {
            // Get the current time and sensor reading
            long now = System.currentTimeMillis();
            int reading = lightSensor.getAverageValue();
            double tempValue=tempSensor.getFahrenheit();
            // Flash an LED to indicate a sampling event
            led.setRGB(0, 255, 0);
            led.setOn();
            Utils.sleep(500);
            led.setOff();
```

```
                // Package the time and sensor reading into a radio datagram and send it.
                dg.reset();
                dg.writeLong(now);
                dg.writeInt(reading);
                dg.writeDouble(tempValue);
                rCon.send(dg);
                System.out.println("Light value = " + reading+" :  Temperature value =
"+tempValue);
                // Go to sleep to conserve battery
                Utils.sleep(SAMPLE_PERIOD - (System.currentTimeMillis() - now));
            } catch (Exception e) {
                System.err.println("Caught " + e + " while collecting/sending sensor sample.");
            }
        }
    }
    protected void pauseApp() {
        // This will never be called by the Squawk VM
    }
    protected void destroyApp(boolean arg0) throws MIDletStateChangeException {
        // Only called if startApp throws any exception other than
MIDletStateChangeException
    }}
```