WIRELESS SENSOR NETWORK FOR STRUCTURAL HEALTH MONITORING

by

Phaneendra K Kolli

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the

Electrical Engineering

Program

YOUNGSTOWN STATE UNIVERSITY

May, 2010

Wireless Sensor Network for Structural Health Monitoring

Phaneendra K Kolli

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:

_____

Phaneendra K Kolli, Student Date

Approvals:

_____

Dr. Frank X. Li, Thesis Advisor Date

_____

Dr. Philip C. Munro, Committee Member Date

_____

Dr. Faramarz Mossayebi, Committee Member Date

_____

Peter J. Kasvinsky, Dean of the School of Graduate Studies and Research Date

ABSTRACT

A wireless sensor mesh network for health monitoring of structures is presented. It is a low cost, easy to deploy, fast and reliable wireless sensor network. Wireless nodes are all identical to each other with on board sensors for measuring acceleration and temperature. The acceleration data from the nodes used to detect the strain of the structure was calibrated using a Vishay P3 strain gauge instrument. These sensor nodes can collect data as well as relay the data of the neighboring nodes. Data from all the nodes reaches the base station through multiple hop relays. The nodes were tested for their performance by using different frequency channels and radio output power levels. This network implements an energy efficient routing protocol which can also handle a node failure in route without losing data. Different power conservation techniques were discussed which can keep the network unattended for a week after being deployed on the structure.

ACKNOWLEDGEMENTS

I would like to thank my advisory committee and the faculty of the Department of Electrical and Computer Engineering at Youngstown State University. My special thanks to Dr. Li for all his help and support. I have gained lot of knowledge, technical wisdom, and self confidence while earning my graduate degree.

I would also like to thank my family, my friends, and my well wishers. Without your support, I would not have been able to complete this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Data_Mon	Data Monitor
FIFO	First In First Out
GUI	Graphic user Interface
LLN	Link Layer Notification
MCK	Master Clock
NST-AODV	Not So Tiny – Ad-hoc On-demand Distance Vector Routing
OTA	Over The Air
RERR	Route Error
RREQ	Route Request
RREP	Route Reply
RSSI	Received Signal Strength Indicator
SHM	Structural Health Monitoring
SPOT	Small Programmable Object Technology
VM	Virtual Machine

CHAPTER I

INTRODUCTION

The Structural Health Monitoring (SHM) is done to assess the condition and evaluate the health of the structures like buildings, bridges and oil rigs. Usually this is done using wired electromechanical sensors by installing them in concrete during construction. Installing wired sensors for the structures is a time consuming process and is expensive [1]. These wired installations have a single data acquisition unit which is usually located at the center of the structure. For huge structures, long cables have to be run from the data acquisition unit to the sensor making them more susceptible to damage and also more prone to pickup noise. Therefore, design of a low cost, easy to install and fast sensor network for the SHM is initiated.

Advances in the wireless communication technology make it possible to develop a wireless sensor network for structural health monitoring. In this approach, sensor nodes with onboard sensors for temperature and acceleration are deployed to structurally critical locations of the structure. These nodes are capable of communicating with each other. They can be used either for data collection or to relay the data transmitted by the neighboring nodes. The wireless sensor network for SHM has a base station which is connected to a data acquisition unit. All the data from the wireless sensor nodes should be delivered to the base station using a quick and energy efficient path which is discussed in Chapter IV in detail. A sensor node is capable of sending data to the base station using different paths. This makes the network resilient to the node failure. The data collected

from all the nodes is used for calculating the health of a structure which determines the steps to be taken to fortify the structure.

1.1 History of Sensor Networks

Like many technologies, sensor networks were initially developed for defense applications. The very first sensor networks included radar networks used in air traffic control and sound surveillance system on the ocean bottom used to detect and track enemy submarines. Information from the sensors was used to inform soldiers in the field using a communication network. During 1980s and 1990s several military sensor networks were developed including Cooperative Engagement Capability (CEC) in which multiple radars collect data on air targets, Fixed Distributed System (FDS) and Advanced Deployable System (ADS) which were used for anti submarine warfare and Unattended Ground Sensors (UGS) used for remote battle field sensing. Initial research on sensor networks started around 1980 at Defense Advanced Research Projects Agency (DARPA) [2]. This resulted in some acoustic sensors, high level communication protocols, algorithms and software. Carnegie Mellon University in Pittsburg implemented a research for a network operating system resulting in a communication operating system called Accent [3] which helped in transparent networking and reconfiguration. Later Massachusetts Institute of Technology developed some signal processing techniques to analyze the data obtained from acoustic microphones for tracking helicopters [4]. They also developed a real-time test bed of acoustic tracking of low-flying aircraft which was successfully tracked. The Micro Electro Mechanical System (MEMS) technology in the

21st century has drastically decreased the size of sensors. Advances in communication and computing technologies helped to develop low-power processors and increase the range of wireless communication [2]. These helped in developing low-weight, small size, easy to deploy sensors for ad-hoc networks. However these sensor networks have variety of applications including infrastructure security, structural health monitoring and industrial sensing.

1.2 Sensor Networks in Structural Health Monitoring

Health of a concrete structure is evaluated by its behavior and performance. These depend on several factors like strain, vibration, deflection due to dynamic loading, and current strength of the materials used in construction. Even though an exact evaluation of the structure could not be made, an approximation on the condition of the structure can be made. This helps in preventing further damage to the structure making it less expensive and easier to repair it. It may even prevent loss of people's lives by preventing the structure from collapsing.

There are many aspects that have to be taken into account while designing a sensor network for structural health monitoring [5]. The sensor nodes have to be light weight and flexible in order to be deployed to any point on the structure. The network should be expandable in case more data is required from additional locations on the structure. Usually data collection from structures demands higher data sampling rates. Because the civil structures are designed not to sustain prolonged vibrations, the vibrations decay very quickly so there is a need for higher sampling rates. Unlike older

data acquisition systems, these sensors have to be remained unattended for several weeks. In case personnel monitoring or analyzing the data need the data at different sampling frequency, they should be able to change the sensor setting remotely. The most limited resource for the sensor nodes is the power. To have a decent estimate of the health of a structure, data should be provided at least for several days. Measures have to be taken that the nodes implement power conservation techniques (which are explained in Chapter V) and are provided with external batteries for more power at the time of deployment.

1.3 Organization

This work is divided into six chapters. Chapter II explains about the sensor used in the wireless sensor network and calibration of that sensor. Chapter III provides an overview of the antenna used in the sensor node and its characteristics. Fourth chapter describes the implemented wireless mesh network and the future expanded network. It also explains the testing of the wireless mesh network. Chapter V talks about the power management of the sensor node and power conservation techniques to implement. The final chapter summarizes the work done and future work. Appendix A provides the Java code for collecting the acceleration data from the sensor node.

CHAPTER II

SENSORS

2.1 Sun SPOT

The Sun Small Programmable Object Technology (SPOT) shown in Figure 2.1 is used as the sensor for the purpose of health monitoring. Sun SPOT is a small wireless battery powered computing device which runs on Java embedded software development platform without an operating system [6]. Sun SPOT has been developed by the Sun Microsystems Inc. The Java virtual machine (VM) running on a Sun SPOT is called Squawk [7]. It is capable of running many applications in one VM. A Sun SPOT has analog and digital inputs, I/O pins, eight tri-color LED's, two switches, and on-board sensors including an accelerometer, temperature sensor, and light detector.

Fig. 2.1 Sun SPOT

The main processor on the Sun SPOT is an Atmel AT91RM9200 system on a chip which is based on an ARM (Acorn Reduced instruction set computer Machine). It has 512 K of RAM and 4 MB of flash memory. The Squawk use 80 K of RAM and the java libraries use 270 K of the flash memory. These libraries enable control over lot of external devices. Sun SPOT has an internal battery of 3.7 V 720 mAh rechargeable lithium-ion cell. It can be charged using a USB mini B connector. The device goes into a deep sleep mode to save power. The power controller of the SPOT keeps a real time clock running to wake up the device whenever necessary. The eSPOT main board (also called the processor board) of the Sun SPOT shown in Figure 2.2 has all the important components of the device including main processor, memory, power management circuit, battery connector and radio communication part which is an IEEE 802.15.4 radio transceiver and antenna.

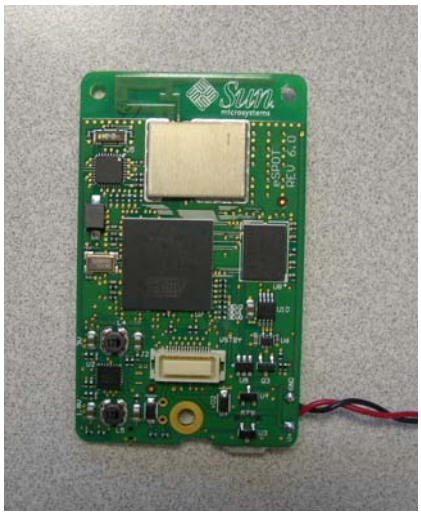

Fig. 2.2 eSPOT main board

The wireless radio communication part of the device is controlled by a CC2420 chip (the large silver colored chip in Figure 2.2) which complies with IEEE 802.15.4

standard [8]. 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. The Sun SPOT can be tuned to 16 different frequency channels and 8 different power levels. The antenna used in a Sun SPOT is an inverted F antenna printed on the top layer of the printed circuit board. Antenna is discussed in more detailed in the next chapter. The Sun SPOT can be programmed for different applications [9] and these applications can be deployed to the SPOT remotely by using Over The Air (OTA) command. This feature is enabled using a tool called SPOT World. It enables the user to handle the applications which are running in different Sun SPOT's and would even let the user to transform the applications from one SPOT to the other.

2.2 Strain Gauge Sensors

Strain gauges are the widely used sensors for measuring strain [10-11]. They are also used in structural health monitoring systems [12]. A strain gauge is usually a metallic foil of electric conductor with an insulated backing layer. It works on a simple principle that when an electrical conductor is temporarily deformed its electrical resistance changes. To measure the strain of an object the strain gauge should be attached to that object using a suitable adhesive. When it comes to the SHM it is not an easy task to install the strain gauges using adhesives and calibrating them after installation. This is also very expensive. However hooking the strain gauges to wireless nodes and wirelessly transmit the data further increases the cost.

As discussed in the previous section a Sun SPOT can be used instead of a strain gauge to potentially detect the stress of a structure. This is cost effective, with less work, fast and easy alternative to the strain gauge sensors.

2.3 The 3D Accelerometer

The accelerometer embedded in the Sun SPOT is ST Microsystems 3-Axis 2g/6g inertial sensor LIS3L02AQ. Orientations of the axes of the accelerometer are, the X axis is parallel to the row of LEDs, Y axis is parallel to the SPOT's surface perpendicular to the row of LEDs and the Z axis is perpendicular to the SPOT's surface. The typical maximum band widths to measure the accelerations are 4.0 KHz for the X and Y axis and 2.5 KHz for the Z axis as per the data sheet of the ST Microsystems. After sampling them internally, low-pass filtering, analog-to-digital conversion, and time taken by the ARM9 processor to run the application the maximum sampling rate at which the SPOT can be used is 320 Hz. This means taking a set of readings every 3.125 ms. This sampling time implies even when other applications are running in the SPOT. If the SPOT application is only dedicated to reading the accelerometer the sampling rates can be at 2.2 KHz or 460 us for all the three axis.

2.4 Calibrating the Accelerometer

For any equipment used to collect data or take readings, calibration of that equipment is necessary. To calibrate the accelerometer of the Sun SPOT, a test was done to compare the acceleration data from the Sun SPOT to the strain data from a Vishay P3 Strain instrument. A steel beam was used for the test. The test was set up as shown in the Figure 2.3.

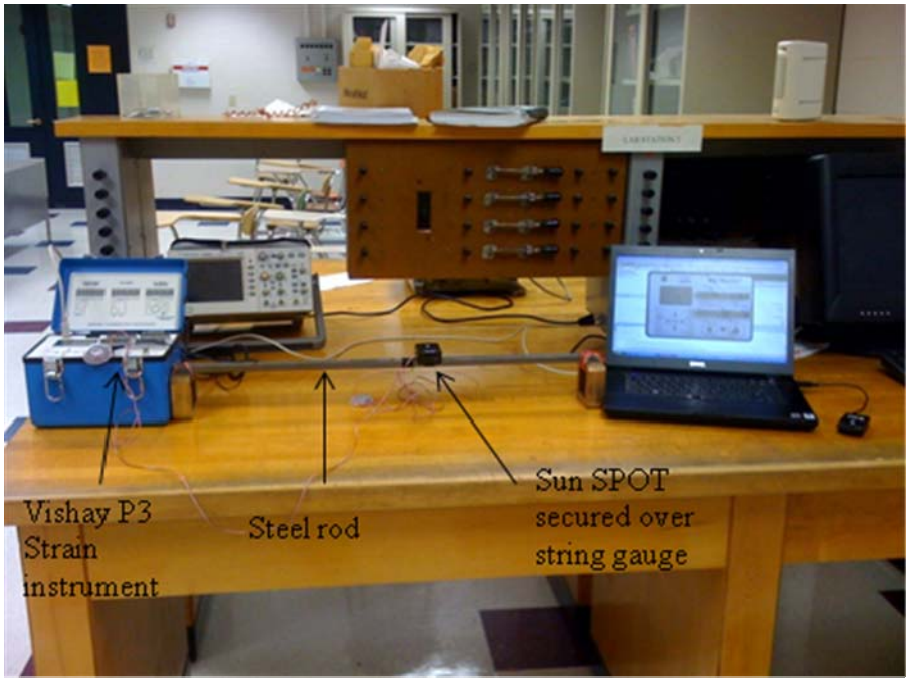

Fig. 2.3 Test setup for calibrating the accelerometer

The strain gauge was glued to the steel bar and the Sun SPOT was secured over the top of the strain gauge using a tape. The strain instrument was hooked to the computer using a USB cable. Periodic forces were applied to the steel bar while the strain

and the acceleration were recorded from the Vishay P3 strain instrument and Sun SPOT respectively. The data recorded from them is shown in Fig. 2.4.

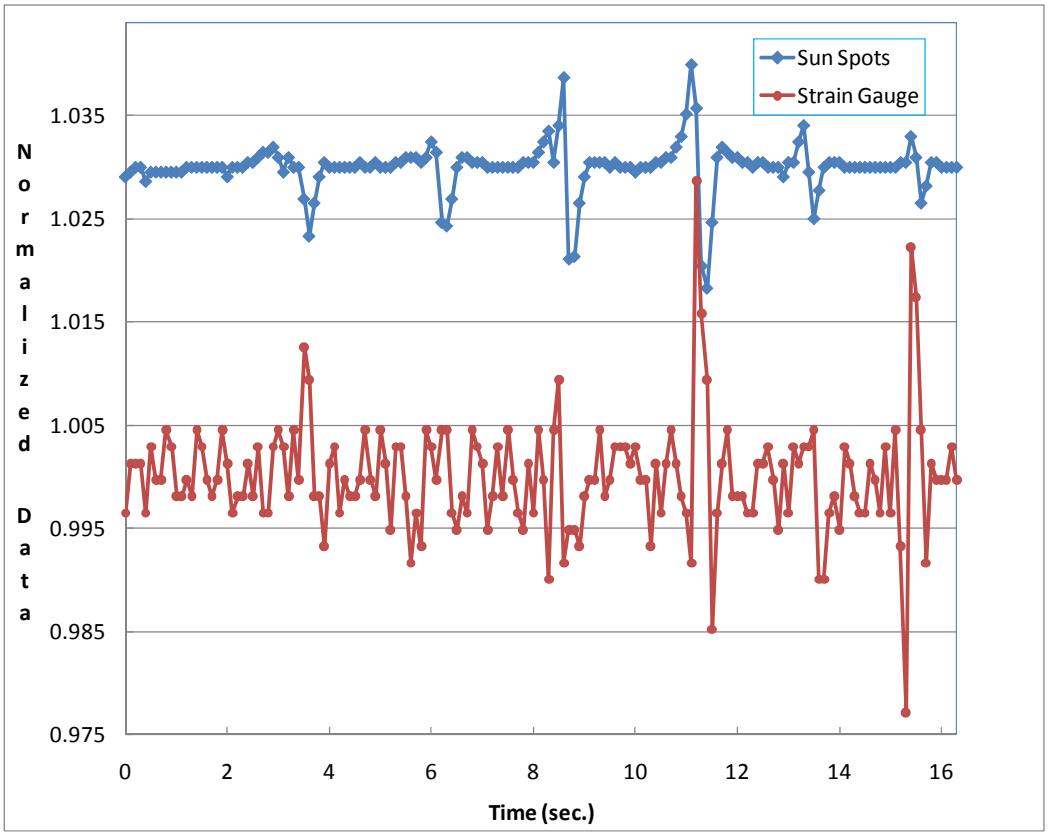

Fig. 2.4 Correlation data from the strain instrument and Sun SPOT

By comparing the data from the wireless sensor node and the P3 strain instrument, it is seen that the data is correlative. The correlation is seen when the deflections occurred at the same time instance from both the equipment. The unevenness in the peaks between both the data is because of the different data sampling rates. This shows that the Sun SPOT's can be used to detect the stress of a structure and thus evaluating its health.

CHAPTER III

ANTENNA

3.1 Inverted F Antenna

The inverted F antenna which is shown in Figure 3.1 is the antenna used in the Sun SPOT. It is an omni directional antenna printed on the top layer of its printed circuit board [13]. Its omni directional radiation is in the plane of the printed circuit board. This antenna designed with the measurements shown in Figure 3.1 has a center frequency of 2.45GHz and operates well for all the frequencies in 2.4 GHz ISM band.

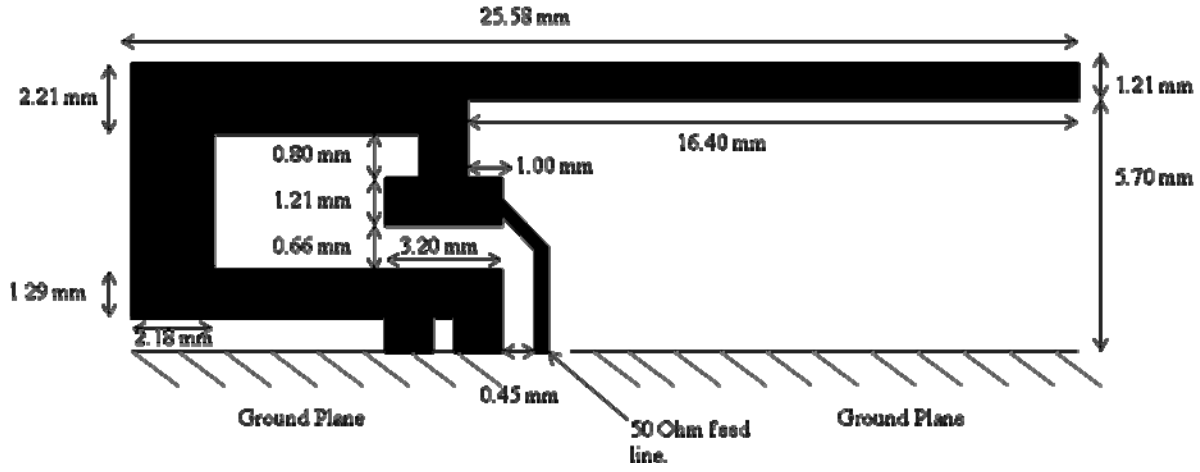

Fig 3.1 Inverted F antenna with measurements

The PCB thickness will have some effect on the performance of the antenna. Hence for better performance the inverted F antenna is printed on the PCB of thickness of 1 mm in the Sun SPOT. The inverted F antenna in the Sun SPOT can be operated at an

output power ranging from a maximum of 0 dBm to a minimum of -25 dBm. The overall antenna size can be given as 25.7 x 7.5 mm. Some of the advantages of the Inverted F Antenna are that it is compact, it has an omni directional pattern, and is cost effective. However the main advantage is its flexibility. It can be modified to achieve desired characteristics [14]. The main application of the Inverted F antenna is for short range wireless devices operating in 2.4 GHz band.

3.2 Inverted F vs. Folded Dipole Antenna

An inverted F antenna and a folded dipole antenna both designed for a radio chip CC2420 (made by Chip Con) operating at a frequency of 2.4 GHz are compared. Both these antennas were operated at an output power of 0 dB. Both of them are omni directional. The maximum of gain of Inverted F antenna is measured to be 3.3 dB where as for the Folded Dipole antenna it is 0.3dB. The size of the Inverted F antenna is 25.7 x 7.5 mm where as that of folded dipole is 46.6 x 9 mm. The width of the Sun SPOT is 40 mm, making the folded dipole inappropriate for use with it, thus making the Inverted F antenna a compact, low cost and high performance antenna to use within Sun SPOT.

3.3 Signal Strength and Distance

The base station can be programmed to ping (done to check the connectivity and find signal strength) the Sun SPOT in order to see the RSSI (Received Signal Strength Indicator). RSSI helps us in estimating the strength of the signal between the SPOT and the base station. RSSI ranges from +60 (max) to -60 (min). To convert the RSSI into decibels, subtract 45 from the received RSSI. The maximum distance at which a Sun SPOT can be placed from the base station that they can communicate with good signal strength has been estimated as explained below.

A demo program was run on both base station and Sun SPOT so that the base station was able to ping the SPOT and get RSSI. Both the base station and the SPOT were initially put close to each other and pinged. The distance between them was increased in steps of 5 feet and pinged each time. The results recorded are shown in the Table 3.1. First row of the table shows the distance between the base station and SPOT in feet. Second row shows the signal strength for the signal from the host (in this case the base station) to SPOT. Third row shows the signal strength for the signal from the SPOT to the host. Table is plotted in a graph as shown in Figure 3.2.

Table 3.1: Signal strength results for maximum distance between nodes

Distance (ft)	0	5	10	15	20	25	30	35	40	45
Host->SPOT (dBm)	-51	-64	-71	-83	-88	-89	-91	-92	-90	-93
SPOT->Host (dBm)	-52	-69	-80	-82	-78	-78	-80	-83	-79	-82

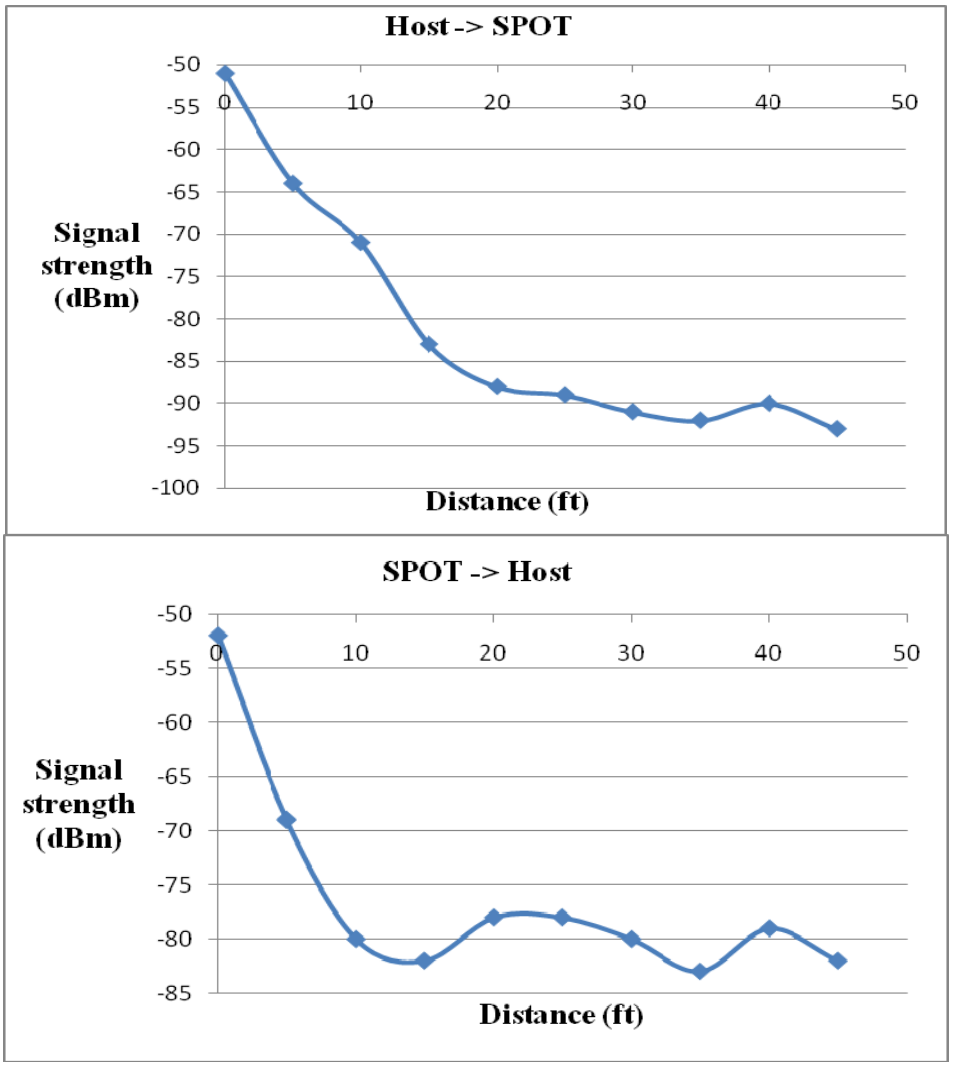

Fig 3.2 Signal strength vs Distance results

From the results shown in Figure 3.2 it is clearly evident that signal strength is decreasing with increase in distance. But after reaching 45 feet, SPOT was moved to 50 feet. Then the nodes have problem connecting to each other. After two unsuccessful attempts at 50 feet they got connected the third time. At that point base station could hardly ping the SPOT. When moved back to 45 feet, connection was good and was able

to ping the SPOT and collect data. This experiment was done indoors in a hallway. Hence from the above results and analysis the maximum distance between them at which SPOTs can be placed is determined to be 45 feet (or 13.7m).

3.4 Channel Change Performance

The Sun SPOT can be operated in sixteen channels. They are numbered from 11 to 26. Each channel is incremented by 5MHz from the previous channel, where the first channel no. 11 operates at a frequency of 2405 MHz. Similar to the evaluation of the signal strength with distance in previous sub chapter, a demo program has been run to estimate the performance of the channels. The radio transmit power is set to a maximum of 0dBm and the nodes (base station and Sun SPOT) were placed at a fixed distance of 10 feet. Channels were switched one after the other while pinging the SPOT simultaneously to record the RSSI of each channel. Due to the FCC regulations channel 26 cannot be operated at 0dBm, so it was operated at -3 dBm. The recorded results are as shown in Table 3.2. and in Figure 3.3.

Table 3.2: Signal strength results for channel change

Channel No.	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Signal strength (dBm)	-78	-78	-83	-84	-82	-84	-80	-79	-78	-85	-89	-86	-86	-87	-89

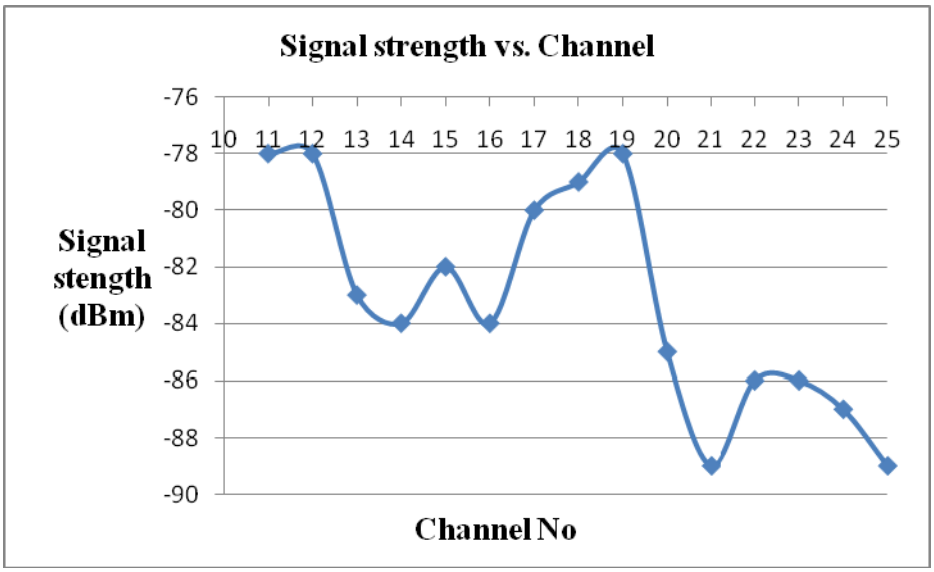

Fig 3.3 Signal strength vs Channel results

The results in Figure 3.3 show that the nodes are efficient in communicating with each other when put to maximum power and low frequency channels (below channel no. 20).

CHAPTER IV

WIRELESS MESH NETWORK

4.1 Introduction

The wireless mesh network designed for the structural health monitoring is a low power, low cost and reliable wireless personal area network (WPAN). The structural health monitoring using a fixed wireless mesh network is made possible by the advances in the technology of wireless sensor network. This multihop mesh network is mainly aimed at the network robustness for node failure and high power efficiency [15]. Because of the very low weight and portability of the sensor nodes, it is easy to deploy the network on any kind of structure and collect the data required to evaluate its health. This makes the network extremely adaptable and scalable.

4.2 Topology

The implemented mesh topology of the wireless mesh network consists of eight sensor nodes and a base station. The network is arraigned as shown in Figure 4.1. Each sensor node has several listeners and a Data_Mon (data monitor). Each listener receives the data broadcasted by the neighboring nodes and relays it to other nodes. Data_Mon collects the sensor data and broadcasts it to the neighboring listeners.

The data finally reaches to the destination (in this case the base station), by getting relayed through the wireless nodes in a peer-to-peer communication style. The routing technique for the nodes to relay the data is discussed in the next section of this chapter.

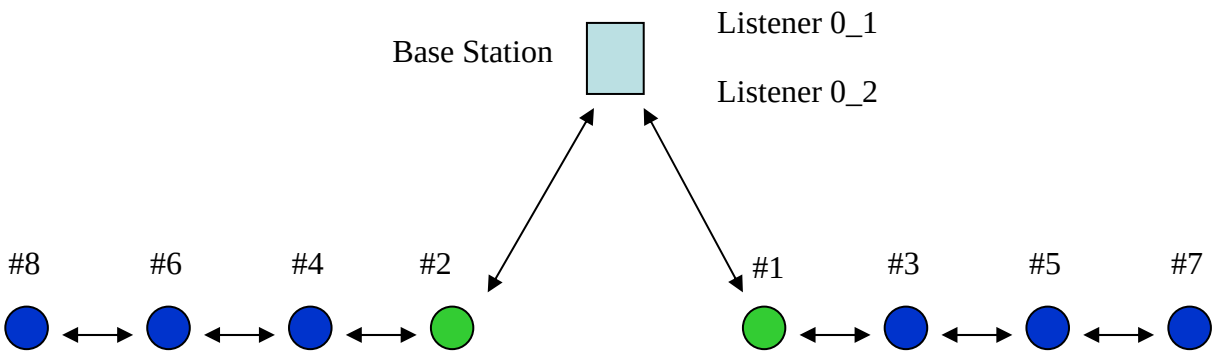

Fig. 4.1 Implemented Wireless Mesh Network

In the Fig. 4.1 node #5 has a Data_Mon #5 that collects its own sensor data and forwards that data and the data broadcasted by node #7 to node #3. The base station has two listeners, Listener 0_1 and Listener 0_2. These two listeners receive the data from node #1 and node #2 respectively. So the data from all the nodes should be relayed to nodes #1 or #2 to reach the final destination to be processed.

A future expanded wireless mesh network shown in the Figure 4.2 is more resilient to node failure since the data can be collected using alternate paths [16]. In case a node failure occurs in the network, the node trying to broadcast data to that failed node

automatically looks for a different route by applying a special routing protocol to send the data to the destination. In the process of searching for a different route, it looks for the closest and energy efficient route [17].

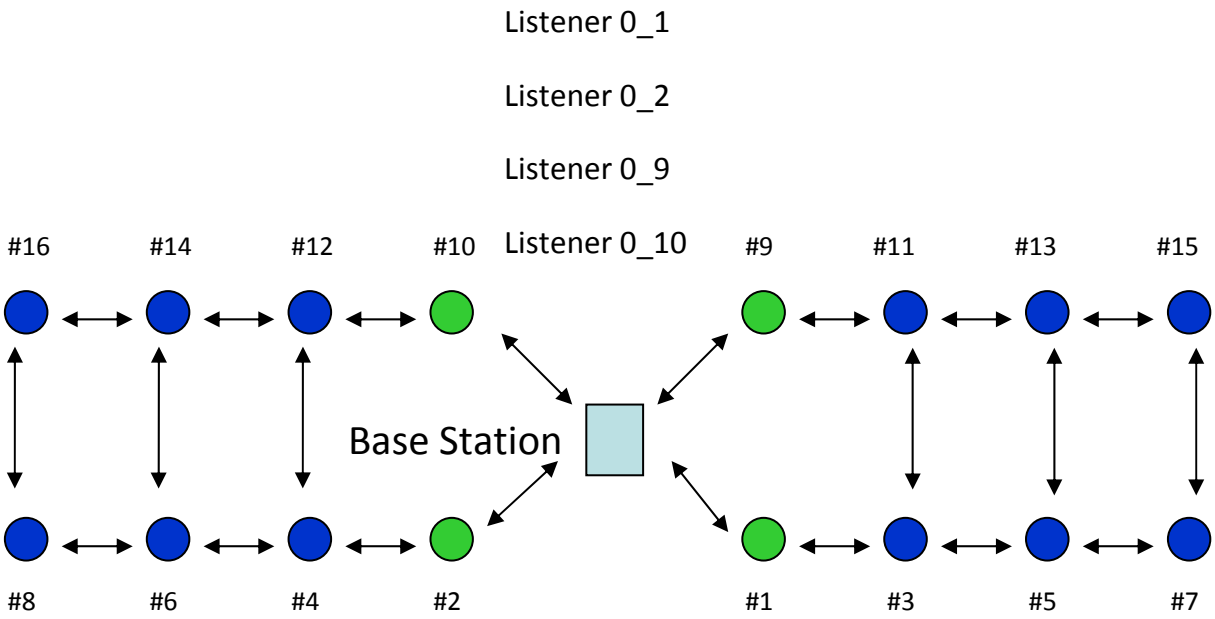

Fig. 4.2 Future Expanded Wireless Mesh Network

In the future expanded network eight more nodes are added to the existing network. Nodes should be deployed on the structure with less than 45 feet spacing between them as discussed in Chapter III. The scenario in case of a node failure in the network is explained below. For example, in Fig. 4.2 say node #13 is failed. Then node #15 has to find an alternative route to deliver its data to the base station. Two alternative routes exist in this case, one is #15 -> #7 -> #5 -> #3 -> #1, and other one is #15 -> #7 -> #5 -> #3 -> #11 -> #9. The routing protocol determines the route which is the best, in this

case the first one, because of the lower number of hops in the first route compared to the second one.

4.3 Routing Protocol

The routing protocol to be implemented for the future expanded wireless mesh network is NST-AODV (Not So Tiny – Ad-hoc On Demand Distance Vector Routing). This protocol is an adaption of the AODV [18]. In a traditional AODV approach the node transmitting the data initiates a route discovery procedure by broadcasting RREQ (Route Request) messages. In NST-AODV it is done using the data packet itself. The destination generates the RREP (Route Reply) messages. An intermediate node can also generate a RREP message if it knows the route to the demanded destination. The routing metric used in this protocol is hop count. When a link break is detected, the upstream node detecting the link failure generates a RERR (Route Error) message and broadcasts it locally. Then a local repair can be performed to deliver the data packet to the destination. In this protocol LLN (Link Layer Notification) mechanism is used for connectivity management. Usually data packets are transmitted to the link layer. After an unsuccessful link layer transmission took place. Two retries triggered by layer three can be performed. Three unsuccessful attempts to transmit the data packet lead to the link failure detection. Then the packet will be buffered and transmitted if a new route can be found.

Two FIFO (First In First Out) queues are used in this protocol. First queue is to save the incoming data packets during an on-going route discovery. The second one is an output queue. NST-AODV when compared to other AODV protocols decreases data

delivery latency, increases reliability of the network and is power efficient [19]. Lot of other protocols broadcasts Hello messages for connectivity management wasting lot of power. In NST-AODV intermediate nodes deliver RREP messages only if they know the destination unlike other protocols which deliver RREP to the source node for every RREQ they hear.

4.4 Experimental Procedure

A bridge structure is used to test the wireless sensor network. As discussed in Chapter II the sensor used for the network has a built-in accelerometer, which records the acceleration from the bridge deflections. One of the local bridges (on Market St, Youngstown, OH USA) was selected for testing the network.

Fig. 4.3 Wireless sensor network deployed on bridge

Four sensor nodes were deployed on the bridge with five meters spacing between them. Figure 4.3 shows the picture of the sensor network deployed on the sidewalk of bridge. The accelerometer readings are displayed by a graph on GUI (Graphic User Interface) frame. These readings can be imported to the spread sheet for further analysis and estimating the health of the structure. Figure 4.4 shows the screenshot of the GUI frame.

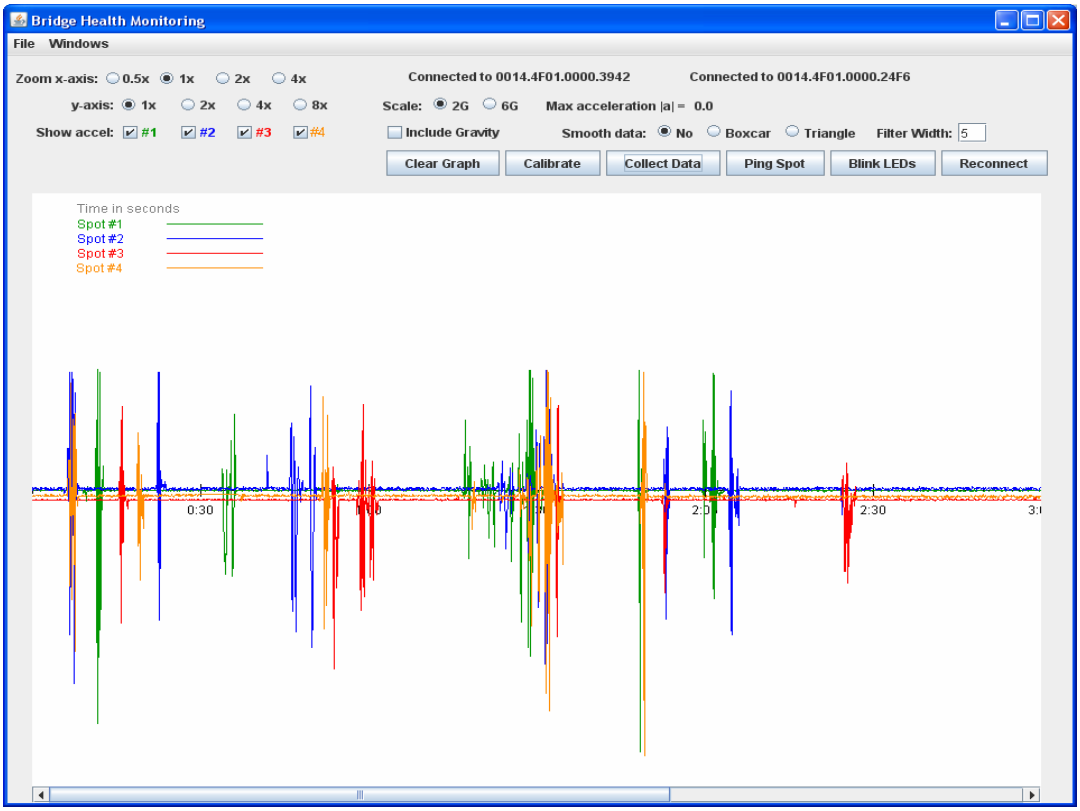

Fig. 4.4 Screenshot of the GUI frame

Figure 4.4 show that in the GUI frame we can see the number of nodes connected to the base station. The accelerometer readings from the Z-axial of the four nodes are displayed in four different colors. The GUI frame also give's options for the user to start, stop recording the data, reconnect to the nodes, ping the SPOT to see if the connectivity

is good and also to clear the graph. In case of very minute deflections in the structure we can also zoom the graph up to eight times.

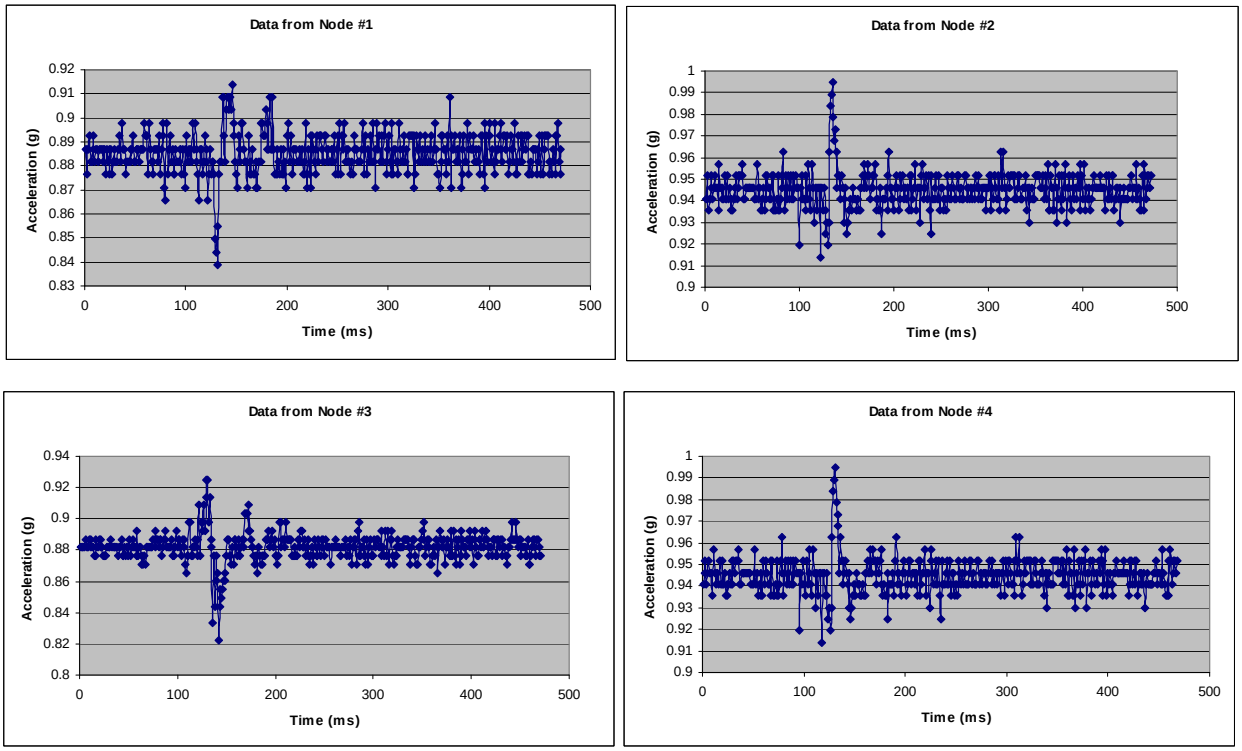

Fig. 4.5 Accelerometer data from four nodes

In figure 4.5 the deflection readings on a graph of the bridge from four different nodes can be seen. The data is recorded for about 47 sec. The nodes were programmed to record the reading for every 100ms. The peaks in the graph occurred at approximately 140 ms are caused by a truck which drove by at that time. Other minor deflections seen in the graph are from minor deflections caused by vehicles passed on the bridge at that time instance.

CHAPTER V

POWER MANAGEMENT

5.1 Power Saving Modes

Basically there are two power saving modes for a Sun SPOT. The first one is a shallow sleep and second one is deep sleep. Deep sleep mode can be enabled or disabled by the programmer depending on the particular power conservation requirements. But the Java thread scheduler inside the VM will choose whether to deep sleep or shallow sleep. The basic difference between the two modes is the Sun SPOT hardware to be turned off during these modes. Table 5.1 gives the list of hardware devices and their power mode for both the power saving modes.

Sun SPOT shallow sleep's whenever all the threads are idle. The decision when to shallow sleep is taken by the Java thread scheduler. Even though not a lot of hardware is turned off during shallow sleep, a reasonable amount of power is saved. Further saving the power in the shallow sleep is discussed in the next section of this chapter. Sun SPOT returns from the shallow sleep whenever an interrupt occurs from the hardware of the SPOT. Even in shallow sleep Sun SPOT responds to the external events because not a lot of hardware is off in this mode. Nodes resume from the sleep without any latency and function as they did before going to sleep. As suppose to do for the deep sleep, no enabling or disabling is done to shallow sleep.

Table 5.1: Hardware power modes during sleep

Hardware	Shallow sleep	Deep sleep
CPU	Power on but clock stopped	Power off
Master system clocks	on	off
Low-level firmware	on	on
RAM	Power on but inactive	Main power off, contents preserved by low power standby supply
Flash memory	Power on but inactive	off
CC2420 radio	on	off
AT 91 peripheral clocks	on if necessary	off
External board	on	Main power off but standby supply available

As discussed earlier java thread scheduler choose whether to deep sleep or shallow sleep. This decision is made based on the duration since it takes some time to wake up from the deep sleep. During the deep sleep, power to most of the hardware is turned off which are supposed to be active in order to interact with the device driver. In order for the java thread scheduler to decide to deep sleep, all the threads must be blocked and at least one thread should be on a timed wait with a minimum deep sleep

time including the wake up time. In case deep sleep time is less than the minimum sleep time or if a driver decides switching off its associated hardware would cause problems, a shallow sleep is performed instead of a deep sleep. During the deep sleep the primary power supply of the Sun SPOT is switched off only low-level firmware and the RAM are powered on. The wake up from the deep sleep is done by programming the low-level firmware to the necessary time.

5.2 Further Power Conserving Techniques

Power conserving has always been a priority in Wireless Sensor Networks. The power conserving techniques should always be implemented to reduce power consumption in battery powered wireless sensor nodes [20-21]. One of the techniques to further conserve the power in shallow sleep is by controlling the speed of the MCK (Master Clock). The default speed of the MCK is 60 MHz which have an interrupt latency time of 81us and draws a current of 22.8 mA. The MCK speed can be reduced resulting in further increase in interrupt latency time and decrease in usage of power. MCK speed can be put to a minimum value of 9.216 MHz having a latency time of 574 us and drawing a current of 10 mA. However it depends on the application running in the SPOT to figure out which speed best suits it. An experiment would be run at different speeds of MCK to pick the suitable speed. This has to be done to ensure the latency that occurs with the MCK speed is not an issue for some drivers with that application. The MCK speed cannot be changed if there are any timers or counters running, because they will not keep to time. The other power conserving technique is to

decrease the radio transmit power when there is better reception between the nodes or when the nodes are placed close to each other. The reception between the nodes can be found out by looking at the RSSI of the signal between nodes as explained in chapter III. Usually the radio transmit power is put to its maximum of 0 dBm for most applications or depending on the distance between nodes. So, there is a chance that power can be conserved by reducing the radio transmit power depending on the situation.

5.3 With an External Battery

An external battery can be connected to the Sun SPOT with a USB cable to power the SPOT for much longer time. There is variety of rechargeable external batteries available in the market which range in power and price. One thing that has to be noticed when using an external battery is SPOT could not be put to deep sleep since the USB host wants the device plugged into the USB port to reply to the requests on the USB bus. A Li-ion 3400 mAh rechargeable USB external battery was used while testing the network to additionally power the nodes. This lasts approximately four times longer than the battery which comes with the Sun SPOT. The lasting times of both the batteries for different modes of SPOT are compared in Table 5.2. The power consumption of the SPOT for the run mode is 120 mA, while in the run mode with the application board running is 400 mA and while in the idle mode it is 24 mA. The least power consumption is while the SPOT is in deep sleep mode which is 32 uA. But the deep sleep cannot be implemented with an external battery connected as explained earlier, so the deep sleep mode is not listed in the table.

Table 5.2 Battery operation times for different modes of SPOT

Different Modes	**Sun SPOT Battery**	**External Battery**
Run Mode	6 hr 30 min	28 hr 15 min
Run mode with application board on	1 hr 50 min	8 hr 30 min
Idle mode	32 hr	141 hr

Deciding whether to connect an external battery for the SPOT or let it run on Sun SPOT internal battery depends on the specific application running in it. If the application allows the SPOT to deep sleep for a considerable amount of time then external battery is not recommended. Because the SPOT battery will only supply 32 uA in deep sleep, it is better to hold off from buying an external battery and keep draining it without allowing the SPOT to deep sleep. If the application demands the SPOT to be active most of time it is recommended to have an external battery to stay powered longer.

CHAPTER VI

CONCLUSION AND FUTURE WORK

6.1 Conclusion

A low-cost, reliable, easy to deploy real time wireless sensor network is presented. The sensor network is a mesh network of sensor nodes capable of measuring acceleration and temperature at any point on the structure. The reliability of the network is achieved because of very few chances of loosing data due to the alternate route delivery techniques adopted and also because of the sensor key security and numerical encryptions implemented by the sensor node. Large amounts of data can be collected for accurate estimation of structural health using this network because of the high data sampling rate. More applications can be run on the same node depending on the requirement. Changes can be made over the air without actually visiting the site.

The wireless sensor network presented is designed to be left unattended on a structure for at least a week. This is achieved by implementing various power conservation techniques and using an external battery for demanding applications. Special routing techniques were proposed for the future expanded network for faster delivery of data and to save power. The network is flexible and can be deployed on any type of structure regardless of shape.

6.2 Future Work

The present sensor network of 16 nodes can be further expanded in future depending on the size of structure. These nodes could be programmed to be smart sensor nodes. As smart nodes they can be enabled to automatically scan the available wireless channels and pick the best channel (one with good reception) to communicate with the neighboring nodes and this could be tested. They should also be able to automatically set the radio output power to meet the minimum requirement of the signal strength coordinating with the communication channel. Emphasis can be put on the security protocols of the networks which monitor the sensitive data.

The placement distance between two nodes can be further increased in low interference environment and steps should be taken to improve the performance of the nodes in interference. Effort should be put on further increasing the operating time of the nodes without attending from more than a week. Possible chances of an onboard rechargeable battery should be explored. Since the nodes have analog inputs more sensors (like a humidity sensor) can be added if needed for the application.

APPENDIX A

DATA MONITORING

This section contains the java code for Data _Mon part of the wireless sensor node.

```
public class AccelMonitor extends PeriodicTask implements PacketHandler,
PacketTypes {

   // Definitions for raw accelerometer packets
   private static final int ACC_HEADER_SIZE = 8 + 1;   // time (long) + num samples (byte)
   //private static final int ACC_SAMPLE_SIZE = 4 * 2;   // delta t + acc_x + acc_y + acc_z (all
shorts)
   private static final int ACC_SAMPLE_SIZE = 2 * 3;   // delta t + acc_z01 + acc_z03 (all
shorts)
   private static final int ACC_SAMPLES_PER_PACKET =
         (PacketTransmitter.SINGLE_PACKET_PAYLOAD_SIZE - ACC_HEADER_SIZE) /
ACC_SAMPLE_SIZE;

   private LIS3L02AQAccelerometer acc;
   private ITriColorLED leds[];
   private int index = 0;
   private byte[] packetHdr = { ACCEL_2G_DATA_REPLY, ACCEL_6G_DATA_REPLY };

   private TelemetryMain main;
   private int sampleInterval;     // in milliseconds

   private PacketTransmitter xmit;
   private Radiogram currentPkt = null;
   private int currentSample = 0;
   private long startTime;

   /**
    * Create a new accelerometer controller.
    *
    * @param m reference to the main program getting commands from the host display
    * @param sampleInterval how often to sample the accelerometer, in milliseconds
    */
   public AccelMonitor(TelemetryMain m, int sampleInterval) {
      super(3, sampleInterval, Thread.MAX_PRIORITY);
      this.sampleInterval = sampleInterval;
      main = m;
      leds = EDemoBoard.getInstance().getLEDs();
```

```
      acc = (LIS3L02AQAccelerometer)EDemoBoard.getInstance().getAccelerometer();
      acc.setScale(LIS3L02AQAccelerometer.SCALE_2G);        // start using 2G scale
      index = 0;
   }


   /**
    * Get the PacketTransmitter & PacketReceiver to use to talk with the host.
    * Register the commands this class handles.
    *
    * @param xmit the PacketTransmitter to send packets
    * @param rcvr the PacketReceiver that will receive commands from the host and dispatch
them to handlePacket()
    */
   public void setPacketConnection (PacketTransmitter xmit, PacketReceiver rcvr) {
      this.xmit = xmit;

      rcvr.registerHandler(this, GET_ACCEL_INFO_REQ);
      rcvr.registerHandler(this, SET_ACCEL_SCALE_REQ);
      rcvr.registerHandler(this, CALIBRATE_ACCEL_REQ);
      rcvr.registerHandler(this, SEND_ACCEL_DATA_REQ);
      rcvr.registerHandler(this, STOP_ACCEL_DATA_REQ);
   }

   /**
    * Callback from PacketReceiver when a new command is received from the host.
    * Note only the commands associated with the accelerometer are handled here.
    *
    * @param type the command
    * @param pkt the radiogram with any other required information
    */
   public void handlePacket(byte type, Radiogram pkt) {
      try {
         switch (type) {
            case GET_ACCEL_INFO_REQ:
               getAccInfo();
               break;
            case SET_ACCEL_SCALE_REQ:
               setScale(pkt.readByte());
               leds[1].setRGB(0,30, is2GScale() ? 0 : 30);
               leds[1].setOn();        // green = 2G, blue-green = 6G
               Utils.sleep(200);
               leds[1].setOff();
               break;
            case CALIBRATE_ACCEL_REQ:
               if (!isRunning()) {
                  leds[1].setRGB(0,0,50);     // Blue = calibrating
                  leds[1].setOn();
                  calibrate();
                  leds[1].setOff();
               }
```

```
            break;
         case SEND_ACCEL_DATA_REQ:
            start();
            leds[1].setRGB(0, 30, is2GScale() ? 0 : 60);
            leds[1].setOn();        // green = 2G, blue-green = 6G

            break;
         case STOP_ACCEL_DATA_REQ:
            stop();
            leds[1].setOff();
            break;
      }
   } catch (IOException ex) {
      main.closeConnection();
   }
}

/**
 * Is the accelerometer using the 2G scale?
 *
 * @return true if the accelerometer using the 2G scale
 */
public boolean is2GScale () {
   return (index == 0);
}

/**
 * Send a packet to inform host of current accelerometer settings.
 * Tell if scale is 2G or 6G. Send zero offsets, gains & rest offsets.
 */
public void getAccInfo() {
   try {
      Radiogram dg = xmit.newDataPacket(GET_ACCEL_INFO_REPLY);
      dg.writeByte((is2GScale() ? 2 : 6));
      double offsets[][] = acc.getZeroOffsets();
      for (int i = 0; i < 2; i++) {
         for (int j = 0; j < 3; j++) {
            dg.writeDouble(offsets[i][j]);
         }
      }
      xmit.send(dg);
      dg = xmit.newDataPacket(GET_ACCEL_INFO2_REPLY);
      offsets = acc.getGains();
      for (int i = 0; i < 2; i++) {
         for (int j = 0; j < 3; j++) {
            dg.writeDouble(offsets[i][j]);
         }
      }
      xmit.send(dg);
      dg = xmit.newDataPacket(CALIBRATE_ACCEL_REPLY);
      offsets = acc.getRestOffsets();
```

```
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 3; j++) {
                dg.writeDouble(offsets[i][j]);
            }
        }
        xmit.send(dg);
    } catch (IOException ex) {
        // ignore errors - display server can repeat request if need be
    }
}

/**
 * Set the accelerometer to use either the 2G or 6G scale.
 * Will send a packet to acknowledge the request.
 * The reply will include the current scale (2 or 6) or
 * be 0 if an invalid scale was requested.
 *
 * @param b the scale to use = 2 or 6
 */
public void setScale(byte b) {
    try {
        Radiogram dg = xmit.newDataPacket(SET_ACCEL_SCALE_REPLY);
        if (b == 2) {
            index = 0;
            acc.setScale(0);
            dg.writeByte(2);
        } else if (b == 6) {
            index = 1;
            acc.setScale(1);
            dg.writeByte(6);
        } else {
            dg.writeByte(0);
        }
        xmit.send(dg);
    } catch (IOException ex) {
        // ignore errors - display server can repeat request if need be
    }
}

/**
 * Have the accelerometer calculate the current rest offsets.
 * Send a packet back to the host with the 6 offset values.
 */
public void calibrate () {
    try {
        Radiogram dg = xmit.newDataPacket(CALIBRATE_ACCEL_REPLY);
        acc.setRestOffsets();
        double offsets[][] = acc.getRestOffsets();
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 3; j++) {
                dg.writeDouble(offsets[i][j]);
```

```
            }
        }
        xmit.send(dg);
    } catch (IOException ex) {
        // ignore errors - display server can repeat request if need be
    }
}


////////////////////////////////////
//
// PeriodicTask overridden methods
//
////////////////////////////////////

/**
 * Routine called when task execution is about to start up.
 */
public void starting() {
    currentPkt = null;
    System.out.println("Starting accelerometer sampler");
}

/**
 * Routine called when task execution is finished.
 */
public void stopping() {
    try {
        if (currentPkt != null) {
            double offsets[][] = acc.getZeroOffsets();
            int deltaT = (int) (System.currentTimeMillis() - startTime);
            for (int i = currentSample; i < ACC_SAMPLES_PER_PACKET; i++) {
                currentPkt.writeShort(deltaT + i);
                for (int j = 0; j < 3; j++) {
                    currentPkt.writeShort((int)offsets[index][j]);   // fill out final packet
                }
            }
            xmit.send(currentPkt);
            currentPkt = null;
        }
    } catch (IOException ie) {
        // ignore
    }
    System.out.println("Stopping accelerometer sampler");
}


/**
 * Called once per task period to pack up accelerometer readings.
 * When the packet is full it is queued for transmission.
 */
```

```
    public void doTask() {
        try {
            if (currentPkt == null) {
                startTime = System.currentTimeMillis();
                currentPkt = xmit.newDataPacket(packetHdr[index]);
                currentPkt.writeLong(startTime);
                currentPkt.writeByte(ACC_SAMPLES_PER_PACKET);
                currentSample = 0;
            }
            currentPkt.writeShort((int) (System.currentTimeMillis() - startTime));
            //currentPkt.writeShort(acc.getRawX());
            //currentPkt.writeShort(acc.getRawY());
            currentPkt.writeShort(acc.getRawZ());
            currentPkt.writeShort(main.SP3_raxz);
            if (++currentSample >= ACC_SAMPLES_PER_PACKET) {
                xmit.send(currentPkt);
                currentPkt = null;
            }
        } catch (IOException ie) {
            main.queueMessage("IO exception: " + ie.toString());
        }
    }

    /** temporary fix until IService interface fixed */
    public void setName(String who){};
}
```

REFERENCES

[1] Y. M. Gebremichael, W. Li, B. T. Meggitt, W. J. O. Boyle, K. T. V. Grattan, B. McKinley, L. F. Boswell, K. A. Aames, S. E. Aasen, B. Tynes, Y. Fonjallaz, and T. Triantafillou, "A field deployable multiplexed Bragg grating sensor system used in an extensive highway bridge monitoring evaluation tests," IEEE Sensors J., vol. 5, no. 3, pp. 510-519, June 2005.

[2] Chee-Yee Chong, S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges". Proceedings of the IEEE, vol. 91, no. 8, Aug 2003.

[3] R. Rashid, G. Robertson, "Accent: A communication oriented network operating system kernel," in proceedings of 8th symposium of Operating System Principles, 1981, pp. 64-75.

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

[5] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, D. Estrin. "A Wireless Sensor Network for structural monitoring". Proceedings of the ACM Conference on Embedded Networked Sensor Systems, Baltimore, MD, USA, November 2004.

[6] Randall P. Smith, "SPOT World and the Sun SPOT". Proceedings of the 6th international conference on Information processing in sensor networks, Cambridge, MA, USA, 2007.

[7] D. Simon, C. Cifuentes, D. Cleal, J. Daniels, and D. White, "Java(TM) on the Bare Metal of Wireless Sensor Devices – the Squawk Java Virtual Machine". In VEE, Ottawa, July 2006.

[8] The Institute of Electrical and Electronics Engineers, Inc. "IEEE Std 802.15.4, IEEE Standard for Information technology-Telecommunications and Information exchange between systems-Local and metropolitan area networks- Specific requirements".

[9] D. Acharya, V. Kumar, N. Garvin, A. Greca, G.M. Gaddis, “A Sun SPOT based automatic vehicular accident notification system”. International conference on Technology and applications in Biomedicine. Shenzhen, May 2008.

[10] Nolten, U.; Mokwa, W., "Strain gauge foil for the measurement of elastic deformations in orthopedic milling tools," Sensors, 2008 IEEE , pp.1476-1479, 26-29 Oct. 2008.

[11] Francesco, C.; Giannetti, R., "*A new approach to strain gauge selection in outdoor stress analysis* ," Instrumentation and Measurement Technology Conference, 1997. IMTC/97. Proceedings. 'Sensing, Processing, Networking'., IEEE , vol.2, pp.1094-1097, 19-21 May 1997.

[12] Haksoo Choi; Sukwon Choi; Hojung Cha, "Structural Health Monitoring system based on strain gauge enabled wireless sensor nodes," Networked Sensing Systems, 2008. INSS 2008. 5th International Conference on, vol., no., pp.211-214, 17-19 June 2008.

[13] C. Soras, M. Karaboikis, G. Tsachtsiris, V. Makios, “Analysis and Design of an Inverted-F Antenna Printed on a PCMCIA Card for the 2.4 GHz ISM Band”, IEEE Antenna’s and Propagation Magazine, Vol. 44, No. 1, February 2002.

[14] P. W. Chan, H. Wong, E. K. N. Yung, “Dual-Band Printed Inverted-F Antenna for DCS, 2.4 GHz WLAN Applications”, Antennas & Propagation Conference, Loughborough, UK March 2008.

[15] L. Frye, Liang Cheng, Shenfu Du, M.W. Bigrigg, “Topology Maintenance of Wireless Sensor Networks in Node failure-Prone environments”, Proceedings of the 2006 IEEE International conference in Networking, Sensing and Control, 2006. ICNSC ‘06.

[16] R. Lee, K. Chen, S. Chiang, C. Lai, H. Liu, and M. Wei, “A Backup Routing with Wireless Sensor Network for Bridge Monitoring System,” Proceedings of the 4th Annual Communication Networks and Services Research Conference, April 2006.

[17] Richard Draves, Jitendra Padhye, Brian Zill, “Routing in multi radio, multi hop wireless mesh network”. International Conference on Mobile Computing and Networking, Philadelphia, Pa, 2004. pp.114-128.

[18] C. Perkins, E. Belding-Royer, S. Das, “Ad Hoc On Demand Distance vector Routing (AODV)”, RFC 3561, July 2003.

[19] C. Gomez, P. Salvatella, O. Alonso, J. Paradells, “Adapting AODV for IEEE 802.14.5 mesh sensor networks: Theoretical discussion and performance evaluation in a real environment”. Proceedings of the 2006 International

Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06).

[20] N. Aakvaag, M. Mathiesen, G. Thonet, "Timing and Power issues in wireless sensor networks-an industrial test case". Proceedings of the 2005 International Conference on Parallel Processing Workshops, June 2005.

[21] Jae-Hwan Chang, L. Tassiulas, "Maximum lifetime routing in wireless sensor networks". Transactions on Networking, IEEE/ACM, Aug 2004.