RELIABILITY OF DATA COLLECTION AND TRANSMISSION IN WIRELESS SENSOR NETWORKS

by

Basheer Al-Qassab

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the

Electrical Engineering

Program

YOUNGSTOWN STATE UNIVERSITY

August, 2013

Reliability of Data Collection and Transmission in Wireless Sensor Networks

Basheer Al-Qassab

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:

Basheer A. Al-Qassab, Student Date

Approvals:

Dr. Frank X. Li, Thesis Advisor Date

Dr. Philip Munro, Committee Member Date

Dr. Faramarz Mossayebi, Committee Member Date

Dr. Salvatore Sanders, Associate Dean Graduate Studies and Research Date

Abstract

A network of wireless sensor nodes that are connected to a centralized base station is presented to conduct a study on reliability of data collection and transmission in wireless sensor networks (WSNs) with focus on data loss and data duplication. Software applications for specific sensor nodes called Sun SPOTs are presented, and programming techniques, for example packet transmitting time delay and data checking for loss and duplication, are implemented in these software applications to improve the functionality of the network. Acceleration data on a vibration plate are collected at sampling frequency of 100 Hz to validate the operation of the network. Additionally, the wireless sensor network is optimized to enhance the synchronization of data collection from different nodes. The result of this research shows that the reliability of the network is related to data sampling frequency, synchronization of the wireless data traffic, wireless sensor node signal strength, and wireless data routing protocols. The indoor tests on signal strength show the limitation of -70 dBm and higher for optimum data collection without data or packet loss.

Acknowledgements

I am very thankful for all the people who make this project to be done. First and foremost, I like to thank my advisor, Dr. Frank X. Li, for supporting me on the entire project time and guiding me through the development of the project. I would like to express my gratitude to my family and friends for their encouragement and support. I am also very thankful for all Higher Committee for Education Development (HCED) members in Iraq for giving me the opportunity to pursue my master degree in The United States of America.

Table of Contents

List of Figures

List of Tables

Abbreviations

ADC	Analog to Digital Convertor
API	Application Programming Interface
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
FFT	Fast Fourier Transformation
GUI	Graphical User Interface
JVM	Java Virtual Machine
LQRP	Link Quality Routing Protocol
LR-WPAN	Low Rate Wireless Personal Area network
MEMS	Micro-Electro-Mechanical System
OOP	Object Oriented Program
PRB	Packet Reply Byte
RF	Radio Frequency
RISC	Reduced Instruction Set Computer
SPI	Serial Peripheral Interface
SPOT	Small Programmable Object Technology

SPP	Samples Per Packet
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Chapter 1

Introduction

Over the last decade, the technology of Wireless Sensor Networks has been widely used in many applications. The reason behind the popularity of this technology is the integrity of sensing, processing, and communicating in one small wireless sensor node. Most wireless sensor nodes are capable of measuring temperature, acceleration, humidity, or even the level of gases or chemical materials in the environment. These measurements are converted to digital binary numbers by using analog to digital convertors, and then the sensor data are transferred through wireless communications to the hosted computer where the data are stored and processed. The challenge of the wireless sensor network technology is to make the sensor nodes small, power efficient, and inexpensive. In this thesis, the most important aspect of the wireless sensor network, the reliability of wireless communication, is discussed. Necessary applications have been developed and tested in order to study the data traffic between wireless sensor nodes. Many issues are raised concerning point to point communication of nodes, synchronization of data collection, and calibration of sensors.

1.1 Background

Wireless Sensor Networks (WSNs) have been used in many real life applications, including medical, habitat monitoring, environment observation system, transportation,

industrial and control, smart home monitoring, office automation, and entertainment applications. In addition, this technology becomes even more attractive in military and other risk-associated applications. WSNs technology has become high priority in the scientific community. The use of these sensors and the possibility of organizing them into networks have revealed many research challenges and have highlighted new ways to cope with certain problems such as mobility and coverage problems of sensor nodes in some applications implemented WSNs [1].

Our interest leans toward developing a WSN for data collection to study the reliability of data communication between sensor nodes and a centralized base station. A robust wireless communication in any WSN application requires suitable choice of wireless network topology. It also requires some reliable protocol to be used for data routing and transmission. The implementation of this technology has advantages over the traditional methods, such as wired sensor nodes, in many applications. For instance, the installation process of the wireless nodes is much easier than wired nodes, and it requires less installation time and fewer crew members.

Usually the wireless sensor networks consist of a large number of sensor nodes due to the nature of some applications that require dense implementation of sensor nodes. This large number of sensor nodes increases the risk of data transmission problems such as data loss, data duplication, and unsynchronized data collection. Many published papers and research have been targeted the topics related to performance and reliability of WSNs. Many of them have proposed or developed new algorithms and approaches to increase the reliability of data collection and transmission. For instance, Junsuk Shin et al., [2] published a paper on reliability of packet delivery for all-to-one communication

pattern in dense WSNs. Also, this paper suggests how to enhance the well-known protocols that help boost the performance to an acceptable level.

Another significant problem is the interference. Since the selected sensor nodes for this research and most of other sensor nodes use the license-free 2.4-GHz frequency band, they are always vulnerable to interference with other RF signal sources that use the same frequency band, such as Bluetooth, Wi-Fi, and microwave oven. Most of the time, WSNs have to be operated in a very harsh environment. Some research papers have been done concerning the interference problem. For example, Wenqi Guo, et al., [3] published a paper that investigate the impact of interference sources on the performance of IEEE 802.15.4 WSNs. Several experiments have been performed in different scenarios. The results of this paper indicate a rough mutual interference among systems that operate in the same band. These published papers showed how the designers should plan in order to minimize these problems.

1.2 Basics of the Research

The basic of this research is to develop software applications for the wireless sensor nodes in order to investigate the data collection and transmission. These wireless sensor nodes come with many different built-in sensors. The embedded accelerometer sensors have been used as sources of data generation in the nodes. These sensors collect the acceleration data on a vibrated plate at a sampling frequency of 100 Hz. The sensor data have been time stamped in order to study the synchronization of data coming from different sensor nodes during the data collection. This time stamping can also be used to detect the packet or data loss in the collected data, hence the reliability of collected data. Sensor nodes are programmed to collect the data periodically and send them by the

wireless connections to a centralized node called base station. The base station is a special type of wireless sensor node that is connected to a hosted computer through a USB connection. On the hosted computer, a software application is developed to communicate with active wireless sensor nodes. This application gives instructions to all sensor nodes and receives the periodically collected data.

The sensor nodes that are used to be studied in this research are from the Oracle Inc. previously, Sun Microsystems. They can be programmed entirely by using Java programming language. Chapter 2 will be dedicated to the architecture of WSNs.

1.3 Organization

This thesis gives a detailed description about hardware, developed software architecture, and reliability verification of sensor nodes.

- The first chapter starts with an introduction to research and the idea behind it. It also covers some background about the WSNs and the previous research that has been done.
- The second chapter is about the sensor architecture and the hardware description of the Sun SPOTs.
- Chapter three explains the topics related to wireless networking and communication protocols.
- The fourth chapter is mostly dedicated to the software development, and it explains in detail how the underlying software operates on these wireless sensors. This chapter also covers some of the packet traffic issues and shows how these issues have been solved programmatically.

- Chapter five covers the operation of the software and verification of these sensor nodes in a laboratory environment. This chapter also concludes the reliability of the WSNs.
- The last chapter in this thesis is dedicated to the conclusion and the recommendations for future research.

Chapter 2

Sensor Node Architecture

The wireless sensor nodes that are used for this research are called Sun SPOT (Small Programmable Object Technology) [4]. This chapter is dedicated to explain the hardware infrastructure of these particular sensor nodes.

2.1 Brief Description of Sun SPOT

The Sun SPOT is a small wireless computing platform that runs Java applications. The system comes with an on-board set of sensors and I/O pins for easy connection to external devices [5]. In addition, the accessibility of hardware resources is quite simple by using the java Application Programming Interface (API) library.

The devices come as a kit, and each kit contains one base station Sun SPOT unit and two free-range Sun SPOT units. The purpose of the base station unit is to be connected to the host computer via USB cable. This base station unit gives the host computer the ability to communicate with free-range units over the radio connection. The free-range Sun SPOT units are designed to be standalone units which are mobile devices without any wired connection. The free-range units have their own rechargeable batteries, and they have the same exact main board as the base station unit. This main board hosts the basic components including the main processor, memory components, power management circuitry, and the network radio transceiver. As shown in Figure 1, the free-

range SPOT comes with an application board, which has all the sensors, LEDs, push switches, and some I/O interface pins for further expansion.

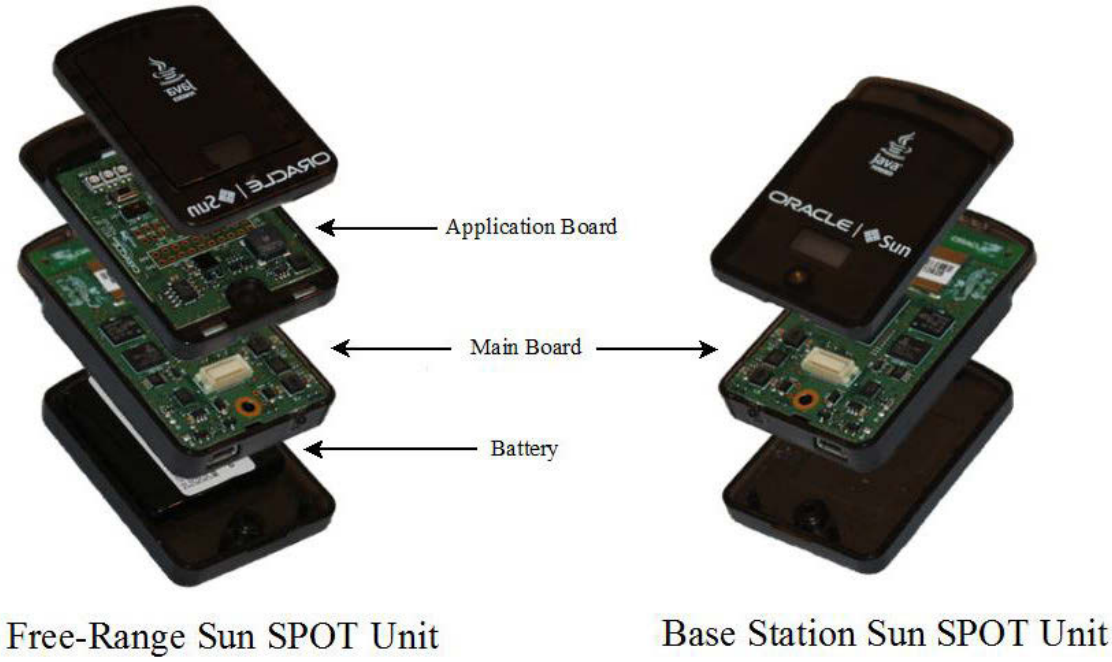

Figure 1 Free-Range and Base Station Sun SPOT Units

2.2 Main Board Components

In this section, the technical specifications of the main board and its components are discussed in detail.

2.2.1 Processor and Memory

The SPOT's main board hosts a 32-bit 400 MHz ARM9 processor running a Java VM, 2.4 GHz network radio transceiver and power management circuitry [4]. The processor is AT91SAM9G20 from Atmel. The maximum clock frequency is 400 MHz. This processor runs with a core voltage of 1 volt. The IO voltage is at 3 volts. The processor builds on the ARM architecture which uses the Reduced Instruction Set

Computer (RISC). There are two internal 16 kbit SRAM memories. The 133 MHz 16-bit external data bus is used to connect the processor to 8 MByte of Flash memory that is organized as (4 Mbit x 16 bit) and 1 MByte of SRAM that is organized as (512 kbit x 16bit). The power on the flash memory shuts down when the SPOT is in deep sleep mode to save power. However, the SRAM is always on even during the deep sleep mode. This gives the SPOT the ability to wake up quickly and resumes the normal running condition. The flash memory has 128 Bytes, which has been used to store 64-bit IEEE EUI extended unique identifier by the factory. The IEEE EUI makes the SPOT identifiable with a unique number as they connect to base station.

2.2.2 Power Management Circuitry

The Free-range SPOT is powered by an external battery with 770 mAh capacity and an output voltage of 3.7 volts, but it can also be powered by USB power or externally connected power supply. If the SPOT is supplied 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 on 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. The active mode is when the CPU is busy and processing the running threads. For this research, the deep sleep mode has been deactivated in order to respond to base station commands quickly.

2.2.3 SPOT Network Circuitry

The SPOTs' wireless network is based on the IEEE 802.15.4 standard. This standard and all other IEEE 802.15 standards are categorized under Wireless Personal Area Network (WPAN). The WPAN is meant to convey information among the wireless devices that are within a small area of coverage, usually a few centimeters to tens of meters in diameter. The IEEE 802.15.4 standard targets a wireless device that requires a simple design complexity, ultra-low cost, ultra-low power consumption, and low data rate wireless connectivity. The WSNs are usually required to have these same specifications as in the IEEE 802.15.4 standard. In this thesis, chapter 3 has been dedicated to WSNs and Radio communication standards.

The wireless transceiver chip that controls the network communication of the SPOTs is on the main board. The chip model is CC2420, which operates on the frequency band of 2.4 to 2.4835 GHz ISM unlicensed band. This band is divided into 16 channels which are named channel 11 to channel 26. Each channel has its own central frequency as shown in Table 1. The effective bit rate for this chip is 250 kbps. This chip can be adjusted to transmit the signal in different power levels. The CC2420 chip can be set to 22 different power levels.

Table 1 Channels and Central Frequencies

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

This chip is connected to inverted-F antenna, which is printed on the top layer of the main board. The dimension of this antenna has been optimized by manufacturer to operate at 2.4 GHz frequency band, and it has characteristic impedance of 115 ohm.

2.3 Application Board Components

The application board comes only with the free-range SPOTs. It stacks on the top of the main board through the Hirose DF17-30 connector.

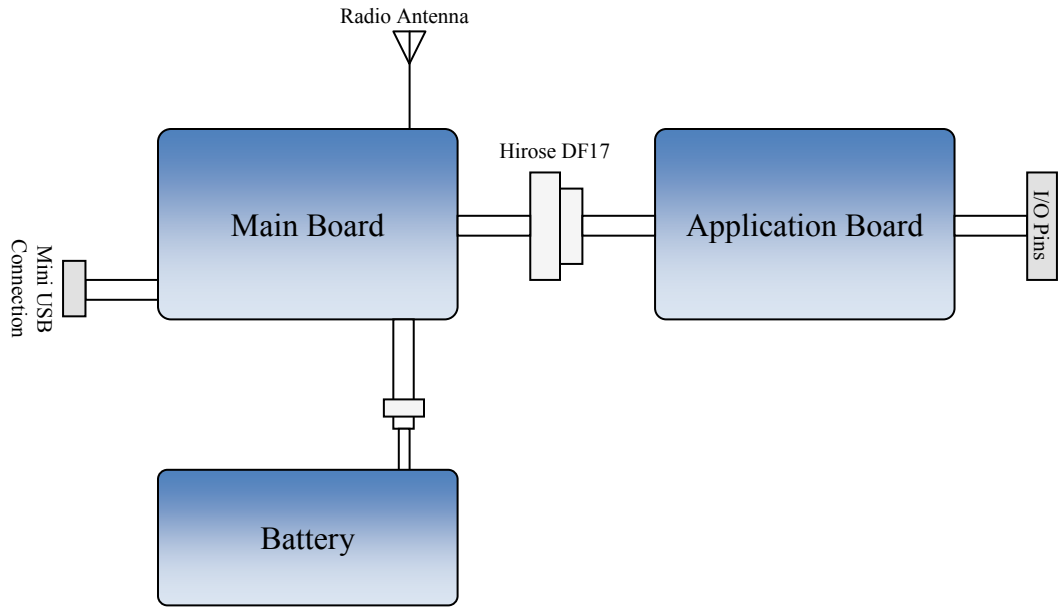

Figure 2 Free-Range Sun SPOT Hardware Architecture

The communication between the two boards is performed by using the SPI (Serial Peripheral Interface). The SPI is a synchronous serial data link standard where the main board works as a master device and can communicate with one or more slave devices, which is the application board in free-range SPOT [6].

The application board hosts a good number of sensors and input/output interfaces, including a three-axis accelerometer, RGB color sensor, thermometer, infrared

emitter/detector, 2 push buttons, small speaker, and 8 RGB LEDs. Also, it has some external I/O interface that can be used to connect external sensors or devices for further expansion.

The accelerometer that is mounted on the top of application board is a Micro-Electro-Mechanical System, commonly referred to as MEMS technology. Figure 3 shows the location of accelerometer and other sensors on the application board. This transducer or sensor can sense static force like the gravity of earth or the dynamic acceleration force like exposing the sensor to some movement.

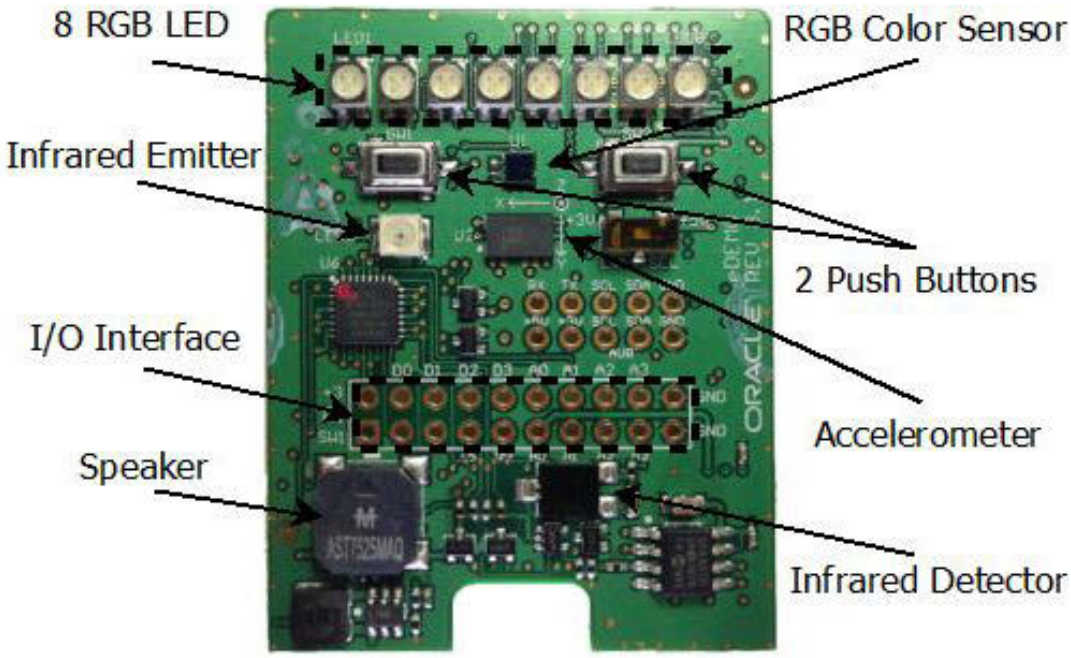

Figure 3 Top-View of Application Board

The accelerometer can measure the acceleration in three axes X, Y, and Z. As indicated in Figure 4, the acceleration value in a considered axis will increase if the free-range SPOT moves toward the plus (+) sign and decrease if it moves toward minus (-) sign. If the SPOT is set stationary on the top of a flat area, the Z axis acceleration will indicate +1g and the X and Y axes will indicate 0g. This happens because the gravity of the earth is 1g (9.8 m/s^2).

Figure 4 Accelerometer Dimensions on the Free-Range SPOT

The acceleration can be used in different scales ±2 g, ±4 g, and ±8 g. These scales can be changed through programming. Each scale has a different sensitivity. The sensitivity can be defined as the minimum change in the MEMS sensor that affects change of the less significant bit of the digital output. Since the output of ADC (Analog to Digital Convertor) uses 8-bit to represent one digital measurement when the accelerometer is in the ±2 g and ±4 g ranges, the sensitivity of accelerometer is 64 LSB/g and 32 LSB/g, respectively. That gives us (≈0.0156 g minimum difference in change /

change in less significant bit) for 64 LSB/g and (≈0.031 g minimum difference in change / change in less significant bit) for 32 LSB/g. In the case of the ±8 g scale range, the output of ADC uses 10-bit to represent digital measurements. This representation of data gives the sensitivity of 64 LSB/g, which is the same as in ±2 g scale.

Chapter 3

Networking and Radio Communication

In this chapter, the general concepts of Wireless Sensor Networks (WSNs) are presented. The topics covered in this chapter are related to the operation of network infrastructure, the radio communication between nodes, and the topology of wireless sensor networks.

3.1 WPAN and LR-WPAN Standards

Wireless personal area networks (WPANs) are used to convey information over relatively short distances. Unlike wireless local area networks (WLANs), the network connections via WPANs use little or no infrastructure. This feature allows small, power-efficient, inexpensive solutions to be implemented in a wide range of devices [7]. There are many commercial WPAN technologies available. Some well-known examples are 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.

The Sun SPOT is based on the IEEE 802.15.4 standard or LR-WPAN. The LR-WPAN stands for Low Rate Wireless Personal Area network. 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 simple in

structure and designated to do a specific task with limited low-rate data transportation. This standard also preserves the simplicity and flexibility of the communication protocols. The maximum data rate for this standard is 250kbits/s, which is very slow compared to other standards. However, it is sufficient for applications that require very limited data transmission.

3.2 Network Layers Architecture

To simplify the understanding of the sensor network architecture, the operation of the network is separated into different layers. Each layer has some specific tasks to manipulate the data packets. These layers can be divided into three general layers as shown in Figure 5. The following explains how the data packets get processed in these layers.

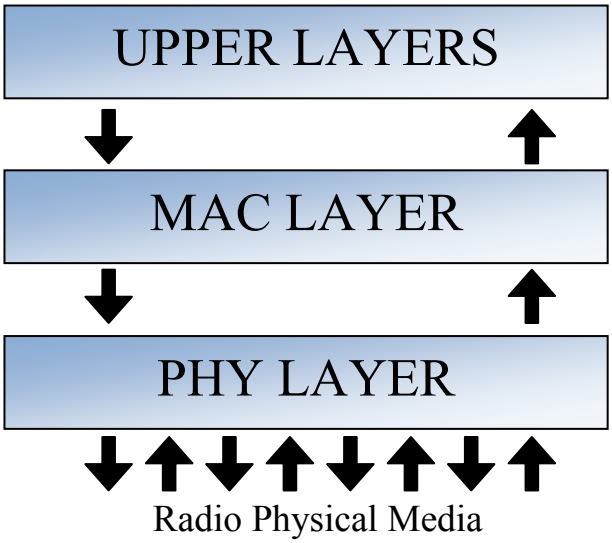

Figure 5 LR-WPAN Network Layers Architecture

The physical layer is the bottom layer. The responsibility of this layer is to manage how to place data on the radio physical medium or the wireless medium. It also has to be able to receive the data by sensing electromagnetic signals in the air. Sometime this layer is considered as the actual transceiver hardware. The other physical layer responsibilities are activation and deactivation of radio transceiver, radio channel selection, power setting for the current channel, and Link Quality Indicator (LQI) for received packets.

The layer above the physical layer is MAC (Media Access Layer) layer. This layer is responsible for providing a reliable link between two peered sensor nodes. In this layer, the data are encapsulated with some information to the header of each packet such as the packet sequence number, the source and destination MAC addresses, and also the PAN Identifier of the current device. This layer is also responsible for generating the acknowledge packet that confirms the reception of the data packet. Another responsibility of this layer is to manage the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism. This mechanism is very important in order to give the nodes the ability to send the packet on the shared access media [8]. The basic idea of CSMA/CA is to give the node the ability of sensing the availability of the physical medium. Whenever the node wants to use the physical medium, it will sense the physical medium first to check for the availability. If the physical medium is idle, the node will use it to transmit packets. However, if it is busy, the node will back off for a random time, and it will try to send the packets again after the random back off time is over. If the node finds the physical medium is still busy for the second time, it will wait for another random period of time. This process will be continuous until the packet is successfully

sent, or it will terminate with a channel access failure after reaching a limited number of tries [7].

Since the IEEE 802.15.4 standard only covers the definition of physical and MAC layer, the top layers of the network architecture model may vary from one sensor node to another from a different company. In the following section, the Sun SPOT protocols in the upper layers are described in more details.

3.3 Radiostream and Radiogram Protocols

The Sun SPOT has two protocols in the upper layers which are *radiostream* and *radiogram*. In order to make a data communication between two SPOTs, any of these protocols can be used. The *radiostream* protocol is used in peer-to-peer buffered stream-based communication where the two SPOTs can open radio communication with a destination MAC address and a desired port number that ranges from 0 to 255. Whenever 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.

The *radiogram* protocol is a datagram protocol which allows the exchange of data packets between two devices. This protocol can be operated as point-to-point or client-server communication model. As in the *radiostream* protocol, if the point-to-point communication model is needed, 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. On the other hand, if the requirement is to have a 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 for any communications. 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. Concerning the hop counts, in the broadcast packets, the hop counts can be set during the process of opening the radio communication. In that way, the broadcast packets can be controlled in terms of how far they can be reached.

In both protocols, there are acknowledging packets that are exchanged among the nodes to inform senders the reception of received packets. These packets are generated by the MAC layer, and they provide reliability to the protocols. Also, it has to be mentioned that whenever radio communication is opened, the SPOT uses the current radio channel and current PAN identifier that have been set in the properties of the SPOT. One can easily change the radio channel and PAN identifier by accessing the radio property of the SPOT. This can be done either programmatically or by changing the properties of SPOT using the Sun SPOTManager program that comes with the kit.

3.4 Network Topology

The network topology can be defined as a schematic layout of communication paths as they connect one node to another. These communication paths can be arranged in different layouts to represent the network topology. For instance, star, mesh, and tree are typical types of network topology as shown in Figure 6. The layout of physical node connections does not have to represent the actual logical layout of the network. The physical connection layout is usually called physical topology, which can be considered

as how the network is physically connected. However, the important topology is the logical topology, which is the layout of actual data path communications between nodes.

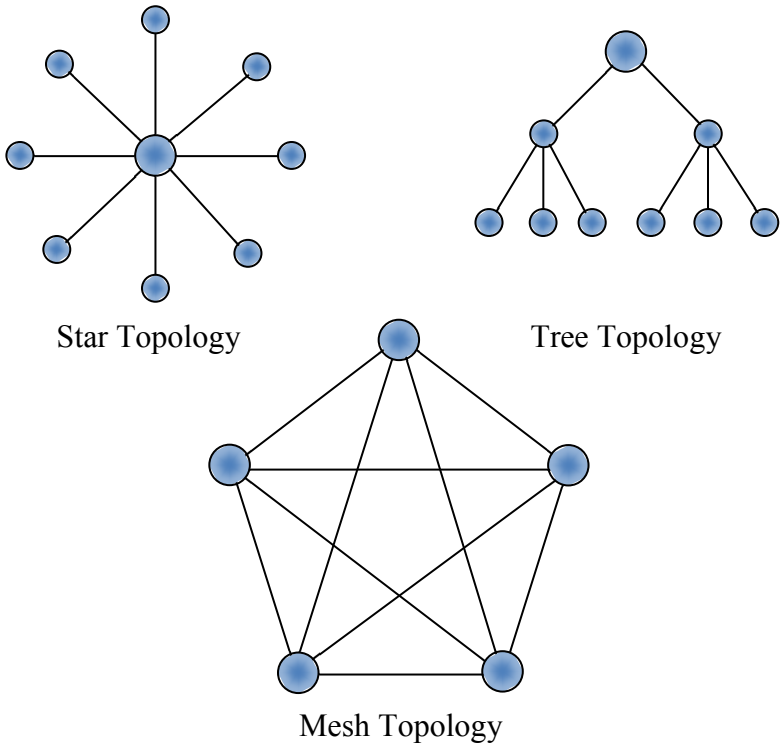

Figure 6 Network Topologies

In this research, the base station is responsible for connecting with all SPOTs, and it works as a centralized node. The logical topology can be considered as a star topology. However, after the base station is connected with free-range SPOTs, a dedicated point-to-point communication will be opened with each SPOT, and instructions can be sent through these point-to-point channel communication.

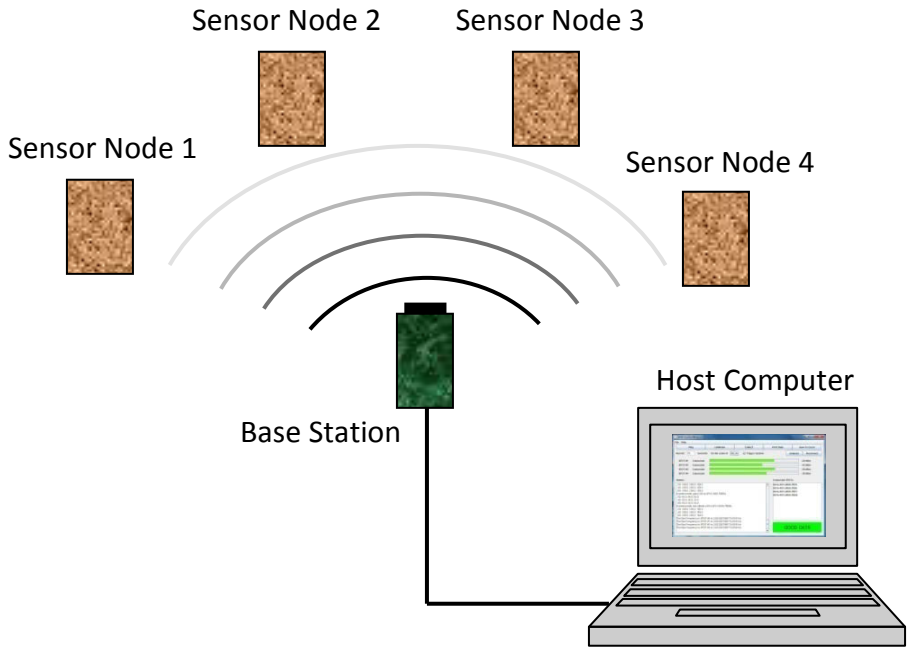

Figure 7 Network Topology of WSN

Figure 7 shows the topology of WSN for this research. This topology only shows 4 sensor nodes. However, the hosted computer application has been programmed to be flexible with number of connected free-range SPOTs. The number of point-to-point communication depends on the number of connected SPOTs. Since we have tested the developed network on 4 nodes, 4 point-to-point communication channels have been opened.

3.5 Routing Protocol

Routing protocol is one of the important aspects in WSNs. Most of the time, the coverage area of the entire network is much bigger than the coverage area of a single sensor node. As a result, there will be multi-hope packet traffic in the network. To manage this type of communication traffic, there should be a routing protocol that

manages data transmission between source and destination over multi intermediate sensor nodes.

To manage this traffic routing, the sun SPOT sensor nodes use the LQRPManager, which is a built-in software that takes care of Link Quality Routing Protocol (LQRP). This software enables the SPOT to act as a mesh router. Whenever the running application begins to open a radiostream or radiogram connection, the LQRPManager will start to send and receive LQRP packets. The actual operation of LQRP is based on an algorithm which attempts to determine the best route. In this algorithm, route requests (RREQs) for a particular targeted SPOT are broadcast by a requester and then re-broadcast by any Sun SPOT that receives them. Each Sun SPOT that knows how to reach the requested target sends a reply back to the requester. The route that will be used is the one with the best link quality [9].

In this thesis, the sensor nodes are not too far from the base station, and this makes the base station signal cover all 4 sensor nodes. As a result, the nodes can reach the base station using one hop. The routing protocol will always determine that the destination address is one hop from the source. Therefore, whenever the sensor node wants to send a packet to the base station, it will directly send the packet to the base station's address without sending the packet to any intermediate sensor nodes. This helps the system to work more efficiently in the sense that the sensors will only dedicate its resources to take the acceleration measurements and send them to the base station without wasting the node hardware resources to process other traffic.

Chapter 4

Software Description and Development

In this chapter, an overview of the software structure of the Sun SPOT will be explained and then a complete description of the developed applications will be given. There are two main applications that are involved in this project. These applications are the host computer application and the free-range SPOT application. This chapter will also cover packet structure description, which will help in understanding how the data are placed inside the packets in order to be recognizable by all applications. In addition, the chapter covers synchronization issue and provides a practical solution for this issue. Finally, we will go through the process of calibration and explain how this process is done step by step.

4.1 Sun SPOT Software Architecture

The Sun SPOT applications can be completely developed by using the java programming language, which is designed to be a machine-independent programming language that is both safe enough to transverse networks and powerful enough to replace native executable code [10].

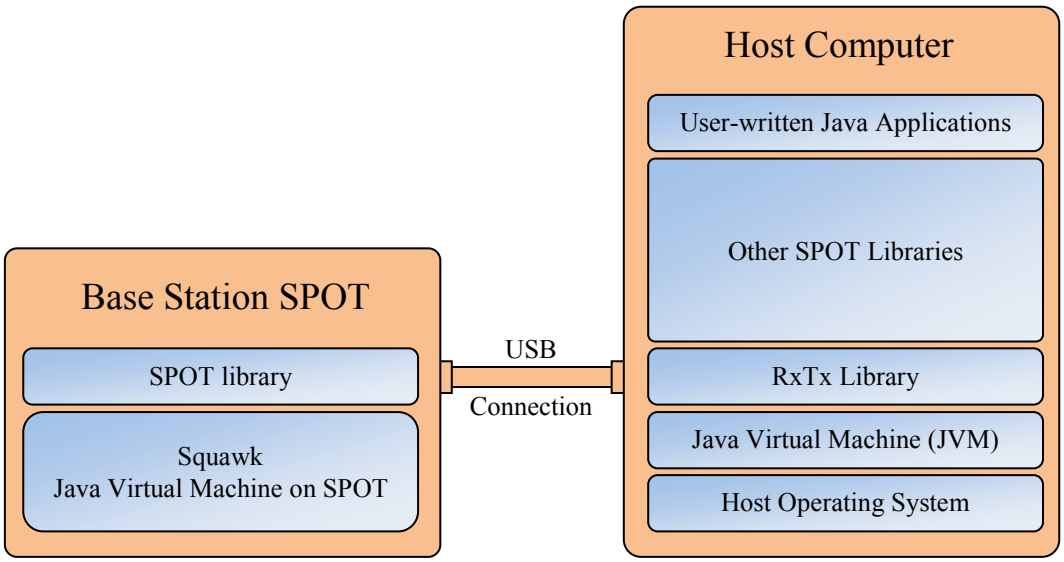

Figure 8 Host Application Architecture

Figure 8 shows the host application architecture on both the host computer side and base station side. To communicate with the base station, the *RxTx Library* is used to perform the serial I/O over the USB connection [9]. It is also obvious from the figure that the *User-written Java Applications* layer, which is the only part that can be written by the host application developer, runs on the host computer. The base station is an intermediate device that provides the host application with radio access to communicate with free range SPOTs.

On the free-range SPOT side, the software architecture is different. It is also developed under different java platform, Java ME (Micro Edition). This type of platform is used to develop java application for embedded system devices. In Sun SPOT sensors, there is a special kind of java virtual machine that runs directly on the processor called Squawk JVM. The Squawk JVM is the result of an effort to write a Connected Limited Device Configuration (CLDC) compliant JVM in Java which also provides OS level mechanisms for small devices [11].

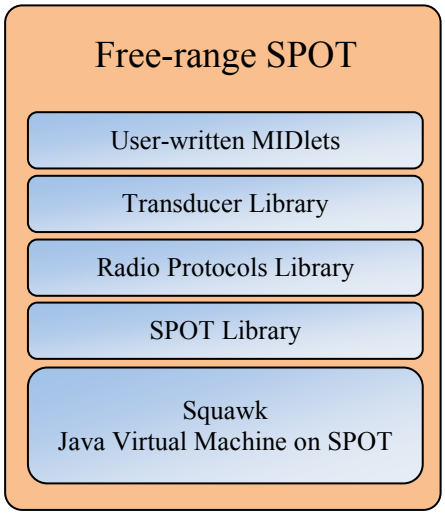

Figure 9 Free-Range SPOT Application Architecture

Figure 9 shows the software stack model of the free-range SPOT. In this diagram, the different java libraries that are used in developing the free-rage SPOT application are shown. All of these libraries run on the Squawk JVM. Above the Squawk, there is *SPOT Library*, which can be used to access the SPOT device and basic I/O. It can also be used to access the low-level MAC radio protocols. The *Radio Protocols Library* is used to access the high-level radio protocols like Radiogram and Radiostream. The *Transducer Library* can be used to access the hardware resources on the application board of the free-range SPOT like accelerometers, LEDs, switches, etc. On the top of all these libraries, the *User-written MIDlets* can be defined. MIDlets are Java ME platform applications that are extended as MIDlet class. It is the software developer job to write these classes for the free-range SPOT applications [9].

To develop an application in Java SE platform, the application should include a static method named *main()*. When the program runs after compiling, it starts executing the statements that are defined in this method. However, to develop a MIDlet application

as Java ME platform, there are two main points that should be considered. First, the main class should be extended as MIDlet class. The main class is the class from which the application starts. Second, the main class has to have three static methods, which are *startApp()*, *pauseApp()*, and *destroyApp()*. When the program runs, the *startApp()*executes its statements first. It is equivalent to the *main()* method in the Java SE platform applications.

4.2 Host Computer Application

In this section of this chapter, a comprehensive description about the host computer application will be explained. This explanation will be more related to the underlying software classes that interact with free-range SPOT application and compose the host computer application. Before going through the explanation of each of these application classes, the main tasks of the host application are briefly listed below:

- Listen to free-range SPOT requests to make a new connection.
- Spawn a new thread for each free-range SPOT successfully connected to base station.
- Update the RF signal strength bars.
- Ping the free-rang SPOT to check for the connectivity.
- Calibrate the free-range SPOTs.
- Start and stop collection of data from connected free-rang SPOTs.
- Set the time for data collection.
- Save the collected data in an excel file.
- Perform FFT on collected data.

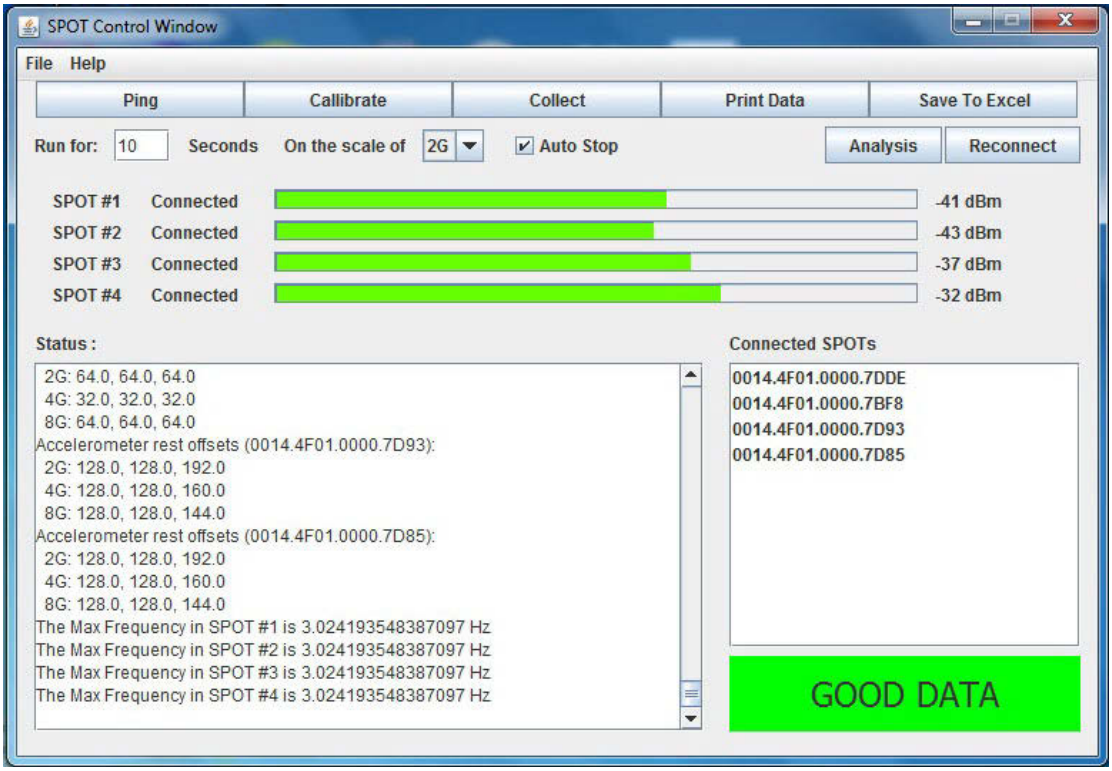

Figure 10 Main Window GUI of Host Computer Application

Figure 10 shows the main window GUI. This application has been developed as regular Java SE. The application itself consists of three core java classes which are *SpotListener*, *MainWindow*, and *AccelerometerListener* as well as one interface, which is *PacketTypes*. It also consists of some other utility classes such as *SpotData* and *ReadExcel*. These classes are just simple text files written in java language.

The main class in this application is *SpotListener*, which has the *main()* static method. This class is responsible for setting up some initial steps. First, it instantiates an object from the *MainWindow* class and sets the visibility of this window to true. Then, it sets up broadcast radiogram connections. The radiogram is a two-way connection. The first broadcast radiogram connection is a transmission connection which is used to send out a broadcast request to all Sun SPOT in the range of base station connection. This

broadcast connection is opened on the port 43. The second radiogram connection is reception, which is used to listen to any requests from new free-range SPOTs. This connection is opened on the port 42. Whenever the base station receives a connection request from any free-rang SPOT that wants to open a connection, the base station will instantiate a new object from *AccelerometerListener* class. This object will be a new thread that will take care of any future requests from this particular free-range SPOT.

MainWidow class is designated to work as the code behind the *SPOT Control Window*. See Figure 10. It carries all pieces of code that will get executed when any button gets pushed on this window by the user. This class is instantiated by the *SpotListener* as a new object at the beginning of the initialization process. Java swing package has been included in this class in order to build a Graphical User Interface (GUI). If any button that is responsible for giving instruction to free-range SPOT such as *ping*, *calibrate*, and *collect* gets pushed, the instruction code does not get executed in this class. However, for a more efficiently working application, the command will instead be given to the threads that are responsible forgiving instructions to the connected free-rang SPOTs through the point-to-point radio connections.

The third class *AccelerometerListener* is instantiated multiple times based on the number of free-range SPOT that gets connected to base station. This class is extended as thread. As a result, it has to implement a function called *run()*. The *run()* function is the starting point of execution for the thread. Whenever a new free-range SPOT wants to connect to the base station, the *SpotListene* will instantiate a new object of *AccelerometerListener* class, and that will be a new thread responsible for giving commands to this new particular free-range SPOT. The radio communication between

AccelerometerListener thread object and the free-range SPOT will be point-to-point communication. That means, whenever the base station wants to send a packet, it uses the free-range SPOT MAC address as the destination address, and free-range SPOT uses the base station MAC address as the destination address to communicate.

The *PacketTypes* is a java interface. The interfaces in java language program are written to define the behaviors that classes implement, and they only contain abstract methods and the declaration of constants [10]. The *PacketTypes* interface does not implement any methods or functions. Instead, it is used to define constants that represent the packet types and port numbers. This is a common programmatic practice that helps the development of the software. The purpose of this interface is to define port numbers used for opening the communications and to define all requests and reply commands used for labeling the packets of different types. This interface has been implemented in other classes. As a result, constant names can be called instead of numbers.

The other classes are utility classes that can be used to do some specific tasks. The *SpotData* can be used to represent the data of one SOPT in the program as an object. This class has the ability to perform the FFT on the data by using *DoFFT()* function that is programmed inside this class. Finally, the class *ReadExcel* can be used to read a specific Excel file and return an array of data for a specific column in this Excel file. This class is useful for reading the data from Excel files that are saved on the computer during the data collection.

4.3 Free-Range SPOT Application

Free-range SPOT application runs on the wirelessly connected sensor nodes. All 4 sensor nodes are deployed with identical free-range SPOT applications. The Squawk, which is the Java Virtual Machine of the SPOT, operates this application directly on the processor. The following briefly lists all the tasks that this application is responsible for:

- Sending out a periodical broadcast request to search for the base station and waiting for reply.
- Replying back to the ping command from the base station.
- Replying back calibration information to help the host application to calculate the calibration of the free-range SPOT.
- Collecting periodical measurements of the accelerometer and sending them to the base station.

The free-range SPOT application has been organized into two packages. A package in java language is a namespace that is used to collect the related classes and interfaces together. The packages can be thought of folders that are used to organize files on computer. The main program classes have been placed in one package while the other package contains several utility or helper classes and interfaces. These utility classes and interfaces are working as simple tools that help the main classes to use them whenever they need. Down below is a table that contains the name of utility classes and interfaces and their job in the program.

Table 2 Utility Package Classes and their Task

Class or Interface Name	Class or Interface job
LocateService	Helper class to handle locating a remote service. It broadcasts a service request periodically and listens for a response. Whenever located, it calls back to report the IEEE address of the located service.
LocateServiceListener	Interface used by a class wanting to be called back when the requested service has been found. This interface has to be implemented in the class that wants to use this call back or event.
PacketHandler	Interface used by a class wanting to be called back whenever packets are received with a packet type. The contents of the first byte of the packet determine its type. This interface has to be implemented in the class that wants to know about the receipt of packets.
PacketReceiver	Simple helper class to monitor a radiogram connection and redirect packets to handlers, classes that have implemented the PacketHandler interface and registered an interest in some packet types.
PacketTransmitter	Simple transmit loop to pull packets off of a transmit queue and send them.
PeriodicTask	A class that provides for running a task on the SPOT, such as taking samples of accelerometer at a regular interval.

The main class is extended as MIDlet, and it is named *TelemetryMain*. In developing Java ME platform applications, as long as the main class is extended as MIDlet, it will be mandatory to implement the three *startApp()*, *pauseApp()*, and *destroyApp()* methods. The *startApp()* method is the starting point of the application. It is similar to the *main()* method in the Java SE platform applications. If the application gets paused by the Squawk JVM for some reason, the method *pauseApp()* will be called. The *destroyApp()* is called when the application wants to exit. This method can be used to release resources that are used by the application. This class also implements the utility

interfaces *LocateServiceListener* and *PacketHandler* to handle the response of connection requests that are used to acquire a base station and to handle the packets that will get received after the connection is completed with the base station. This class opens a broadcast radiogram communication to discover base station. Whenever a base station is connected, it will open a point-to-point radiogram communication with the base station. This class also takes care of ping response that may be sent by the user operating the host computer application. Ping response carries some information about the current status of the SPOT such as signal strength and the battery level.

The *AccelMonitor* class has been programmed to read the accelerometer measurements regularly and send them over the point-to-point radiogram communication. It can handle the packets that are sent to set the scale of accelerometer between 2, 4, and 8 g to accommodate the maximum vibration. The maximum vibration is the limit of acceleration without clipping. Also, it can handle the packets that request the current accelerometer scale and send back the response to the base station. In addition, the *accelMonitor* contributes to the calibration process. Whenever there is a request to send accelerometer calibration information, the class sends packet carrying calibration information that can be helpful on the host computer application to determine zero level of the sensor. More information about calibration process will be given in a later section of this chapter.

As in the host application program, the free-range SPOT application has the *PacketTypes* interface that is used to define all constant names. These constants represent port numbers of radiogram communications. Also, they represent request and reply byte values of packets. When any class wants to use the constants, the interface has to be

implemented inside the class. For example, the *TelemetryMain* and *AccelMonitor* classes implement this interface.

4.4 Packets Structure

Understanding the structure of the packets is very important because it tells us how packets are addressed in order to reach the desired destination. More importantly, it can tell us how we can place information that has to be sent over the radiogram communications. In this section and the next one, we will talk about the packet header information and also the payload data that are used to carry the acceleration and other information as the packets gets exchanged between the network nodes.

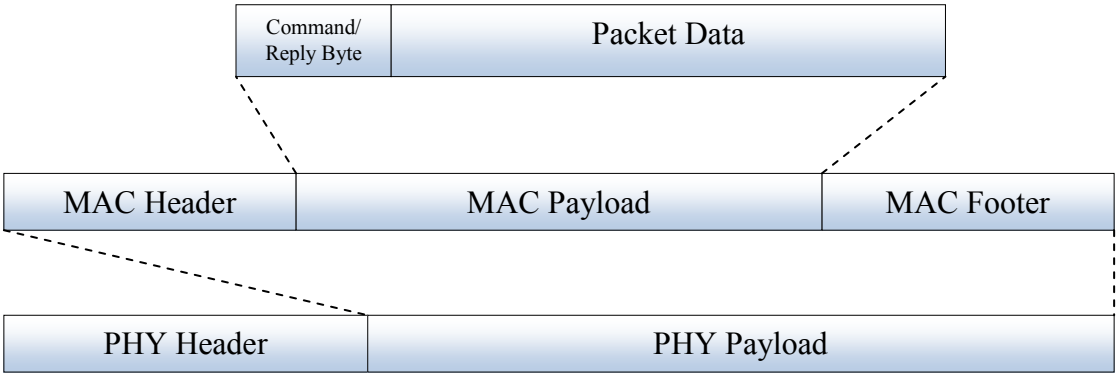

Figure 11 General Format of Packets

Figure 11 shows the general format of the packet in different layers of the network architecture model as mentioned in the previous chapter. These layers are Physical Layer, MAC Layer, and Upper Layers. The packet in the physical layer consists of physical header and physical payload. Physical payload is the whole packet as it is in the MAC Layer. While in MAC Layer, the packet consists of MAC header, MAC footer, and MAC payload. The source and the destination MAC addresses are placed in the

MAC header. This header also contains the frame sequence number and other frame controls. The MAC Footer contains the Frame Check Sequence (FCS) which is used by the receiver to perform the checksum. The checksum is a method of error detection that tells us if there is any error in the packet upon receiving it. The last part of packet in this Layer is the MAC payload, which is the packet as it is in the upper layers.

The data that we want to transfer can be carried inside the MAC Payload. In order to send different types of packets, the first byte of the data is used to distinguish between different types of packets. Whenever the packet is received by either the host computer application or the free-range application, the application will check the first byte. Based on that byte, it will manipulate the rest of the data to get the correct information from the packet. In next section, the acceleration packet type, which holds the acceleration data from the free-range SPOT to host computer application, will be discussed in details.

Table 3 shows command and reply byte values and the meaning of these values. One can generally separate these values into two categories. The first category is command packets, and they are only one byte packet size. This type of packet is used to request information. Both of the host computer application and the free-range SPOT application can request information from each other. The second category is the reply packets. They may be different in packet size. These types of packets are the answer to the command packets. Always the first byte is the type of reply packet followed by the information or data that are requested.

Table 3 Command and Reply Byte Values and their Meaning

Command/Reply Byte Value	Packet meaning
1	Remote SPOT command to locate a display server.
2	Host command to indicate it is restarting.
4	Host command to request the current accelerometer scale and calibration information.
5	Host command to specify the accelerometer scale to be used.
6	Host command to request the accelerometer to be calibrated.
7	Host command to request accelerometer data to be sent.
8	Host command to request accelerometer data to be stopped.
9	Host command to ping the remote SPOT and get the radio signal strength.
101	Host reply to indicate that it is available.
104	Remote SPOT reply to indicate the current accelerometer scale and zero offsets for the calibration process.
106	Remote SPOT reply to indicate the current accelerometer scale being used.
107	Remote SPOT reply to indicate the current accelerometer rest offsets.
108	Remote SPOT reply with current accelerometer readings taken using the 2G scale.
110	Remote SPOT reply to a ping includes the radio signal strength and battery level.
112	Remote SPOT reply with current accelerometer readings taken using the 4G scale.
113	Remote SPOT reply with current accelerometer readings taken using the 8G scale.

4.5 Data Transmission and Synchronization

The accelerometer data samples have to be transmitted by packets. In each packet, there are 16 accelerometer data samples stamped with time. The internal clock of the free-range SPOT is used for this time stamping. Each sample needs, however, 8 bytes to write the whole time. A better way to reduce the amount of bytes is to write the time

stamp that represents the time of taking the first sample. This time stamp is then placed in the beginning of the packet after the Packet Reply Byte (PRB), and the other samples use time offset. The packet time stamp follows one byte that represents the number of Samples per Packet (SPP) which in our case is 16. SSP byte is helpful for host computer application to determine the number of samples per each packet. To time stamp the other samples, we can simply use the time offset that is determined based on the packet time stamp, which is basically the time stamp of the first sample. In this way, two bytes will be enough to represent the time stamp of each sample. We have also used two bytes to write each raw acceleration sample for the Z axis. As a total each packet is 74 bytes in length. When the packet carries accelerometer samples, the PRB can be 108, 112, or 113 based on which acceleration scale has been used 2G, 4G, or 8G, respectively. Figure 12 shows the whole structure of the data transmission packet.

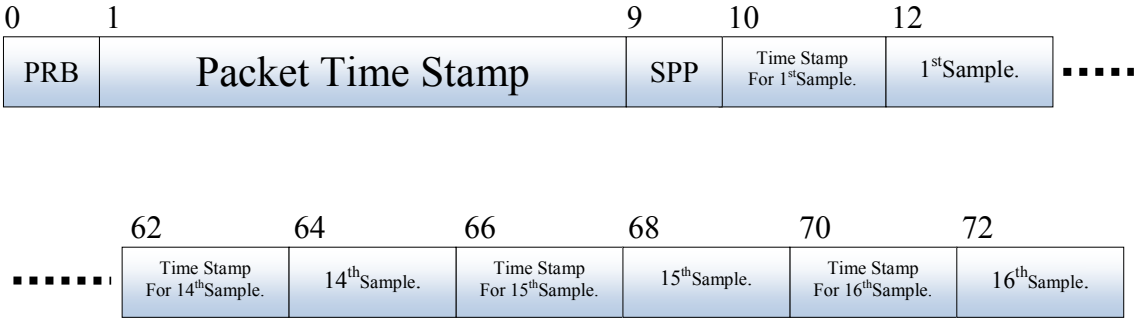

Figure 12 Data Transmission Packet Structure

In some applications of wireless sensor networks, it is a crucial to synchronously gather data from different sensor nodes. As we mentioned before, each sample is time stamped, and this time is taken from the local clock of the sensor node. However, the

time of all clocks is not synchronized with each other. For that reason, time offset has been used to represent the time stamp of the sample. In other words, since we have chosen the sampling frequency to be 100 Hz for data collection, the first sample on the host computer will be stamped with zero milliseconds, the second sample with 10 milliseconds, and so on until the last one. In order to match up all data from different sensor nodes, however, all sensor nodes should start data collection at the same time. The base station can instruct all connected sensor nodes to start data collection by sending out unicast packets for all connected sensor nodes. This would be possible, but all unicast packets cannot be sent out at the same time. As a result, unsynchronized data will be collected. Figure 13 shows the data collection of two sensor nodes using uncast packets.

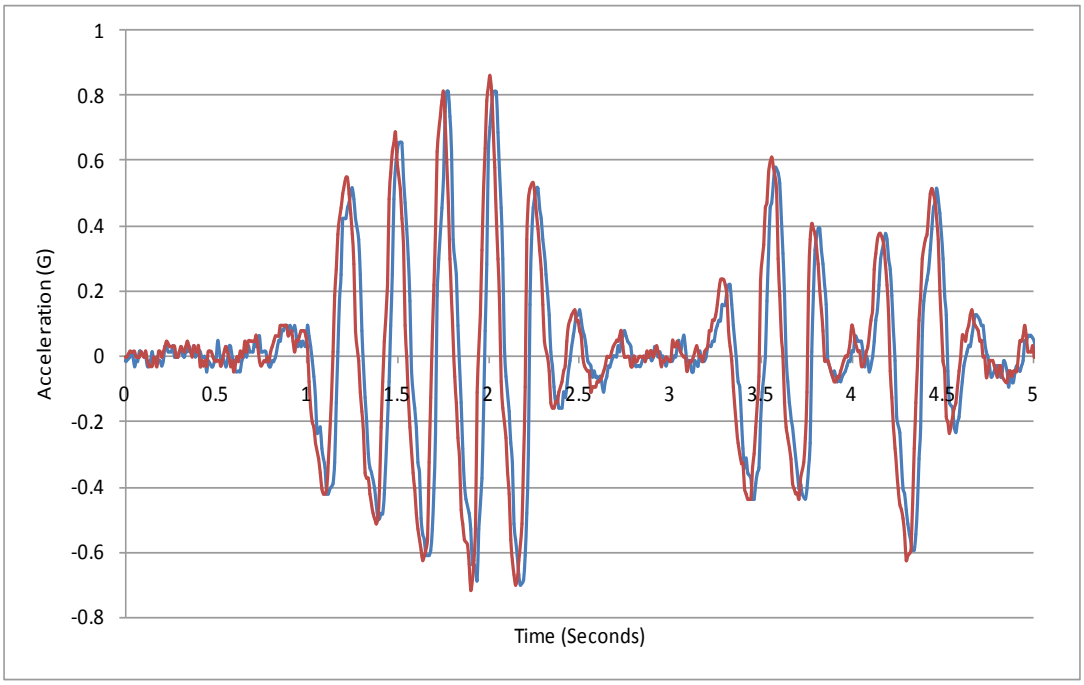

Figure 13 Unsynchronized Data Collection between Two Nodes

To achieve better synchronized data, the host computer application can be programmed to broadcast a start data collection packet to all connected sensor nodes by using one broadcast packet. The broadcast packet will be processed by all SPOTs at the same time. All free-range SPOTs are identical, and they run on the identical programs. As a result, they all start data collection at the same time. Figure 14 shows the data collection of the same two sensor nodes by using a broadcast packet.

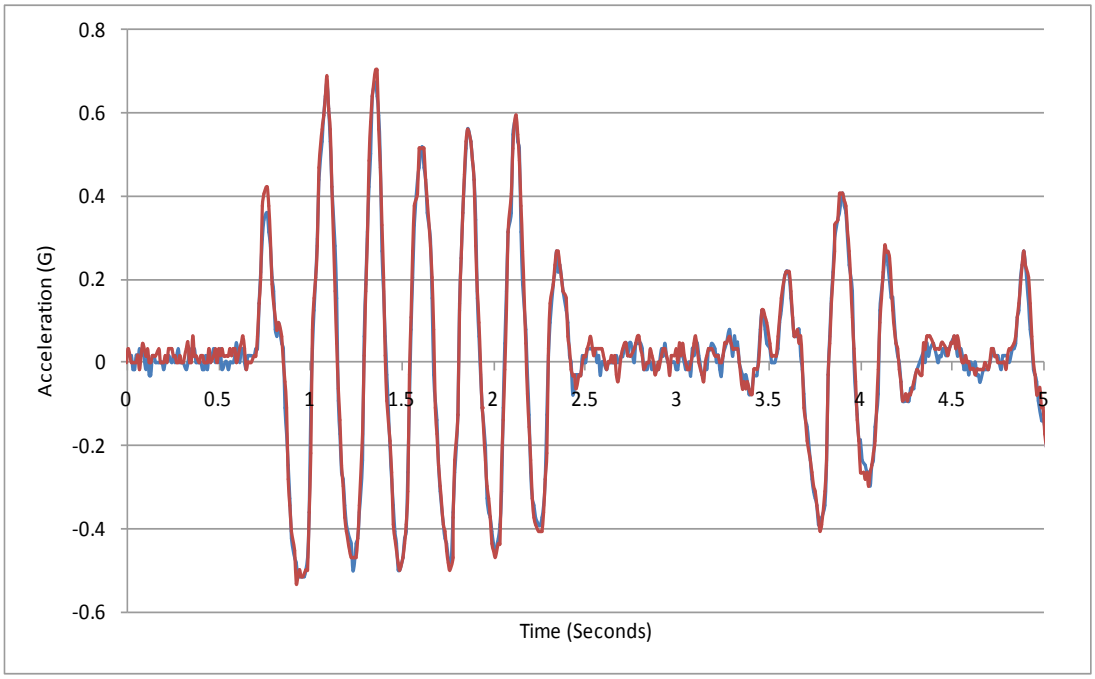

Figure 14 Synchronized Data Collection between Two Nodes

4.6 Packet Traffic Issues

Using the broadcast packet, mentioned in the previous section to synchronize the data, raises high traffic congestion. This congestion happens because all sensors start collecting and sending data together. As a result, all packets carrying the data reach the base station at the same time. The base station may not handle all these packets at the same time. Figure 15 shows this problem graphically for four connected sensor nodes.

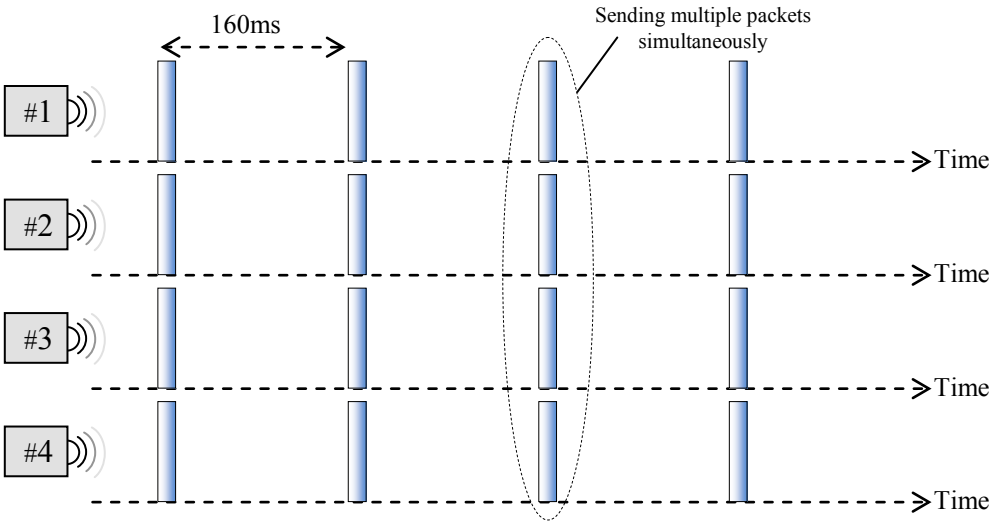

Figure 15 Sending Packets Simultaneously (Without Delay)

To avoid sending multiple packets simultaneously, different time delay has been used in each node. Because we have 100 Hz sampling frequency, a sample of acceleration data is taken every 10ms. Also, since there are 16 samples per each packet, there is a packet to be transmitted to base station every160 milliseconds. As a result, we have a time window of 160 milliseconds when the wireless radio medium is idle, and that idle time can be utilized for other sensor nodes to use it. The time delay of 40-millisecond difference in each sensor node has been used during the transmission of packets. The first node will send the packet immediately whenever it is ready while the second one will be delayed by 40 milliseconds before sending its first packet, the third one will be delayed by 80 milliseconds, and so on. Figure 16 shows this process visually.

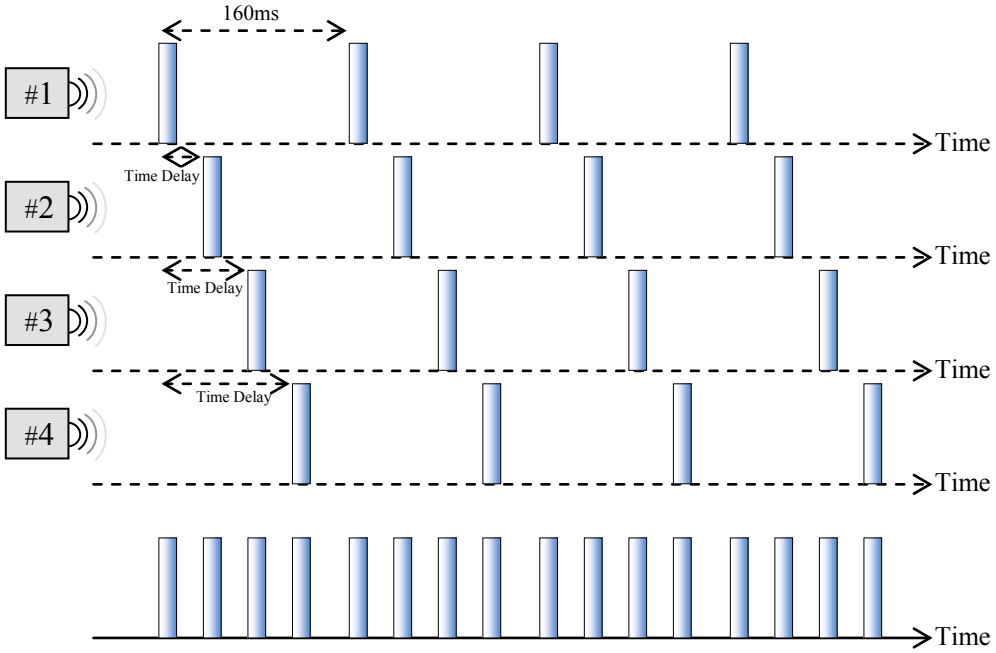

Figure 16 Sending Packets with Delay between Different SPOTs

Another problem that more likely to occur during this traffic congestion is accepting a packet by base station without informing the sender that the packet has been received. As a result, the base station will receive the same packet twice. That causes a problem of having duplicated data sets. The solution to this problem has been addressed by checking the packet time stamp, which gives a unique identification to the packet, of current processing packet with the previous one. If the current and previous packets have the same packet time stamp, the current packet will be ignored, and the next packet will be processed.

4.7 Sensors Calibration

The calibration is the process of determining the rest offset matrix or zero leveling. This matrix is 3 by 3. The columns of this matrix represent the rest offsets of

three axes X, Y, and Z, and the rows represent three different scales 2G, 4G, and 8G. This matrix helps to determine where the rest point of the accelerometer sensor is.

There are two methods used to calibrate the accelerometer. The first method can be performed by changing the zero offset matrix which is permanently saved inside the memory. This method is not practical for this research because it needs to deploy a special program to all free-range SPOT and perform the calibration SPOT by SPOT. However, this method can be implemented one time, and the change in the matrix will be permanent.

The second method is done by measuring the acceleration values of different axes and different scales many times and taking the average value. Then these average values have to be taken and placed in the offset matrix that is saved in the host computer application. This matrix can be used by the host computer application to adjust the raw acceleration data offset. This method is more practical for this project because it can be accomplished very easily. However, the drawback of this method is that it has to be done every time the SPOTs are restarted or moved to a different place. Below is a summary of tasks when the system goes through the calibration process:

- After the user clicks the calibrate button on the main window, the base station will send out command packets to request the connected SPOTs to calibrate their accelerometer sensors.
- All connected free-range SPOTs will receive these command packets and start the process of calibration.

- The process of calibration on each free-range SPOT takes 25 measurements of the accelerometer and saves the average. This process will take place for all three axes and all three scales. The result of the whole process is a 3 by 3 matrix.
- The matrix will be placed in a reply packet, and it will send by the radiogram communication to the base station.
- The host application will receive these rest offset matrices from all connected free-range SPOTs and save them.
- These matrices will be used by the host application to adjust any future raw acceleration samples sent by the SPOTs.

Chapter 5

Related Work

This chapter is related to SPOT configuration, developed software operation, laboratory tests, data collection test, and the reliability of the network. SPOT configuration is explained in order to understand how the WSN can be setup. The software operation section gives some highlighted points about how to operate the software and how to prepare the software for data collection. The laboratory tests are performed to check the functionality and the performance of the network. Some of these tests are shown in this chapter. Then, we will go through data collection test, and we will show the result of this test.

5.1 SPOTs Configuration

Before operating the Sun SPOTs, the configuration of SPOTs is necessary to operate properly. Many system properties of Sun SPOTs can be changed for both the free-range SPOT and the base station through a program, called Sun SPOTManager that comes with the kit. This program which can be also downloaded from the official Sun SPOT website allows us to query the configuration of individual Sun SPOTs that are connected to the computer through the USB cable. Different property of SPOTs can be changed though this program, such as channel frequency, transmitting power, PAN Id, disable or enable routing protocol, and many other properties. Changing the channel frequency can be useful in case if the topology of the network requires operating more

than one WSN at the same time. In this case, each WSN can have different channel frequency in order to be physically separated for other WSNs. There is a possibility of choosing 15 different channel frequencies. The second system property that is necessary to be changed is the power level. Since we want the receiving signal to be as strong as possible, we have to increase the transmitting power as much as possible in order to study the WSN in best possible conditions. To make changes to these properties, the system property *radio.channel* has to be changed to a corresponding channel number, and to change the transmitting power, the system property *radio.transmit.power* has to be changed to its maximum value 31 in all SPOTs. Since these properties are permanently saved in memory, they have only been changed once.

After connecting all SPOTs one by one to the computer through the USB cable and configuring the system properties, the developed free-range SPOT application has been deployed on the SPOTs. The deploying of this application on the SPOTs can be done through the NetBeans IDE, where the application has been developed. The process is identical for all SPOTs except for one thing; which is the time delay that we previously mentioned in Ch. 4, Sec. 6. This time delay can be changed for each SPOT by injecting a sleep java statement in the free-range SPOT application. This statement lets the thread responsible for packet transmitting to be delayed before sending each packet. The time delay parameter in the sleep statement has been varied for each SPOT. When making this time delay, only the thread that is responsible for sending packets will sleep for the specified time. The other thread will not be affected by this delay. For instance, the thread that is responsible for collecting data and placing them into packets will not experience any delay.

5.2 Software Operation and GUI Explanation

After all SPOTs are deployed with the free-range application in NetBeans IDE, the SPOTs are fully capable of communicating with base stations. The base station can be connected to the computer and run the host application. Whenever any of deployed SPOT is turned one, it will send out requests periodically every 10 seconds to locate the base station in the range of its signal coverage area. Meanwhile, the base station will listen to free-range SPOT service requests. Whenever the base station receives a request for the SPOT, it will respond to it with the base station's MAC address. At that point, a point-to-point radio communication will be open between the base station and the SPOT. This point-to-point communication will be managed by a thread that will be dedicated to any further communications to this connected SPOT

While requesting the base station, the two red LEDs on the free-range SPOT will be turned on to indicate that it is searching for a base station. After the free-range SPOT is connected, only one green LED will stay on as it is shown in Figure 17. Meanwhile, on the host computer application, the status of the SPOT and the signal bar will indicate that a new SPOT is connected.

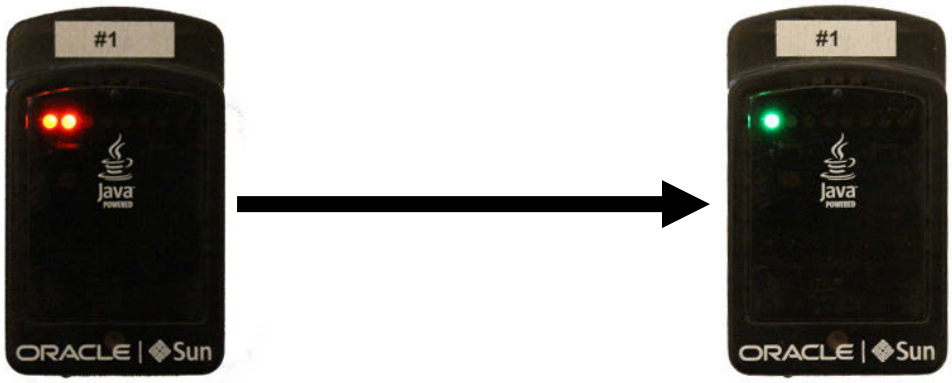

Figure 17 Free-range SPOT LEDs Indication during the Connection Process

The other SPOTs can be turned on in the same fashion, and the other signal bars should rise as the SPOTs get connected. These signal bars are updated only if there is an activity with connected SPOTs, such as pinging or data collection. Figure 18 shows the *SPOTControl Window* with a brief explanation for each part in the window. It also visually shows the signal strength bars as SPOT #1 is connected.

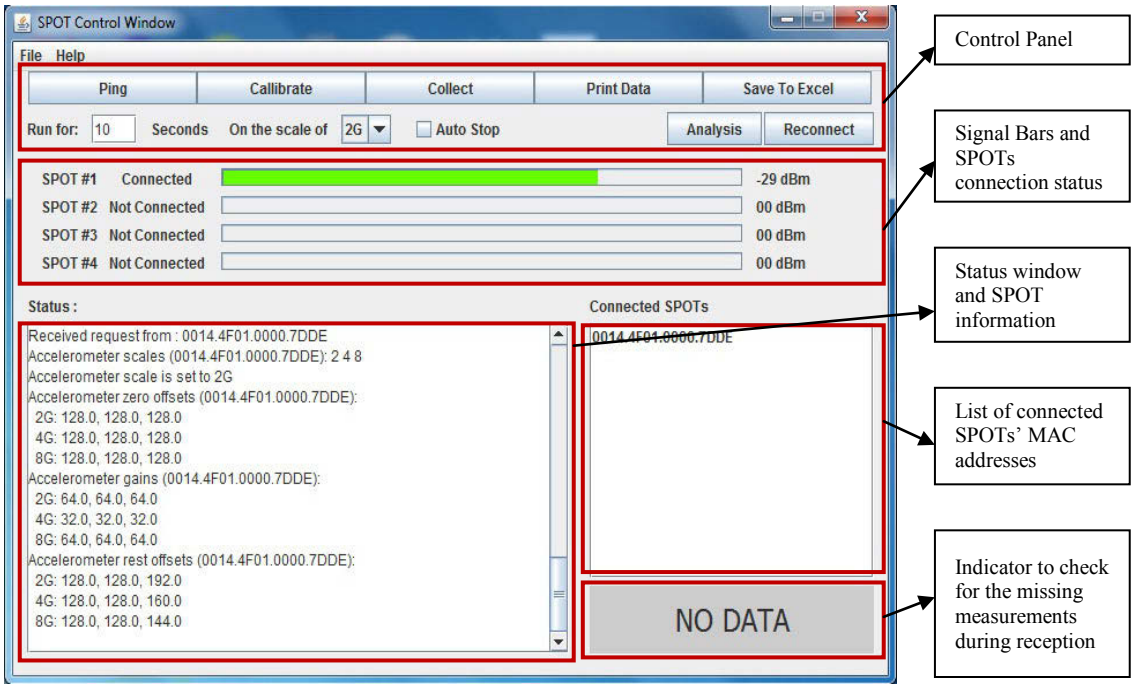

Figure 18 GUI of Host Computer Application with Explanation

Before starting data collection, the SPOTs should be secured at their positions to perform the calibration process. The calibration process is necessary in order to get accurate data. It can be done by clicking on the *Calibrate* button. It takes 5 seconds to calibrate the sensors. While calibrating, the second LED on all POTs will turn on with a blue light. After the calibration is done, the SPOTs will be ready to collect the data. To perform a data collection, the *Collect* button can be used. By clicking on that button, the label of this button will change to "collecting…," which indicates that the data are now in the process of data collection. It can be stopped by clicking on the same button. If the *Auto Stop* check button is activated, the data collection will be automatically stopped after the specified time is over.

After the data collection is stopped, the data can be saved as a Microsoft Excel File by clicking on the *Save To Excel* button. The program will create a table in excel with all sensors' acceleration data samples and their time stamps.

There are also other GUI parts in the *SPOT Control Window* that give us useful information such as *Status* panel which is used to see more information that is coming from the SPOTs. For example, the zero offset matrices of all SPOTs are shown in this panel after calibration process. There is also a *Connected SPOTs* panel which shows all connected SPOTs' MAC addresses.

5.3 Laboratory Tests

Many tests have been conducted concerning RF signal strength. These tests were intended to find a relation between signal strength and the missing packets, as well as to find the best antenna orientation. In order to be more accurate and to know what exactly

this relationship is, two applications have been developed. The development of these applications is based on finding the percentage of received packets as the signal decrement. One of these applications has been deployed on only one free-range SPOT, and the other has been executed on the host computer that has a connected base station. The application on the free-range SPOT broadcasts packets periodically - the period of time is chosen to be a packet every 50 milliseconds - to the base station. The packets carry a serial number that is generated by the free-range SPOT application. On the other side, the host computer application receives packets and checks the serial numbers to determine if there is any lost packet.

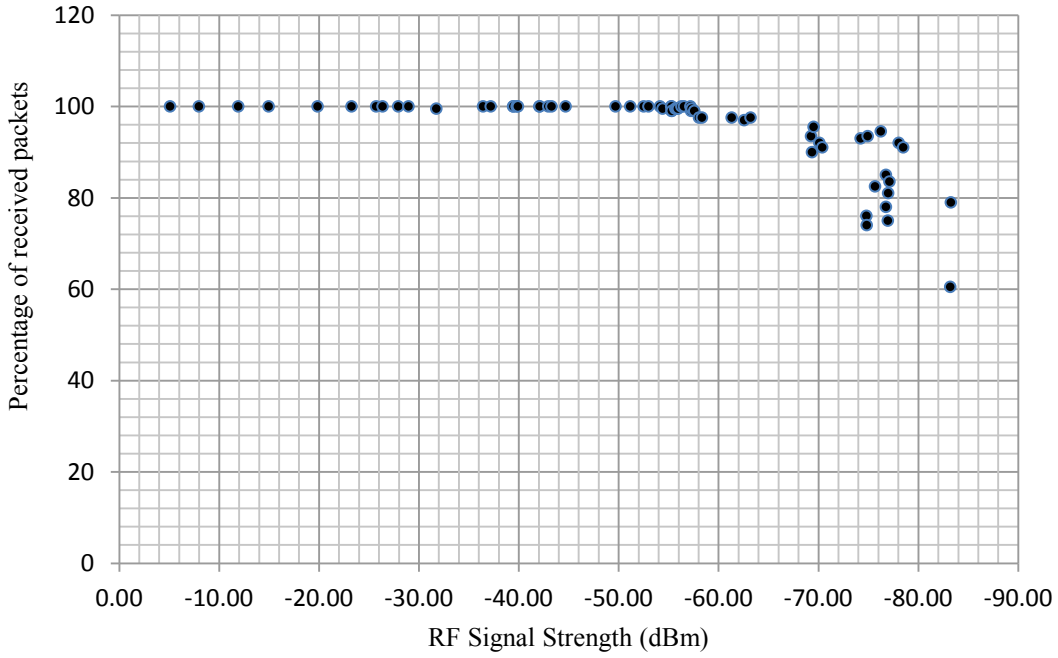

Figure 19 Received Packet Percentage versus RF Signal Strength

As it is shown in Figure 19, each dot represents one run of the program on the host computer. In each run, the free-range SPOT has been moved farther and farther to study the decrease of the RF signal strength. The SPOT is programmed to transmit 200

packets. In the low RF signal strength, not all of these packets have been delivered. It is highly recommended to avoid the RF signal strength less than -70 dBm in order to prevent a miscommunication between the base station and the connected SPOTs when collecting the data. For these tests, a small amount of packet loss is acceptable because these tests were done using broadcast communication. In the point-to-point communication, which is the case with data collection, there is an acknowledgment of received packets, which means the packets delivery is more reliable.

Besides doing tests on the signal strength and radio connection, many tests have been done on the actual program to check functionality of the whole system together. These tests are performed to check the data synchronization among all sensor nodes. In one of these tests, the sensor nodes are put on a flat area like a table, and the program is run. While running the program, the table has been pounded to cause some vibration on the surface of the table. This test causes the sensor nodes to be vibrated at the same time. The expected data graphs from all sensors should be synchronized and identical.

As shown in Figure 20, the acceleration excitations of all sensors happen at the same time. This synchronization might not be very accurate. However, it is visually in an acceptable range.

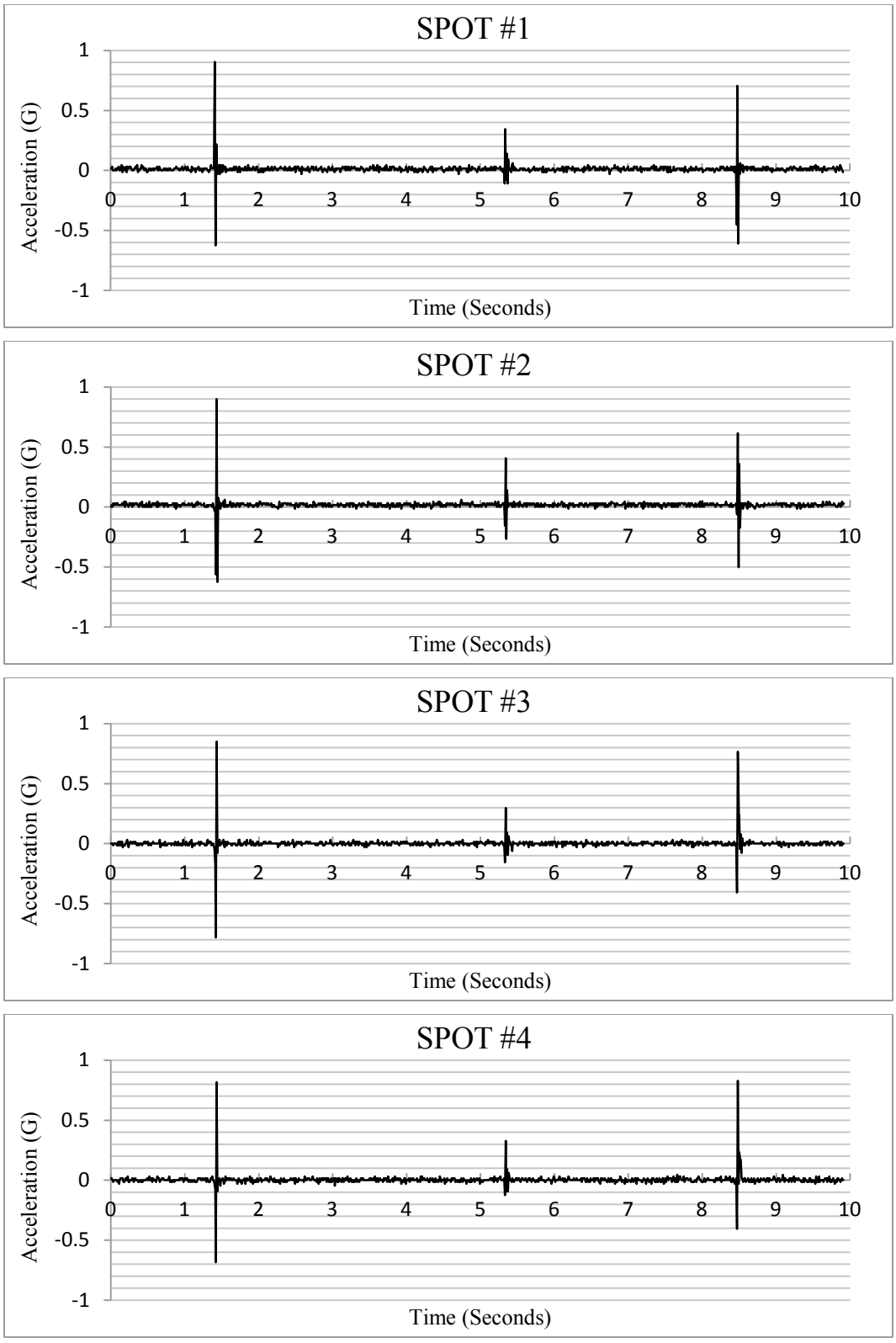

Figure 20 Table Test Result

5.4 Fast Fourier Transform (FFT) Analysis

Preparing for the vibration plate test, the developed host application should be able in somehow to transfer the collected acceleration data from the time domain to frequency domain. This transformation will tell us if we are collecting the data with a correct dominant frequency. To perform this transformation, JTransforms API library has been used. The JTransforms is an open source java library that helps integrate the ability of performing FFT in scientific applications written by java programming language. Implementing this transformation in the host computer application gives us the ability of analyzing the collected acceleration samples in the frequency domain that, in turn, shows us the frequency components of collected data.

The FFT is an efficient way of computing the Discrete Fourier Transform (DFT). DFT is equivalent to Fourier Transformation in continuous signals. It is, however, used to transform discrete signal to discrete frequency spectrum [12]. According to Nyquiest sampling theorem, the maximum frequency that can be observed in frequency domain is one half the sampling frequency. Since, in this project, we have used 100 Hz sampling frequency to collect the data, after performing the FFT, the maximum expected frequency in the spectrum will be 50 Hz.

In the host application, the Jtransforms library is included in the utility class, *SpotData*. This class is used to create objects that represent the acceleration data of sensor nodes. The acceleration data and time stamps are saved in two different arrays in each object. The *DoFFT()* function of this class performs the FFT on the acceleration data. By performing this function on any created object, a new array will be created to represent the FFT data. This array represents the absolute values of the FFT result. After

taking the raw acceleration data on the vibration plate, which will be explained in the next section, these objects are used to manage the collected data and their FFT results. Furthermore, these objects make it very easy to export data as excel files for graphing and storing the data.

5.5 Vibration Plate Test

In order to excite the accelerometers with vibration, a vibration generator has been used. This device is used in many scientific experiments in order to study the oscillations and resonance on different objects. It basically consists of a loudspeaker covered by a plastic box. A metal pin is mounted to this speaker in order to attach different accessories for the purpose of conducting different experiments. The vibration generator can be connected to an electrical function generator. By changing the frequency of the function generator, the vibration generator can vibrate with different frequencies.

This device generates uniform mechanical vibrations that can be used to test the developed WSN. A special plate has attached to the vibration generator pin to hold all sensor nodes. The sensor nodes are tightly attached to this plate to eliminate any vibration between plate and sensor nodes. The main reason of this test is to study the connected sensor nodes when they work together. The acceleration data have been collected on this device with different vibration frequencies. Then, the collected data have been transferred to frequency domain in order to check for the dominant frequency that supposes to occur at the same frequency as the function generator's frequency.

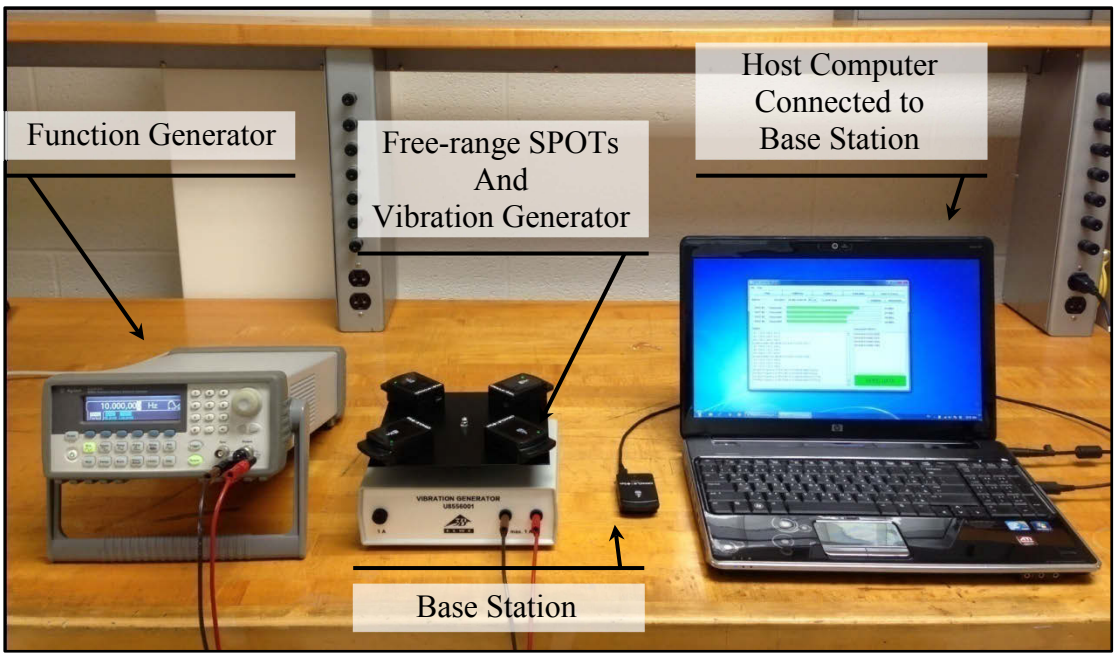

Figure 21 Lab Environment

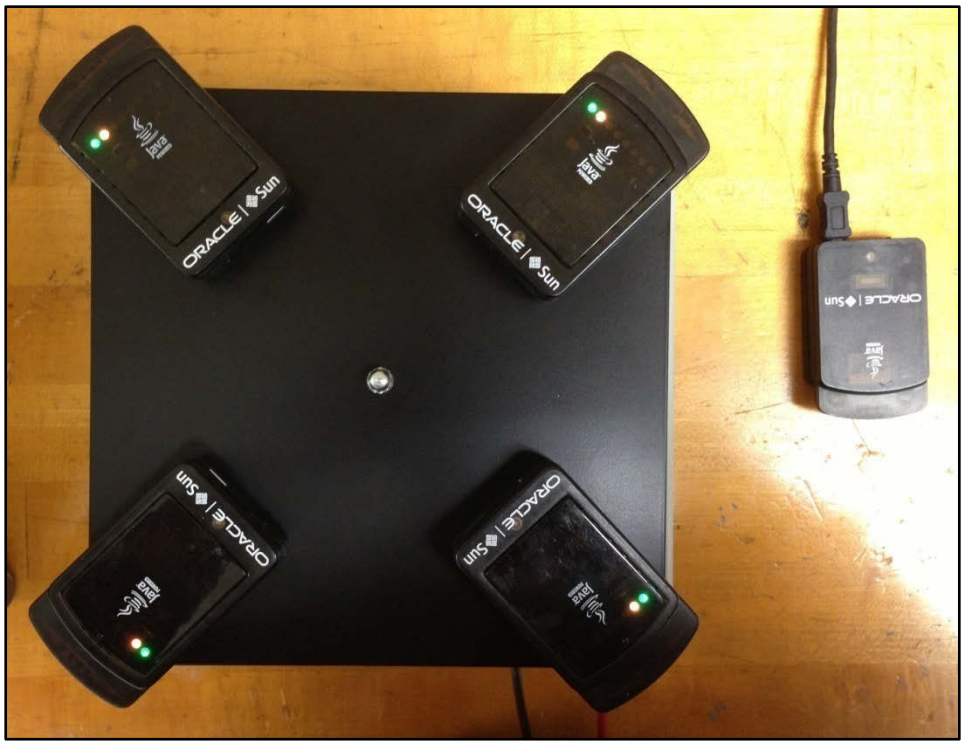

Figure 22 Setup of 4 Free-Range SPOTs on Vibration Plate

Figure 21 and Figure 22 show the lab environment while doing the vibration plate test. This test has been done several times with different input frequencies. Figure 23, Figure 24, and Figure 25 are graphical results of three tests. In these three tests, the function generator has been set to three different frequencies, 10, 15, and 20 Hz. The time domain and the frequency domain of collected acceleration data have been graphically shown. Since all sensor nodes approximately have the same graph, only two sensor nodes' graphs are shown for each tested frequency. By looking at the amplitude of the dominant frequency of each figure, it will be noticed that the amplitude is different from test to test. However, the largest amplitude among these graphs can be seen when the function generator is set on 15 Hz. This gives us an approximate indication that resonant frequency of the attached object, including the plate and the 4 sensor nodes together as one object, occurs around 15 Hz. More tests can be done around 15 Hz in order to find more accurate resonant frequency. While conducting these tests, the received data have been check for losing packets or data samples. The developed software on the host computer is integrated with a mechanism of data checking. An indication on the main control window shows if the data are complete without any data missing.

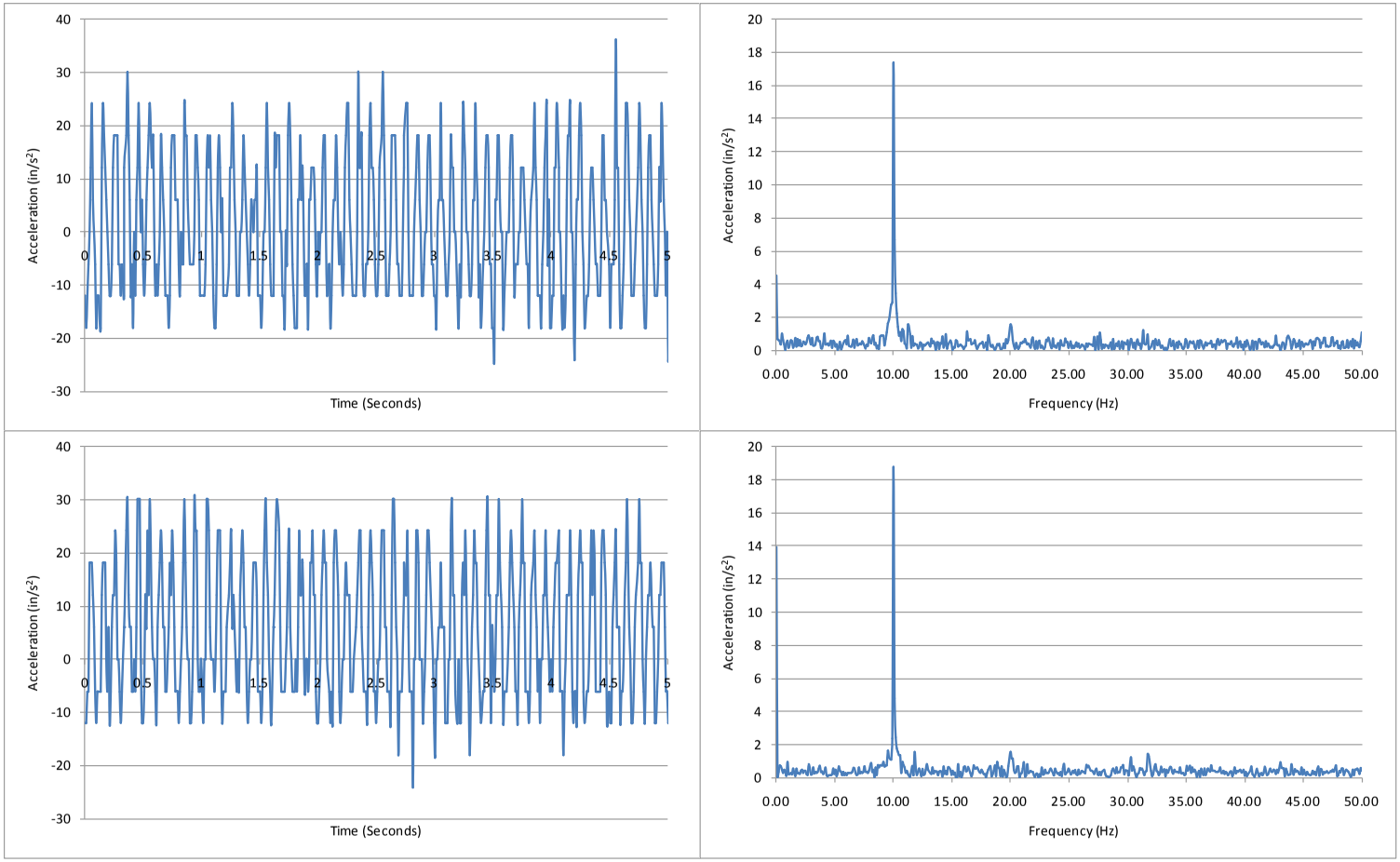

Figure 23 Vibration Plate Test at 10 Hz

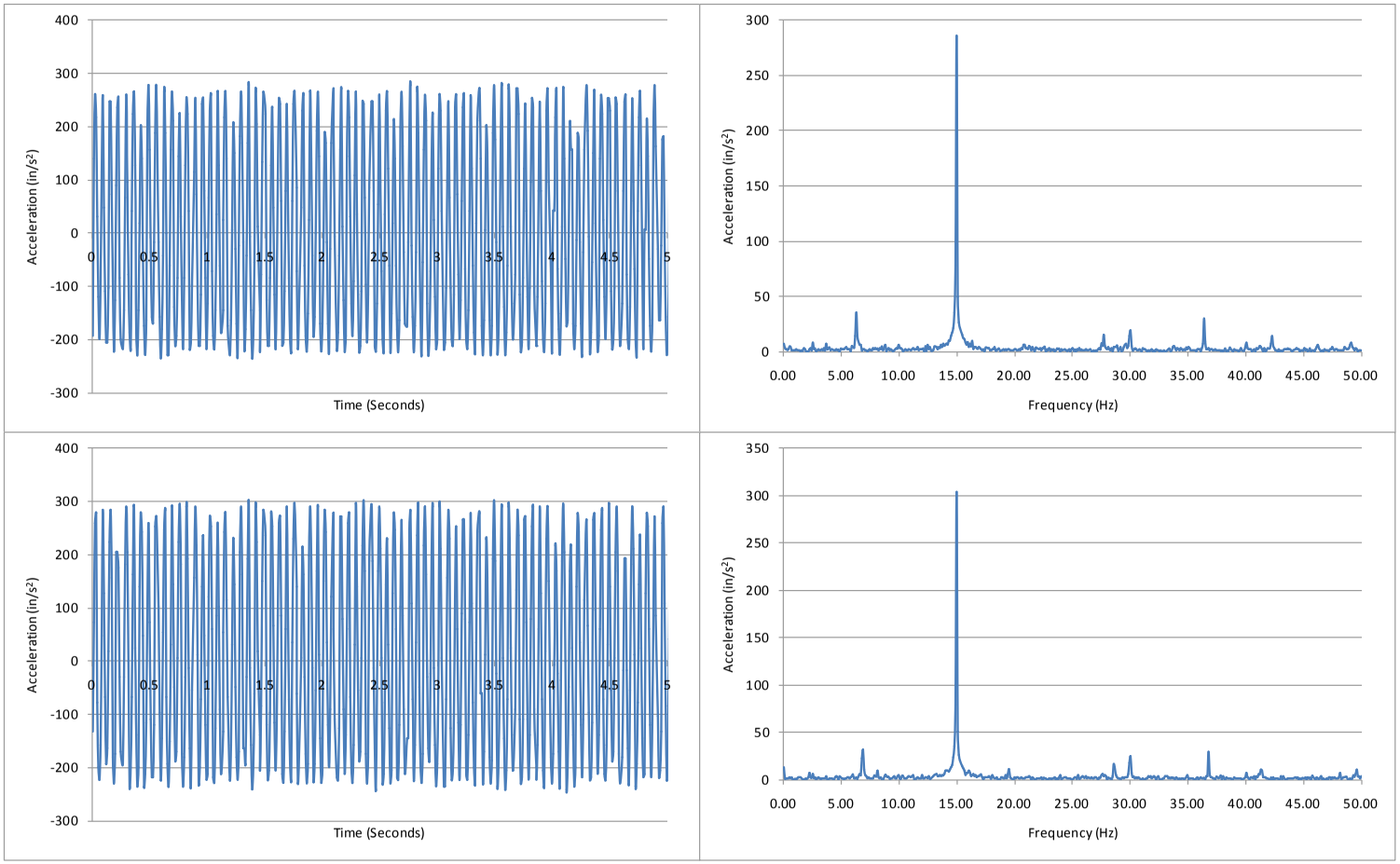

Figure 24 Vibration Plate Test at 15 Hz

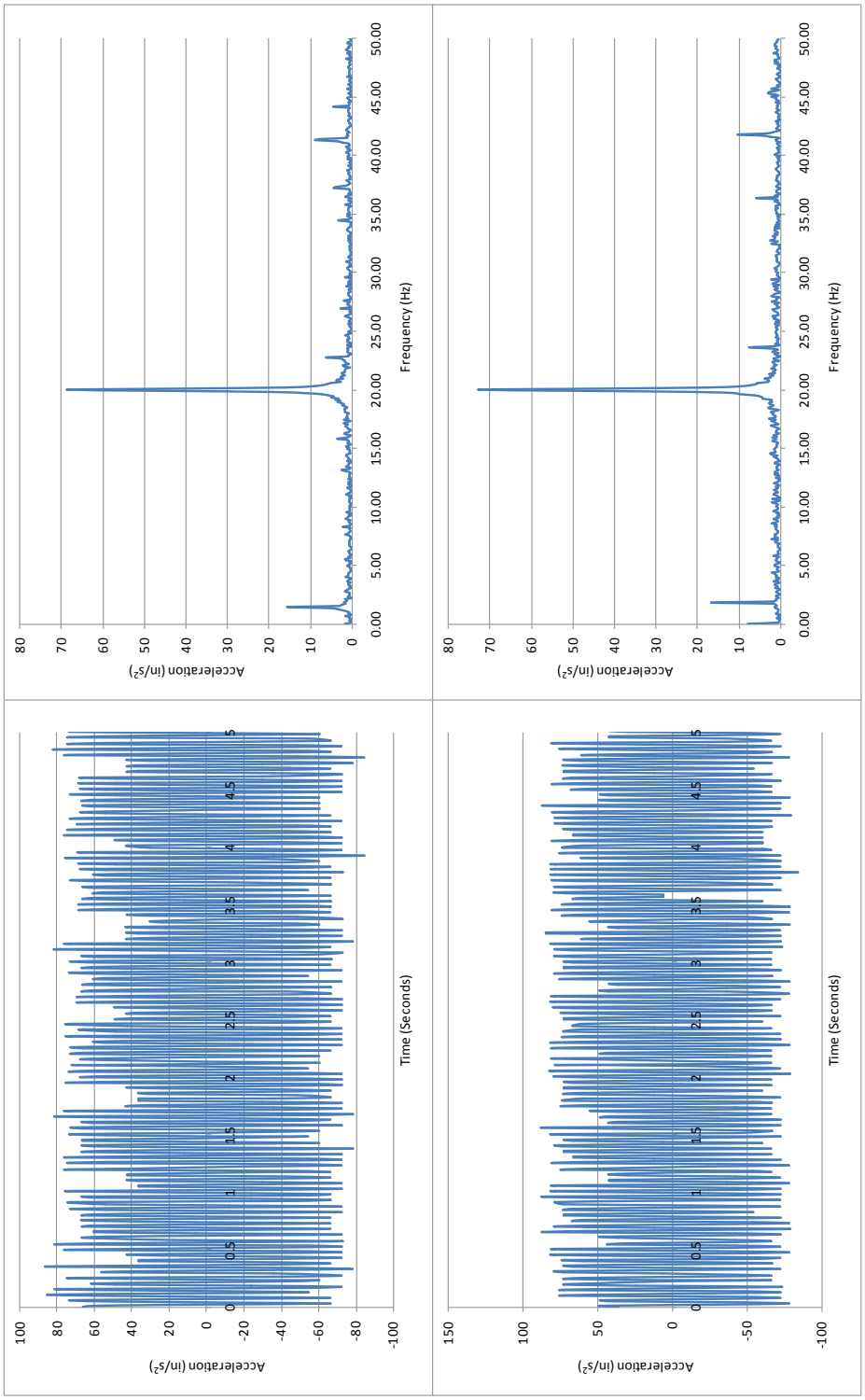

Figure 25 Vibration Plate Test at 20 Hz

5.6 Reliability of the network

The evaluation of the developed wireless sensor network for reliability is very important, especially in applications that require sensitive information gathering. Missing some measurements or packets may lead to wrong data analysis. While the implemented protocols and data transmission mechanisms, such as packet acknowledgment, in Sun SPOTs help improve the reliability of network, the network still experiences some packet loss. Especially if more than two free-range SPOTs are connected to one base station as in our case. For that reason, we intended to check the reliability and to implement a method that checks the credibility of collected data. Missing any data or packets, the hosted computer application will show a visual indicator saying that data are not completely received. In this section, we will go briefly over the technique that is implemented in the host computer application for the purpose of checking the credibility of collected data.

The implemented technique is based on the time stamp of the collected data. After the process of data collection is stopped, the host computer application goes through all the time stamps and looks for the maximum time difference between two consecutive time stamps. Finding a large time difference will indicate missing measurements. Learning from the experiments that we have done, most of the time complete packets were missed during the transmission. This can be easily detected since missing packets can lead to a large time gap between consecutive samples. Missing one packet will leave a 170-millisecond time gap instead of normal 10 milliseconds of time. This is due to that 16 samples per packet are used to transfer data, and every 10 milliseconds one sample is

collected. If this large time gap is detected, the operating user will be notified through an indicator on the main control window of the developed program.

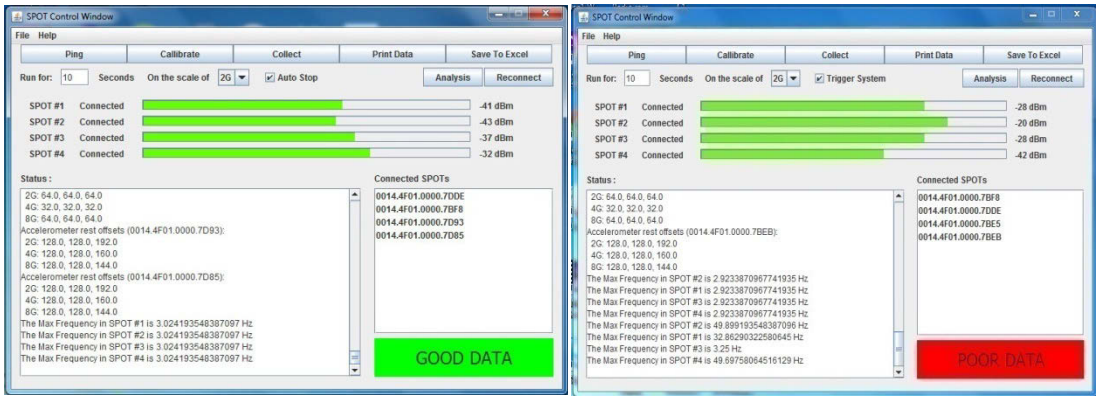

Figure 26 SPOT Control Window in Good and Poor Data Collection

Figure 26 shows the main control window in both the good data collection and poor data collection. If the indicator turns green the data collection is completed without any missing measurements or packets. However, turning this indicator to red will show that poor data with missing measurements has been collected.

Chapter 6

Conclusion, Future Work, and Recommendations

In this thesis, the development of the WSN has been discussed. The main reason for this research was to study the reliability of particular sensor nodes. These particular nodes were chosen to be Sun SPOTs. In order to conduct this study, some applications have been developed on both the sensor nodes and the hosted computer. The accelerometer, which is a built-in sensor on the application board, has been used for data collection in the sensor nodes. The accelerometer data samples have been collected in different tests in order to check the performance of the network in different scenarios. The results of this research indicate that the SPOTs can be used to develop a robust, wirelessly connected network with a capability of collecting data in the sensor nodes and analyzing on the hosted computer.

These wireless sensor nodes can be applied in many real life applications. The amount of sensors on the application board makes the sensor nodes to be easy adopted in useful applications, yet the I/O interface on the application board helps the nodes to be even more flexible for attaching additional sensors such as humidity sensors, pressure sensors, strain gauges, gas and chemical detectors, and many other available sensors.

The developed WSN is a convenient tool to be adopted for condition monitoring of industrial machines such as induction motors. Vibration analysis of these motors is one of the most successful techniques used for condition monitoring. Using the embedded

accelerometer on the application board can be used to perform this vibration analysis. Some of typical issues that may be diagnosed by using this technique are bearing failure, mechanical imbalance, structural resonance and foundation problems, winding damage, etc [13].

Concerning the sampling frequency, the current system works on 100Hz sampling frequency, but it can be increased up to 250Hz. Increasing the sampling frequency up to this value may increase the risk of uneven time interval between samples. The uneven intervals happen because of limited capability of the nodes' hardware resources. For example, the sensor nodes need some time in order to get the sample readings from accelerometers.

Bibliography

[1] Th Arampatzis, John Lygeros, Stamatis A. Manesis , "A Survey of Applications of Wireless Sensors and Wireless Sensor Networks," in *Proceedings of IEEE International Symposium on Intelligent Control, Mediterrean Conference on Control and Automation*, Limassol, 2005, pp. 719 - 724.

[2] Junsuk Shin, Umakishore Ramachandran, M. Ammar , "On Improving the Reliability of Packet Delivery in Dense Wireless Sensor Networks," in *Proceedings of 16th International Conference on Computer Communications and Networks*, Honolulu, 2007, pp. 718 - 723.

[3] Wenqi Guo, W. M. Healy, MengChu Zhou , "Impacts of 2.4-GHz ISM Band Interference on IEEE 802.15.4 Wireless Sensor Network Reliability in Buildings," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 9, pp. 2533 - 2544, September 2012.

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

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

[6] Sun Labs, "Sun SPOT - eDEMO Technical Datasheet," Sun Microsystems, Inc., Technical Datasheet 2010.

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

[8] Hyeopgeon Lee, Kyounghwa Lee, Seunghak Ryu, Sang-Hong Hong Lee, Kwanho Song, Yongtae Ongtae Shin , "An efficient slotted CSMA/CA algorithm for the IEEE 802.15.4 LR-WPAN," in *International Conference on Information Networking (ICOIN)*, Barcelona, 2011, pp. 488 - 493.

[9] Sun Labs, "Sun SPOT Programmer's Manual," Sun Microsystem, Inc., Technical Datasheet 2010.

[10] P. Niemeyer, J. Knudsen, *Learning Java*, 3rd ed. Sebastopol, CA, United State of America: O'Reilly Media, Inc., 2005.

[11] D. Simon, C. Cifuentes, D. Cleal, J. Daniels, D. White, "Java(TM) on the Bare Metal of Wireless Sensor Devices - The Squawk Java VM," in *Second International Conference on Virtual Execution Environments*, Ottawa, Canada, 2006.

[12] Vinay K. Ingle, John G. Proakis, *Digital Signal Processing Using Matlab*, 3rd ed. Stamford, United State of America, 2012.

[13] M. Tsypkin , "Induction motor condition monitoring: Vibration analysis technique - A practical implementation," in *IEEE International Electric Machines & Drives Conference (IEMDC)*, Niagara Falls, ON, 2011, pp. 406 - 411.