NSEL Report Series
Report No. NSEL-018
August 2009

Flexible Smart Sensor Framework for Autonomous Full-scale Structural Health Monitoring

Jennifer A. Rice
and
Billie F. Spencer, Jr.

NSEL
NEWMARK STRUCTURAL ENGINEERING LABORATORY

Department of Civil and Environmental Engineering
University of Illinois at Urbana-Champaign

UILU-ENG-2009-1806

ISSN: 1940-9826

Contact:
Prof. B.F. Spencer, Jr.
Director, Newmark Structural Engineering Laboratory
2213 NCEL, MC-250
205 North Mathews Ave.
Urbana, IL 61801
Telephone (217) 333-8630
E-mail: bfs@illinois.edu

This technical report is based on the first author's doctoral dissertation of the same title, which was completed in July 2009. The second author served as the dissertation advisor for this work.

Financial support for this research was provided in part by the National Science Foundation under NSF grants CMS 03-01140 and CMS 06-00433 (Dr. S. C. Liu, Program Manager) and Intel Corporation Research. The first author was supported by a Vodafone-U.S. Foundation Graduate Fellowship and the CEE Alumni Teaching Fellowship. Finally, we would like to gratefully acknowledge the efforts of numerous collaborators in this work, including Gul Agha, Chung-Bang Yun, Hyung-Jo Jung, Tomonori Nagayama, Yozo Fujino, Kirill Mechitov, Shinae Jang, Sung-Han Sim, Lauren Linderman, Ryan Giles, Soojin Cho, Jongwoong Park, Robin Kim, Mariella DeFino, Lama Nachman, Robbie Adler, Ralph Kling, and Aaron Headley.

The cover photographs are used with permission. The Trans-Alaska Pipeline photograph was provided by Terra Galleria Photography (http://www.terragalleria.com/).

ABSTRACT

The demands of aging infrastructure require effective methods for structural monitoring and maintenance. Wireless smart sensors provide an attractive means for structural health monitoring (SHM) through the utilization of onboard computation to achieve distributed data management. Such an approach is scalable to the large number of sensor nodes required for high-fidelity modal analysis and damage detection. While much of the technology associated with smart sensors has been available for nearly a decade, there have been limited numbers of full-scale implementations due to the lack of critical hardware and software elements. This research develops a flexible smart sensor framework for full-scale, autonomous SHM that integrates the necessary software and hardware while addressing key implementation requirements. The Imote2 smart sensor platform is employed herein, providing for the first time the enhanced computation and communication resources that support demanding sensor network applications such as SHM of civil infrastructure. A multimetric Imote2 sensor board with onboard signal processing specifically designed for SHM applications has been designed and validated. Flexible network management software combines a sleep/wake cycle for enhanced power efficiency with threshold detection for triggering network wide operations such as synchronized sensing or decentralized modal analysis. A cable-stayed bridge in South Korea serves as one of the test beds for this effort, both informing and driving system development. This research has resulted in the first autonomous, full-scale implementation of a wireless smart sensor network for structural health monitoring.

CONTENTS

Chapter 1

INTRODUCTION

Civil infrastructure is the foundation of our society and has a widespread impact on the quality of our daily lives. Monitoring the safety and functionality of the world's buildings, bridges, and lifeline systems, is critical to improving maintenance practices, minimizing the cost associated with repair and ultimately improving public safety. Structural health monitoring (SHM) provides the means for capturing structural response and assessing structural condition for a variety of purposes. For example, the information from an SHM system can be used to fine-tune idealized structural models, thereby allowing more accurate prediction of the response due to extreme loading conditions, such as an earthquake. SHM also can be used to characterize loads in situ, which can allow the detection of unusual loading conditions as well as validate the structure's design. In addition, real-time monitoring systems can measure the response of a structure before, during and after a natural or man-made disaster, that can be used in damage detection algorithms to assess the post-event condition of a structure.

While the sensors and data acquisition system required to measure structural response are not new technologies, more of an effort in recent years has been directed toward use of this data to assess the current state of a structure. These algorithms take the measured structural response along with varying degrees of information regarding the structural model and the input excitation and attempt to determine if the structure has sustained measurable changes in its condition. Analyzing the measured data in this way is useful for both periodic structural monitoring to track the state of a structure over time as well as for the assessment of a structure following a strong loading event such as an earthquake. In both cases, the result is the ability to implement evacuation, repair and retrofit strategies that ultimately improve public safety and limit the life-cycle cost of the structure.

Gaining a clear understanding of structural behavior to allow a reasonable assessment of its as-built condition requires high-fidelity sensor data to build accurate models. In addition, potentially problematic structural changes, such as corrosion, cracking, buckling, fracture, etc., all occur locally within a structure. It is expected that sensors must be in close proximity to the damage to capture the resulting change in response while sensors further from the damage are unlikely to observe measurable changes. To achieve an effective monitoring system that is capable of generating informative structural models and detecting critical structural changes, a dense array of sensors will be required. Due to the cost of deployment and the potential for data inundation, such a dense instrumentation system is not practically realized with traditional structural monitoring technology.

Traditional structural monitoring systems are comprised of a network of sensors distributed throughout a structure. These networks typically rely on a central source of power and data acquisition and therefore require cables to link the sensors with the power and acquisition hardware hub. Implementing modal analysis or damage detection algorithms with wired systems requires all of the sensed data to be collected and the data acquisition center where it is then processed. For a dense array of sensors sampling at the rates required for SHM the result is an enormous amount of data to be communicated and processed at a single location.

Beyond this concern for data inundation, traditional wired monitoring systems can be extremely costly. Çelebi (2002) estimates the cost per sensor channel (including sensor, data acquisition and installation) in the range of $4K for monitoring systems with 12 to 18 channels. The Bill Emerson Memorial Bridge in Missouri is instrumented with 84 accelerometer channels with an average cost per channel of over $15K, including installation (Çelebi et al. 2004). The number of SHM systems is increasing as it has become common practice to install them during the construction of critical and large-scale structures. Some of these systems have pushed the bounds of the number of wired sensor channels that can be achieved.

Advances in wireless technology and embedded processing have made much lower-cost wireless smart sensor networks an attractive alternative to wired data acquisition systems. These sensors reduce the overall expense of implementation, because the time, and therefore cost, associated with installation is greatly reduced. The majority of the work using wireless sensors for structural monitoring has focused on using the sensors to emulate traditional wired sensor systems. As these systems require that all data be sent back to a central processing center, the amount of wireless communication required in the network becomes costly in terms of excessive communication times and the associated power it consumes. For example, a wireless sensor network implemented on the Golden Gate Bridge that generated 20MB of data (1600 seconds of data, sampling at 50Hz from on 64 sensor nodes) took over 9 hours to complete the communication of the data back to a central location (Pakzad et al. 2008).

Wireless smart sensors networks (WSSNs) leverage onboard computational capacity on the wireless sensors to allow data processing to occur within the network, as opposed to at a central location. By implementing data processing techniques, such as modal analysis or damage detection algorithms, in such a distributed manner, the amount of communication that occurs within the network can be reduced, while providing usable information on the structural condition. WSSNs employing decentralized computing have the potential to dramatically improve SHM efforts.

Many challenges arise when implementing a WSSN for structural monitoring that are not present in wired systems. The design of a wireless monitoring system employing a large number of sensors must account for the limited resources of smart sensors, time synchronization, limitations in the types and quality of sensors available, data loss associated with RF communication, network fault tolerance, etc. Additionally, vibration-based SHM requires sensed data that provides a true representation of the structural response, both in amplitude and phase, over a wide bandwidth, while eliminating aliased signals. Communication hardware and protocols must ensure minimal data loss to preserve the quality of the sensed data and the robust performance of the network. Continuous network operation relies on a system with many built-in fault tolerance features to sustain its functionality, even in the presence of hardware or software failures.

In recent years, researchers have made progress toward addressing the inherent road blocks to realizing SHM application that utilize WSSNs. Nagayama and Spencer (2007) successfully implemented a distributed SHM system on a smart sensor network on a laboratory scale truss structure. A few successful full-scale implementations of wireless smart sensor networks for SHM have been completed in recent years; however, most of these systems have simply emulated their wired counterpart, resulting in significant scalability issues. To date, the hardware and software required for SHM have yet to be integrated to achieve a framework that is suitable for autonomous monitoring and distributed data management on a full-scale structure employing a dense network of sensors.

The objective of this research is to provide an enabling WSSN framework to address issues that have limited their effectiveness for SHM systems. In particular, this research addresses the following three aspects of realizing distributed SHM using WSSNs: 1) The development and validation of an enabling, open-source software framework that is modular and adaptable, 2) the development of flexible, multimetric sensors for use in a WSSNs with user-selectable anti-aliasing filters to produce high-quality data appropriate for SHM applications, and 3) the consideration of key implementation issues, such as realizing optimal communication configurations and achieving autonomous network operation while maintaining power efficiency. The integration of these software and hardware components has resulted in a flexible framework that enables autonomous, full-scale implementations of SHM systems. The following paragraphs outline the contents of the research presented in this report.

Chapter 2 provides background on the field of SHM and demonstrates the potential for WSSNs to transform the way effective structural monitoring is conducted. The key components of this technology are presented in the context of their implications on overall network functionality. The limitations in critical hardware and software elements towards realizing full-scale, autonomous SHM implementations are identified.

Chapter 3 presents a new software development architecture that addresses the complexity of application software development for SHM systems deployed on smart sensor networks. This service-oriented software framework provides modular components that may be linked together to build fully integrated SHM systems. The open-source nature of the software, along with detailed instructions and documentation, makes SHM using WSSN accessible to a broad audience.

Autonomous, full-scale network operation is enabled by the software components developed in Chapter 4. The flexible network management software combines a sleep/wake cycle for enhanced power management with threshold detection for triggering network wide operations such as synchronized sensing or decentralized modal analysis.

Chapter 5 presents the development of versatile, multimetric sensor hardware to be integrated with a commercially available smart sensor platform. The sensor board has been designed specifically for vibration-based SHM with flexible sampling rate, gain, and signal conditioning options, including user-selectable anti-aliasing filters. The design has been experimentally validated through static and dynamic testing.

Chapter 6 addresses issues critical to achieving an autonomous, full-scale network deployment: effective utilization of communication hardware and careful power management. A thorough investigation of the radio and antenna hardware is presented, resulting in recommendations for the optimal configuration to maximize communication performance. The implications of the selected hardware and application parameters on the overall power consumption are investigated and the network events with the greatest impact are identified. Because many WSSNs rely on battery power, battery life projections are presented and are experimentally verified by results presented in Chapter 7.

Chapter 7 presents three validation studies that are enabled by the hardware, software and implementation considerations addressed throughout this report. The final, and ongoing, deployment at the Jindo Bridge, in South Korea, demonstrates how each of the components of the WSSN framework developed in this research have come together to create the first fully integrated, large-scale, and autonomous deployment of smart sensors for structural monitoring.

Chapter 8 provides a summary of the research presented in this report and discusses potential future studies to continue the advancement SHM using smart sensors.

BACKGROUND

This chapter provides the background information and motivation for the research presented in this report. First, an introduction and history of the field of structural health monitoring (SHM) is given followed by previous research in the area of wireless smart sensors and how their technology can transform SHM practice. The background on wireless smart sensors includes a description of their various components, a comparison of existing wireless sensor platforms, discussion of their sensor hardware interface, and an introduction to the operating system used by most wireless sensor platforms. Next, wireless smart sensor network (WSSN) data processing methods and their implications are discussed followed by background on issues critical to networking and implementation, such as: time synchronization, communication, data aggregation and software development. Finally, a survey of previous implementations of WSSN for SHM applications is given. The ultimate goal of this chapter is to identify the hurdles that still remain to using wireless smart sensors for fully-integrated, autonomous monitoring of full-scale civil infrastructure systems.

2.1 Structural health monitoring

Structural health monitoring (SHM) is a term that encompasses a wide range of methods and practices aimed at assessing the condition of a structure based on a combination of observation, measurement, analysis, and modeling. Some of the motivation for carrying out SHM for civil infrastructure includes (Farrar and Doebling 1997, Farrar and Warden 2007, Brownjohn 2007):

1) Assessing the performance of the as-built structure
2) Model updating
3) Understanding complex load-response relationships
4) Assessing novel construction or design techniques
5) Evaluating the condition of a structure following an extreme loading event
6) Monitoring during construction
7) Evaluating retrofit measures
8) Monitoring long-term structural degradation or deterioration

The roots of SHM practices can be traced back to the industrial and aerospace industries. Bar-Cohen (1999) gives an overview of the emergence of non-destructive evaluation (NDE) methods for assessing the condition of materials and structural components. Many of these methods involve introducing a high-speed wave to the test subject and observing the response to determine its condition. As early as the 1960s, methods such as Eddy Current and radiography were introduced. The 1970s saw the emergence of acoustic emission and the improvement of ultrasonic technology. It was not until the 1980s that analysis methods reached the level of maturity required for the technologies to see more widespread use. Djordjevic (1990) discusses the use of NDE techniques for space structures as a means of providing critical information about their flight-worthiness prior to launch, during flight, and in post-flight inspection and maintenance. In particular, the concept of structural health monitoring using in-situ sensors is introduced.

Though early NDE research represents the origins of SHM, SHM has emerged as a separate and very important field. While NDE seeks to discover flaws at the material level, and is thus limited to very local damage detection, SHM encompasses a more global approach to the assessment of civil infrastructure. The size and complexity of civil structures demands such global methods for evaluating their performance and condition; information from small, limited portions of the structure may not provide a complete picture of the structural condition. Using a combined local-global strategy for monitoring civil infrastructure that investigates local damage in critical locations while observing the overall response and behavior of the structure, is expected to be most effective. Such a strategy, however, requires a dense array of sensors distributed throughout the structure to capture the local effects of damage.

Over the years, efforts to instrument and measure structural response have yielded valuable insight into structural design, construction, and response. Although the field of SHM has a history spanning several decades, engineers and researchers have yet to realize the one of its ultimate goals of being able to detect significant damage in a structure, regardless of the type of damage and class of structure, prior to catastrophic failure. While it is unreasonable to assume there is a universal approach to SHM that is appropriate for all structures subject to any type of damage, most SHM systems have the following common features:

1) A baseline assessment of the healthy structure
2) A measurement system to monitor the structural response or changes
3) Methods for processing the measured data to extract information on the current state of the structure

SHM of civil infrastructure can prolong the usable life of a structure, decrease total life-cycle cost, and most importantly, improve public safety. Although there are limitations to the depth of information that current SHM practices can provide, ongoing advances in sensing and data processing will lead to enhanced infrastructure management for a wide range of purposes.

2.1.1 Background and motivation

Regardless of the methods used, the need for effective SHM has become increasingly evident in recent years. In 2007 the Federal Highway Administration (FHWA) reported that over 13 percent of the bridges in the United States are structurally deficient or functionally obsolete. Figure 2.1 shows the number of deficient bridges in the US by the year they were built. Given that a sizeable increase in bridge construction began over 50 years ago and that 25 percent of bridges built during that period are already deficient, this figure clearly illustrates the growing issue of our aging infrastructure. The recent collapse of the I-35 Bridge in Minneapolis brought this already critical issue to the forefront of public and political attention.

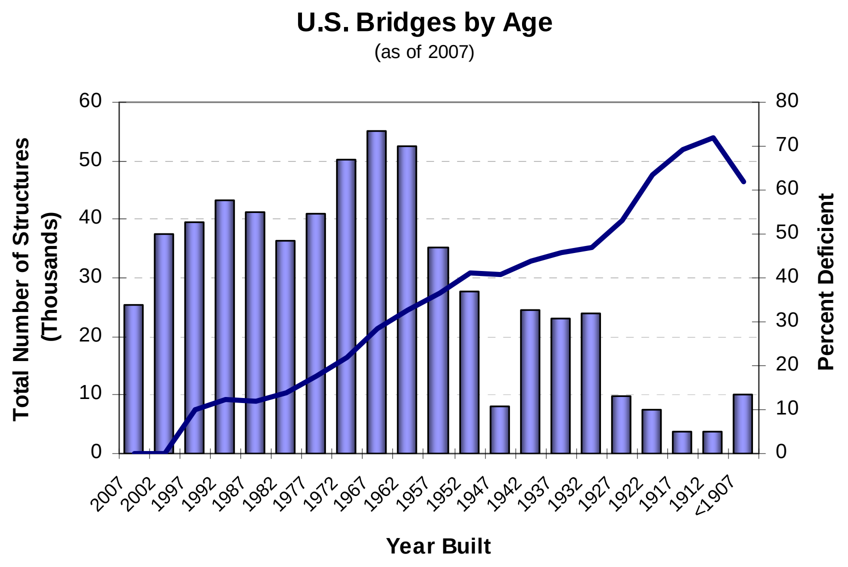


Figure 2.1 Deficient bridges in the US by year built (FHWA 2007).

Beyond visual inspection practices, efforts to monitor civil infrastructure were seen as early as 1937 when the vibrational characteristics of the Golden Gate Bridge were measured in an effort to predict its response to seismic excitation. Indeed, vibration-based approaches for damage detection and monitoring of civil infrastructure are the most widely proposed and investigated approaches.

Doebling, et al. (1996) provides a comprehensive review of vibration-based damage detection approaches, beginning in the 1970s with the investigation of their potential for assessing offshore structures and continuing to the more widespread applications proposed through the 1980s and early 1990s. In general terms, vibration-based methods are categorized by their damage-sensitive feature as follows:

4) Frequency (Salawu 1997)
5) Mode shape
6) Strain mode shape (mode shape curvature)
7) Dynamic Flexibility
8) Stiffness

In addition to detecting changes in the above characteristics, other methods are based on matrix updating and neural networks among others. Sohn (2003) provides a review of SHM literature from 1996 to 2001, looking at structural response measurement techniques and methods for detecting damage from the measured response, including statistical approaches. While a wide variety of vibration-based SHM approaches are presented, there are some common issues that limit their effectiveness, namely:

1) Measurement noise or inadequate signal-to-noise ratio
2) Discrepancy between the structural model and as-built structure
3) Non-linear structural response
4) Inadequate number of sensors
5) Structural complexity/redundancy

6) Separation of the influence of environmental factors from the existence of real damage

Brownjohn (2007) provides an overview of SHM for civil infrastructure that stresses the importance of a holistic approach to damage detection and condition assessment.

2.1.2 Full-scale applications

There are many examples of full-scale structural monitoring employing traditional wired systems over the past few decades (Salawu and Williams 1995, Doebling et al. 1998, Doebling and Farrar 1999, Sohn et al. 2003 and Brownjohn 2007) with widely varying motivations, sensor types, sensor density, data processing techniques, and outcomes. Some monitoring projects are only intended for short-term implementations with specific goals, while others are installed for long-term general monitoring purposes. SHM systems have been used in older structures, as well as new construction. In the last decade, installing monitoring systems in new bridges during construction (e.g. the Bill Emerson Memorial Bridge, the Tsing Ma Bridge and the Stonecutters Bridge) has become common practice. The following paragraphs provide some representative examples for each type of traditional SHM monitoring project. Systems employing wireless sensors will be discussed in a subsequent section.

The Humber Bridge

The Humber Bridge, a suspension bridge, is the longest span bridge in the UK and has been the subject of various monitoring programs since it was first instrumented in the 1980's. The purpose of its instrumentation was the validation of finite element modeling and wind response simulation. In total 63 sensors were deployed on the bridge, including wind sensors, accelerometers, and LVDTs. The measured data was used to evaluate the aeroelastic properties of the bridge as well as the dependence of modal parameters on wind excitation levels. The researchers concluded that while global measurements provide information to validate modeling and simulation, they are not expected to detect damage in the structure. The author proposes that short-term, targeted testing would be more appropriate to monitor the critical components of the bridge (Brownjohn 2007).

This bridge was more recently chosen as a test bed for wireless sensor networks for civil infrastructure applications. The purpose system was to monitor the temperature and relative humidity (RH) at the cable anchorage locations and provide the information in real-time via the web. The researchers encountered hardware difficulties, primarily in communication, that required attention (Hoult, et al. 2008).

The Golden Gate Bridge

The Golden Gate Bridge in the San Francisco, CA has been the subject of several monitoring efforts. Abdel-Ghaffar, et al. (1985) conducted ambient vibration studies of the bridge to determine natural frequencies, effective damping ratios and modes shapes in an attempt to understand and predict the bridge's response to wind and earthquake loading. The bridge was instrumented with 28 accelerometers with 12 on the main span, 6 on the side span, and 10 on the pier-tower structures. The measured data was analyzed, and the vibration modes were accurately identified and compared well to calculated results.

The Golden Gate Bridge has more recently been a test bed for the development and assessment of a wireless sensor network for structural monitoring. This study will be discussed in more detail in a subsequent section.

Alamosa Canyon Bridge

The Alamosa Canyon Bridge in New Mexico was instrumented for the purpose of evaluating the change in measured dynamic characteristics due to variable environmental factors (Farrar, et al.1997). This monitoring application was a short-term experimental evaluation and not intended for long-term monitoring. The bridge was instrumented with 31 accelerometers and 5 temperature sensors and was excited using an impact hammer. To capture natural temperature changes the response data was measured over a 24-hour time period during which 31 impact events and data records were recorded. Data processing included the application of the Eigensystem Realization Algorithm (ERA) in conjunction with a statistical approach to identify the modal properties of the bridge and their associated statistical boundaries. The modal properties of the bridge were accurately determined; however, a fluctuation of approximately 5 percent was observed in these parameters over a 24-hour period due to temperature changes.

Bill Emerson Bridge

The Bill Emerson Memorial Bridge in Cape Girardeau, Missouri, was instrumented in an effort to provide seismic response data, validate models, and gain insight for future cable stayed bridge designs (Caicedo et al. 2002, Celebi 2006). The long-term monitoring system was intended to provide real-time response data via the internet for wide dissemination. Eighty-four accelerometer channels were installed on the bridge, in addition to anemometers to measure wind velocities. The data from this monitoring project has been used for a variety of purposes, ranging from the evaluation of damage detection algorithms to seismic response evaluation and model updating techniques. The monitoring system remains active and is continuing to be used by researches to advance SHM efforts.

Tsing Ma Bridge

The Tsing Ma Bridge is a suspension bridge in Hong Kong. An integrated monitoring system has been installed on the bridge that consists of a variety of sensors and data acquisition, as well as data processing, and structural health evaluation systems (Wong 2004). The sensor system includes acceleration, displacement, strain, wind, and temperature measurements, as well as video cameras and GPS measurement stations, for a total of 326 data channels. This extensive monitoring system is intended for a variety of purposes related to SHM, including: 1) measuring loading sources, 2) determining system characteristics (e.g., global dynamic properties), and 3) measuring the system response (e.g. cable forces and fatigue assessment). The response of the bridge due to various loading conditions is recorded. Any observed abnormalities in the response are expected to alert of potential over loading. This long-term monitoring is still operational.

2.1.3 Challenges to wide-spread use of SHM

The cost of adequately instrumenting a large structure so that relevant information can be extracted has been shown to be high. The total cost of an SHM sensor network is derived not only from the large number and variety of sensors and expensive data acquisition equipment, but also the cost of installation. The second monitoring system installed on the Humber Bridge required 32 km of cabling for 63 sensors (Brownjohn 2007) and the cost of the 326 sensor channels installed on the Tsing Ma Bridge is estimated at $8 million (Lynch and Loh 2006). As engineers and researchers have sought more cost effective methods for SHM of civil structures, wireless smart sensors (WSS) have emerged as an attractive alternative to traditional wired sensors. Eliminating the need for physical power and data lines significantly reduces the cost associated with network installation and ultimately the total network cost (Lynch and Loh 2006, Spencer, et al. 2003).

Besides the prohibitive cost of traditional structural monitoring, there is the issue of the extreme data inundation that results from a large network of sensors. For example, the Tsing Ma and Kap Shui Mun Bridges in Hong Kong together produce approximately 63MB of data every hour (Wong 2004). This issue is the primary motivating factor for developing SHM strategies that employ data compression and information extraction to eliminate the need to send all of the raw data generated by a monitoring system. Smart sensors with onboard microprocessors have the potential to address the issue of data overload by allowing data processing to occur directly at the sensor node. The following section introduces these WSS in detail.

2.2 Wireless smart sensors

Advances in sensor technology, communication and embedded systems have led to the establishment and rapid development and improvement of wireless smart sensor (WSS) technology. Lynch and Loh (2006) provide an extensive and comprehensive review of wireless sensor technology for SHM applications. As previously mentioned, WSSs offer the potential to reduce the cost and improve the efficiency of SHM systems. This potential, however, is highly dependent on the features of the smart sensor, as well as the needs of the monitoring application.

2.2.1 Components of a wireless smart sensor

There are many commercially available WSS platforms, as well as a number of academic prototypes that have been developed. The common hardware components of WSSs include a radio, embedded computing, power, and sensor interface, and are discussed in the following paragraphs.

Radio

The defining feature of a WSS node is the fact that data is transmitted wirelessly with the use of radio frequency (RF) communication. This functionality is typically provided by a radio chip incorporated in the wireless sensor node design and allows the sensors in a network to communicate with one another or with a central data sink node.

Since typical wireless sensor applications are designed so that communication occurs exclusively within the network, relatively short range communication is desired, depending on the size of the structure to be monitored. Short-range communication improves network bandwidth and power resource management. The majority of wireless sensor platforms operate on the 900 MHz, 2.4 GHz or the 5 GHz frequencies, with the lower frequencies resulting in longer ranges. In the United States, these frequencies have been designated by the Federal Communications Commission (FCC) as unlicensed industrial, scientific, and medical (ISM) frequency bands. RF communication operating in the ISM frequencies is limited to a maximum 1W antenna power output, thus limiting the transmission range. Some examples of commonly used radio chips for wireless smart sensor networks (WSSN) are summarized in Table 2.1. The data rate dictates how quickly communication will occur while the maximum transmit (TX) power provides an indication of the communication power, and therefore transmission distance, of the radio. The power consumption (in mA) increases with increasing frequency band, transmission power, and reception (RX) sensitivity. The choice of radio utilized on a smart sensor must balance data rate and RX/TX power with power consumption.

Table 2.1 Wireless transceiver comparison

	Chipcon CC1000	Chipcon CC2420	Digi* 9XCite	Digi* Xstream
Frequency Band	315/433/ 868/915 MHz	2.4 GHz	900 MHz	900 MHz/ 2.4 GHz
Data Rate (kbps)	38.4	250	9.6/38.4	10-20
Power Supply (V)	2.1 – 3.6	2.1 – 3.6	5 – 12	7 – 18
Max. TX Power (dBm)	10 (433 MHz) 5 (868 MHz)	0	6	20 (900 MHz) 17 (2.4 GHz)
Max. TX Current (mA)	26.7 (433 MHz) 25.4 (868 MHz)	17.4	105	170 (900 MHz) 180 (2.4 GHz)
RX Sensitivity (dBm)	-110 (433 MHz) -107 (868 MHz)	-94	-106	-107 (900 MHz) -102 (2.4 GHz)
RX Current (mA)	9.3 (433 MHz) 11.8 (868 MHz)	19.7	65	70 (900 MHz) 90 (2.4 GHz)
Available Channels	9 (optimal)	16	7 (freq. hopping)	7 (freq. hopping)

*Note: Formerly MaxStream

Several Institute of Electrical and Electronics Engineers (IEEE) protocols used in RF communication are available, depending on the intended bandwidth and application. In the 2.4 GHz band alone, there is the 802.11b (Wi-Fi), the 802.15.1 (Bluetooth) and the 802.15.4 standards, among others. These protocols define the Medium Access Control (MAC) and Physical (PHY) communication layers in a wireless network. The MAC layer controls the radio hardware and dictates medium sharing, data packet addressing, and packet error detection. The PHY layer provides information on the received signal quality, activation for the radio transceiver, clear channel assessment, and channel selection. The 802.15.4 protocol defines both the MAC and PHY communication layers for low-power, low data rate applications (up to 250 kbps) such as wireless sensor networks, most commonly operating on the 2.4 GHz band. The ZigBee protocol, designed by the ZigBee Alliance, is based on the 802.15.4 standard, providing an overlying Network layer, in an effort to standardize and promote its use for wireless network applications. Figure 2.2 illustrates the ZigBee network protocol stack.

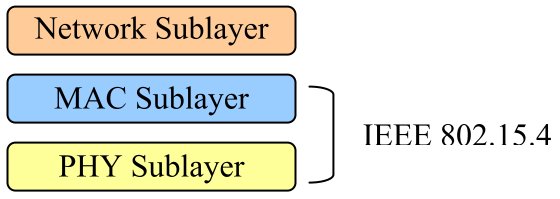


Figure 2.2 ZigBee network protocol stack.

Common practice in sensor network applications is to adopt the 802.15.4 PHY layer while using customized MAC/Network layers to meet application specific requirements. Sensor platforms employing 802.15.4 compliant radio hardware have the ability to communicate with a number of devices sharing the same PHY layer, while preserving flexibility in the communication control (Ali, et al. 2006).

For example, the ZigBee protocol incorporates random waiting times for packet transmission to manage transmission scheduling and improve transmission success rates in large sensor networks. One drawback to this approach is that it undermines very accurate time synchronization, which is critical for SHM applications. In this case, alternative protocols such as the Active Message Protocol (Buonadonna, et al. 2002) or the implementation of strict low-level packet transmission scheduling (Whelan and Janoyan. 2009) can be adopted to improve time synchronization.

RF transmission is inherently unreliable. The reliability of wireless communication is influenced by characteristics such as communication range, physical interference, multi-path effects, and noise, all of which contribute to data loss (Shankar 2002). Significant work has been done to understand these issues in wireless data transmission; for example Zhao and Govindan (2003), Seidel and Rappoport (1992), and Lee and Chanson (2002) discuss data propagation and loss in wireless systems. Pei et al. (2007) explore the influence of environmental factors on reliable real-time wireless communication and more specifically the distribution of lost data within a measurement record. However, there has been limited experimental characterization of the wireless communication hardware typically used by smart sensing platforms used in SHM applications. Care must be taken to determine acceptable levels of data loss for specific applications and employ communication hardware and protocols that ensure the required levels are met in an efficient manner. Many smart sensor platforms provide connectors for optional external antennas that can improve communication ranges and reliability.

Embedded computing

The microprocessor on the wireless smart sensor node provides its computational core that controls all of the functions of the sensor node allows for onboard data processing. The critical specifications of a microprocessor are its bus size, clock speed, power consumption and memory. The bus size defines the internal data bus of the microcontroller with typical values of 8-, 16-, or 32-bit. The clock speed defines the speed at which processing occurs and is directly related to the power consumption of the microcontroller. While larger data bus sizes and higher processor speeds are desirable, especially for high throughput applications such as SHM, microprocessor selection must balance performance with low power consumption.

The onboard memory is a critical feature of a smart sensor platform. The types of memory of interest are:

- Random access memory (RAM) – volatile storage for short term data and calculation results storage. Synchronous dynamic RAM (SDRAM) is a special type of RAM in which data is transferred at a set rate.
- Read only memory (ROM) – non-volatile storage for embedded software. Writing to ROM can be a power-hungry operation and should be limited when possible. Specific types of ROM include:
    - Electrically erasable programmable ROM (EEPROM) used for smaller amounts of information that is written byte-wise.
    - Flash – a type of ROM usually used for larger amounts of data that is written block-wise.

The microprocessor typically possesses some amount of data storage capacity however it is common practice to supplement this integrated memory with external memory. The ability to use smart sensors for data- and calculation-intensive applications relies heavily on the amount of RAM available for intermediate storage of measured and processed data.

Sensor interface

Smart sensor platforms do not possess inherent sensing capabilities. In general, sensors nodes can be separated into two categories: 1) those with onboard analog-to-digital converter (ADC) and 2) those without onboard ADC. If the sensor node has an ADC then it can be interfaced with a wide variety of sensors, as long as the output of the sensors is voltage or current within the specified bounds of the ADC. Platforms that incorporate ADCs allow flexibility in the sensor types that can be used and enable easier programming and development; however, they impose limitations on the quality of the data and the number of channels that can be measured. Because data quality and measurement resolution requirements can vary significantly depending on the particular application the sensed data is being used for, an onboard ADC may be a limitation. The effect of the ADC on data quality will be discussed in more detail in a subsequent section.

Sensor nodes that do not incorporate onboard ADCs require digital signals. Digital signals may be provided by using sensors with incorporated ADCs. The two most common digital interface options are for I/O are I2C (which allows interface to an unlimited number of channels) and SPI (serial data ports limited to one channel per port).

Power

Wireless sensors require a local power source to eliminate the need for cabling to provide power to the node. Regardless of the selected power solution, it can be assumed that the onboard power resources will be limited. These limited resources are a key factor in the design and selection of a smart sensor platform. While ultra-low power consumption is a desirable characteristic in a smart sensor, it cannot be at the cost of the computational capacity required for effective SHM.

The power requirements of a smart sensor network are highly dependent on the selected hardware, network topology, duty cycle, and data processing scheme. In each case, optimization tailored to the specific application may be achieved. In all cases, characterizing how different components and functions of the smart sensor affect the overall power consumption of the sensor node is important. The following functions/components must be considered:

1) Radio
2) Processor
3) Reading/writing memory
4) Sensing

Several options are available for powering smart sensors, and a careful assessment of the constraints and requirements of the application is necessary to determine the most appropriate way to power the sensor nodes. Many platforms are designed to be used with standard batteries. While batteries provide a simple solution, they are not an adequate solution for a long-term SHM installation. Much research has been focused on the topic of powering wireless sensor networks and many promising technologies are in various stages of maturity including solar, wind, mechanical vibration (Paradiso et al. 2005, Roundy et al. 2004) and remote power delivery (Mascarenas et al. 2008).

2.2.2 Platform comparison

Some of the first wireless smart sensor nodes proposed for SHM were developed by Straser and Kiremidjian (1998). Subsequently, a wealth of wireless or smart sensor platforms have been developed. Spencer et al. (2004) gives an overview of smart sensors and smart sensor technology development in which the merits of such technology towards improving SHM applications are introduced. In addition, the summary provided by Lynch and Loh (2006) cites over 150 papers on the topic of wireless sensors for SHM and specifically examines 24 wireless platforms that have been proposed for SHM. The platforms presented in these references are divided them into two broad categories: 1) academic prototypes and 2) commercially available platforms.

The academic prototypes utilize commercial-of-the-shelf (COTS) components to achieve their hardware requirements. The majority of these units utilize an onboard ADC with earlier prototypes incorporating an 8-bit microcontroller and later prototypes using up to a 16-bit microcontroller bus size. The processor speeds and memory capacities vary greatly as do the operating frequencies, data rates, and power consumption (Lynch and Loh 2006). In some cases, these prototypes have been demonstrated in full-scale monitoring applications. The primary limitation to the widespread use of these prototypes is their proprietary nature. The fact that they have limited numbers of users has impeded their use and potential technical contributions by a broader community.

The number of commercially available wireless sensor platforms is ever increasing. While most of these technologies are proprietary, meaning their hardware design and underlying operating software are not exposed to the user, they allow access by a broader community and are more likely to see large-scale industrial applications. The types of companies providing solutions for wireless sensor network applications can be divided into those providing two categories of products: 1) Integrated hardware (chips/modules), and some with integrated software, aimed at developers of embedded wireless sensor network applications and 2) Turnkey wireless sensors that incorporate hardware and software ready for installation and operation aimed at the sensor network end user.

One example of the first category of companies is Ember (2008). Founded in 2001, Ember specializes in ZigBee based wireless sensor and control network technologies. Their hardware products are so-called "systems-on-a-chip" (SoC) that incorporates a microprocessor, RAM, Flash, and ZigBee radio in a single IC package. The chip includes ADC capabilities as well as digital interfaces for sensor inputs. Ember also provides software solutions for embedded ZigBee networks. As with other companies that fall in the same category (Crossbow, Sentilla, Dust Networks, Jennic, etc.) the target market is system and application developers rather than those seeking fully developed and packaged solutions for SHM applications.

There are another group of companies that provide turnkey wireless sensing products with embedded software that is not exposed to the user except for certain parameter modifications. MicroStrain (2008) offers a wide variety of wireless sensors including strain sensors, accelerometers and generic analog input nodes. The SG-Link Wireless Strain Node has a nominal resolution of 1 μs, and the SG-Link Wireless Accelerometer Node employs a three-axis MEMS accelerometer with either a ±2g or ±10g operation. The measurement resolution for the ±2g measurement is 1.5 mg RMS with factory set anti-aliasing filter with a cutoff frequency of 250 Hz. The nodes utilize IEEE 802.15.4 2.4 GHz wireless communication with sampling rates of up to 2048 Hz and real-time data streaming up to 4 KHz for single channel operation. Synchronization between the nodes is estimated at 100 microseconds. The nodes can communicate up to 300 m with the use of high-gain antennas. The nodes are environmentally hardened and have mounting brackets for installation.

Other companies that provide fully-integrated wireless sensors include Millennial Net (2008), Sensicast (2008) and Bridge Diagnostics, Inc. (2008). In all of these cases the wireless sensors simply emulate a wired sensor network and send all of the sensed data in near real-time. In addition, the sensors lack flexibility and limit the ability of the user to implement any data processing at the node, and thus they are not suitable for distributed monitoring applications.

In contrast to the platforms presented in the previous paragraphs, the advancement of more widespread smart sensor applications for SHM in more recent years has come as the result of open source hardware and software platforms with somewhat generic sensor interfaces. In the late 1990s, DARPA (Defense Advanced Research Projects Agency) funded research to develop "smart dust" with the goal of achieving tiny, autonomous, low-power, low-cost systems for use in dense sensor arrays (Hollar 2000). This research led to the creation of the Berkeley family of Motes, which incorporate power, computation, sensors and communication on a single node. These platforms allow users to customize the sensors and the software to their application, thus making them attractive for a wide variety of uses. One of the first prototypes from this family of Motes to be commercialized was the Rene Mote (1999). Later improvements in memory capacity and processor speed led to the third generation of Mote: the Mica.

The original Mica Mote employs an 8-bit Atmel ATmega103L microcontroller with a 4 MHz CPU which possesses 128 kB of ROM and 4kB of RAM. An additional 512 kB of non-volatile (Flash) memory is included in hardware. The single-channel radio (TR1000) operates at 916.5 MHz with data rates up to 19.2 kbps. The 10-bit ADC integrated with the microcontroller provides the sensor interface and a number of sensor boards were made available for the platform (MTS101CA, MTS300A and MTS310CA, (Crossbow)). The primary drawback of the original Mica was the inferior performance of the wireless transceiver which was susceptible to noise and provided unreliable communication.

The Mica2 (Crossbow 2007a) was introduced in 2002 in an effort to improve the radio performance by switching to the ChipCon CC1000 wireless transceiver (see Table 2.1). The microcontroller was also upgraded to the ATmega128L. The most recent version of the Mica is the MicaZ mote. It has similar hardware to the Mica2, however it incorporates an 802.15.4 2.4 GHz radio (CC2420, see Table 2.1) and has a smaller physical size.

The Mica2 and MicaZ platforms have been used in many applications for a variety of purposes. Mainwaring, et al. (2002) reported the use of 32 motes for habitat monitoring in Maine. This application streamed live data via the web. In addition to the success of the monitoring program, the application provided a means to test data acquisition and network functionality beyond laboratory development. SHM applications employing Mica2 motes will be discussed in more detail later.

In an attempt to achieve a smart sensor module with lower power consumption, researchers at Berkeley introduced the Telos mote in 2004 (Polastre 2005). The MSP430 microcontroller was chosen for its low power requirements and quick transition times between operation modes. The Telos mote integrates sensing with the computational core and communication hardware with an onboard ADC in a similar manner as the Mica motes.

Table 2.2 Comparison of commercially available smart sensor platforms.

	Mica2 (Crossbow)	MicaZ (Crossbow)	Telos(B)/Tmote Sky (MoteIV*)	Imote2 (Crossbow)
Processor	ATmega128L	ATmega128L	TIMSP430	XScalePXA271
Bus Size (bits)	8	8	16	32
Processor Speed (MHz)	7.373	7.373	8	13 - 416
Program Flash (bytes)	128 K	128 K	48 K	32 M
EEPROM (bytes)	512 K	512 K	n/a	n/a
RAM (bytes)	4 K	4 K	1024 K	256 K SRAM 32 M SDRAM
Radio Chip	CC1000	CC2420	CC2420	CC2420
ADC resolution (bits)	10	10	12	n/a
ADC channels	8	8	8	n/a
Digital Interface	DIO, I2C, SPI	DIO, I2C, SPI	I2C, SPI, UART, USART	I2C, SPI, GPIO, UART, PWM, SDIO, USB
Active Power (mW)	24	24	10	44 @ 13 MHz 116 @ 104 MHz 570 @ 416 MHz
Sleep Power (μW)	75	75	8	100
Primary Battery	2 x AA	2 x AA	2 x AA	3 x AAA

* Now Sentilla

In 2003 Intel released a new smart sensor platform called the iMote as a result of research collaboration between Intel Research Berkeley Laboratory and UC Berkeley (Kling 2003). More recently, the second generation if Intel Mote, the Imote2, was introduced (Kling et al. 2005, Adler et al. 2005). The Imote2 is built around Intel's low-power X-scale processor (PXA27x). The fact that the processor speed can be scaled based on application needs allows for increased performance without a significant increase in overall power consumption. Another attractive feature of the Intel motes, and specifically the Imote2, is the fact that the platform does not provide an onboard ADC. Without a built-in ADC with set properties there is significantly more flexibility in the sensor design for applications using this platform. Sensor board customization for Mica and Imote2 platforms will be discussed the following section.

Table 2.2 summarizes the commercially available, open-source software wireless smart sensor platforms presented in this section. Of particular note is the comparison between the processor speed of the Imote2 to the processor speeds of the other platforms as well as the increase in RAM of the Imote2 over the Mica platforms. These unique features of the Imote2 make it the only commercially available open source WSS platform that is suited to the computational demands of high-sampling rate SHM applications.

2.2.3 Sensing using smart sensors

A variety of sensors can be used to evaluate the performance of a structure. The types of sensors used depend on the requirements of the particular application. The sensors most commonly and historically used in structural monitoring have been accelerometers to measure the global vibrational response and strain gages to measure local stresses.

Accelerometers are used in structural monitoring to evaluate the vibrational characteristics of a structure, upon which many SHM and damage detection methods are based. There are several sensing principals that can be utilized to measure acceleration including piezoelectric, capacitive, piezoresistive, and force-balance (servo).

Advances in IC technology and the fabrication of Micro-Electro-Mechanical Systems (MEMS) have led to the development of low-cost, high-sensitivity accelerometers that are suited to smart sensor applications. The primary market for MEMS accelerometers in the last 30 years has been in the automotive industry, first for crash testing and more recently for the deployment of airbags; the demand for such sensors for these and a growing number of other applications has led to improved performance and lower cost (Walter 2007).

Strain sensors provide a direct measurement of the strain an element is undergoing. The most commonly used strain sensor is the foil strain gage. While strain gages are inexpensive and very widely used types of sensors, they are susceptible to noise and require a lengthy mounting process. Fast mounting strain transducers, such as those offered by Bridge Diagnostics, Inc. (2008) greatly reduce the time of installation and provide performance comparable to foil gages. These features make them compatible with the quick and easy installation desired in smart sensor network applications.

Displacement and tilt measurements can provide a wealth of information on the static response of a loaded structure as well as its level of long-term deterioration. Displacement measurement techniques have long been difficult to implement. LVDTs provide a means for measuring relative displacement but do not allow for critical absolute displacement measurements. In recent years, laser-based systems have provided the means to measure displacement by using remote stations to capture deflections. While this technology is promising, it is not well suited to smart sensor network applications. One promising technology that could be incorporated in WSSN is Global Positioning System (GPS) technology. The biggest roadblock to GPS for SHM applications is the limited resolution of lower-cost systems; however, ongoing research in this area is likely to lead to its rapid advancement.

Other physical quantities of interest in SHM applications include temperature, relative humidity, light, and wind pressure/velocity. Temperature measurement is of critical importance in SHM systems. Sohn (2007) provides a summary of how significantly temperature and other environmental factors can change the response of a structure. In cases where the goal of the SHM system is damage detection, the effects of temperature on a structure's response can be greater than actual damage. In this case, not adequately compensating for temperature effects can undermine any damage detection techniques. Many sensors, especially strain sensors, are sensitive to temperature fluctuations. Temperature measurements can be used to calibrate sensor output for more accurate measurement results.

Limited but promising work has been conducted on combining different types of sensed data to gain a better understanding of structural response. When the goal of the SHM system is damage detection, multimetric sensing can combine information about both the local and global response of the structure. Analysis has shown that the sensitivity of some damage detection algorithms is improved by combining strain and acceleration measurements (Kijewski-Correa et al. 2006, Sim and Spencer 2007).

A sensor's performance can be measured in a number of ways. Most sensors produce an analog signal that must be digitized prior to being read by the microcontroller. The quality of the digital signal is a combination of the quality and scale of the analog sensor output and the properties of the analog-to-digital converter (ADC). The following list provides some of the measures of the performance of a sensor:

1) *Sensitivity*: The relationship between the physical phenomena that is sensed and the output of the sensor (usually voltage or current).
2) *Linearity*: A measure of how linear the output voltage is with the input physical measurand. For example, in a perfectly linear accelerometer, doubling the acceleration should result in a doubling of the output voltage.
3) *Resolution:* The smallest measurable increment that can be measured based on the number of discrete values that can be produced by the ADC. For example, an 8-bit ADC results in $2^8 = 256$ discrete values. If the voltage range of the input spans 2V, the resulting measurement resolution is 2V/256 = 7.8 mV.
4) *Signal-to-Noise Ratio (SNR)*: The ratio of the maximum measurable signal to the noise floor. The noise floor of a sensor is a combination of the inherent electrical noise in the analog signal and noise introduced by quantization in the A/D conversion.
5) *Bandwidth*: The frequency range over which measurements can be reliably observed.
6) *Drift*: The amount the mean value is expected to change over time (often in response to temperature fluctuation).
7) *Cross-talk*: A measure of the effect of coupling between sensor channels.

Careful consideration of the levels of excitation and response must be made in the selection of the appropriate sensor. Many SHM applications are limited to ambient excitation, which can result in very small acceleration and strain levels. This fact can lead to very low signal to noise ratios that may not be well tolerated by SHM algorithms. For example, the Golden Gate Bridge sees peak vibration levels due to ambient vibration on the order of 100s of μg (Abdel-Ghaffar et al.1985). On the other end of the spectrum, the sensors must possess adequate range to capture the response to extreme loading events without overloading the sensor and ADC.

In vibration-based SHM approaches, consideration of the expected range of response frequencies of the structure is critical. This range will dictate the required bandwidth and sampling rate of the sensors used. While many long-span bridges have natural frequencies below 5 Hz, higher modes may provide insight into local changes within the structure.

2.2.4 Sensor board customization

In an effort to tailor smart sensor platforms for SHM applications, some researchers have customized sensor boards to interface with commercially available platforms. Special attention has been given to improving vibration measurements through the use of higher-quality MEMS accelerometers and signal conditioning circuitry. Kurata, et al. (2006) examined the performance of several commercially available MEMS accelerometers in the context of SHM applications. Table 2.3 summarizes some of the sensors that were evaluated, as well as some more recently available MEMS accelerometers.

Table 2.3 MEMS accelerometers.

Manufacturer	Name	Axes	Range (g)	Sensitivity (mV/g)	Bandwidth (Hz)	Noise Floor	Supply Voltage (V)	Supply Current (mA)	App. Cost (USD)
Silicon Designs	SD1221	2	±2 g	1000	0 - 400	5 μg/√Hz	4.75 – 5.25	14	$250
ST Micro	LIS3L02AS4	3	±2 g	660	0 - 1500	30 μg/√Hz	2.4 – 3.6	0.85	$10
Analog Device	ADXL202	2	±2 g	167	0 - 1000	200 μg/√Hz	3.0	1.0	$15
Colybris	Siflex SF3000L	3	±3 g	1200	0 - 100	300 ng/√Hz	±6 - ±15	36	$1650

Ruiz-Sandoval et al. (2003) conducted studies on the performance of the Mica with the available MTS310CA sensor board which employed the ADXL202E accelerometer. Poor accelerometer performance was observed and it was found that the Mica's onboard, 10-bit ADC severely limited the sensor resolution. In addition, dynamic measurements required for SHM require the use of anti-aliasing (AA) filters prior to signal digitization. The lack of AA filter on either the sensor board or the main board also limited the performance of the module. Ruiz-Sandoval et al. (2003) designed a sensor board to interface with the MICA that incorporated a much higher-quality accelerometer (SD1221), but could not overcome the negative effects of the ADC. The Tadeo sensor board was subsequently adapted for the Mica2 platform with similar results.

Nagayama et al. (2004) developed a strain sensor board for the Mica2 for use in SHM applications. The strain board utilized a typical Wheatstone bridge configuration and 4.5-kΩ strain gages for lower power consumption and wide applicability. An operational amplifier and four-pole low-pass filter were incorporated to improve the signal quality and help minimize the effects of the low-resolution ADC. Test results showed excellence agreement with wired strain gages.

Pakzad and Fenves (2004) developed an accelerometer sensor board for use with the Mica2. Their motivation was SHM of civil infrastructure subjected to both ambient and large-scale seismic excitation; they also aimed to analyze the cost-performance tradeoffs of different quality accelerometers. Both the ADXL202 and the SD1221 accelerometers were used on their sensor board to achieve these goals (see Table 2.3 for specifications). The ADXL202 is chosen to measure larger acceleration levels with a range of ±2g and a resolution of 1mg over a 25-Hz bandwidth and the SD1221 measures ambient acceleration down to 10μg. Anti-aliasing of the accelerometer signals is provided by single-pole low-pass filters following the accelerometer with cut-off frequencies of 25 Hz, thereby limiting the use of this sensor for higher frequency applications. The signals are digitized by a 16-bit ADC (ADS8325) prior to interfacing with the Mica2. Acceleration is measured at 1 kHz and downsampled to 50 Hz by averaging to reduce noise in the signal. The RMS noise level observed over a 25 Hz bandwidth was for the SD1221 was 32 μg. In addition to acceleration measurements, a temperature sensor is also incorporated on the sensor board. Testing showed that the response of the SD1221 drifted over a 30-minute period in response to temperature changes at a rate of 0.18 to 0.78 mg/°C. Sampling rate jitter (variance in the time between samples) of less than 10 μs was observed as a result of the handling of simultaneous tasks by the microcontroller during sensing.

Researchers from Clarkson University have developed a complete wireless sensing module called the Wireless Sensing Solution (WSS). The WSS is developed around the Tmote Sky (MoteIV) smart sensor platform, a second generation of the Telos mote (see Table 2.2 for specifications) and design specifically for bridge monitoring using static and vibration-based approaches (Whelan, et al. 2007a and Whelan and Janoyan 2009).

The accelerometer chosen for the WSS is the three-axis ST Microelectronics LIS3L02AL (see Table 2.3 for specifications). Signal conditioning includes an analog 5-pole Butterworth anti-aliasing filter with a 100-Hz cutoff frequency. The signal is digitized using an 8-channel, 12-bit ADC (integrated in the TIMSP430 microcontroller). By oversampling at 512 Hz, digitally filtering the signal with a 55-tap digital low pass filter, and down-sampling to 128 Hz prior to data transmission, quantization noise can be reduced (effectively achieving an additional bit in resolution) and the poor roll-off characteristics of the anti-aliasing filter can be overcome. The sampling rate jitter typically associated with heavy loading of FIFO microprocessors that are also responsible for data sampling was addressed with low-level software to allow direct hardware interrupts at the completion of an ADC conversion. The accelerometer sensor design was validated using a Scanning Laser Vibrometer as a reference during a series of sinusoidal and sine-sweep small shake table tests.

The single strain channel provided by the WSS is centered on the ZMD31050 application-specific integrated circuit (ASIC) for strain signal conditioning and digitization. The chip includes gain amplification, temperature compensation, and a 15-bit ADC. The strain transducer chosen in the study is a Bridge Diagnostics, Inc. (2008) reusable transducer. The strain sensor was validated in the laboratory by comparing the output of the strain transducers with traditional wired gages interfaced with a National Instruments data acquisition system with good agreement of the measured time histories.

The WSS can be configured with a number of sampling rates; however, the effective measurement bandwidth is limited to 100 Hz due to the analog anti-aliasing filter. The challenge associated with complex analog anti-aliasing filters is that the components of the filter have inherent variability, which can in turn affect the phase and amplitude response of the filter and introduce error in the measured signal. For example, the resistor within the accelerometer used in this sensor (LIS3L02AL, see Table 2.3) has an error of ± 20% (STMicroelectronics, 2005). This resistor is incorporated into one of the poles of the ant-aliasing filter, potentially compromising the filter response; tests to observe the presence this behavior were not presented. Additionally, the effectiveness of the strict interrupt timing approach used to control jitter is not discussed. Section 2.5 discusses the use of the WSS in a full-scale bridge deployment.

2.2.5 TinyOS

TinyOS (www.tinyos.net) is the open-source operating system used on many smart sensors (Levis, et al. 2005). It utilizes a *component-based* architecture that makes it well suited to the extreme memory constraints of WSS. There is a large TinyOS user community and many successful implementations of sensor networks employing TinyOS.

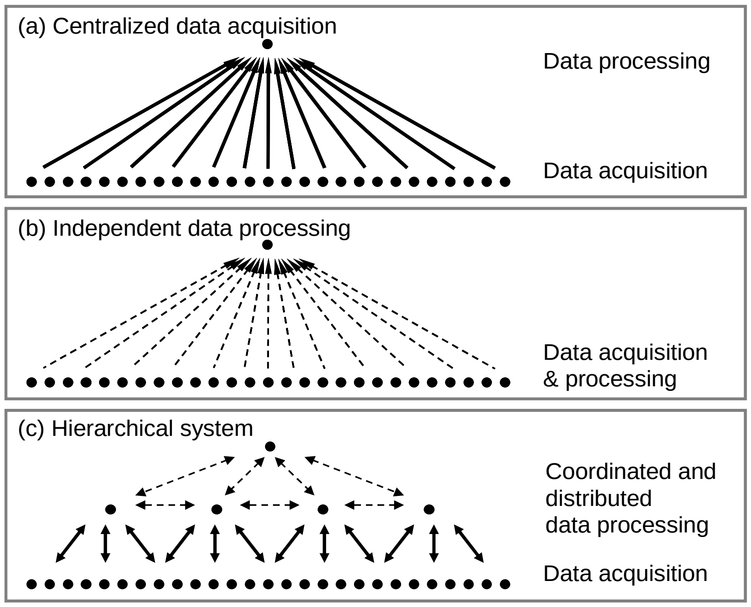
TinyOS employs non-blocking I/O which means that it has only a single memory stack. This system only supports two types of executions: tasks and hardware event handlers. Tasks are executed in a FIFO manner in the order they are posted and run to completion. This concurrency model makes programming in TinyOS complicated as it requires the use of many small event handlers and does not support real-time operations as a result of the concurrency model that is used. This concurrency model supports only two types of executions: tasks and hardware event handlers, making control of execution timing difficult. Tasks are executed and run to completion in an FIFO manner in the order they are posted. Hardware event handlers can preempt the execution of a task. Nagayama, et al. (2006) discuss the potential effects of this limitation as it pertains to achieving synchronized sensing in a smart sensor network and some methods for overcoming it. In particular, an uncertain delay in the start of sensing due to the lack of strict timing control is an issue that must be addressed if synchronized sensing is to be achieved.

TinyOS applications are written in nesC, a C-like language which supports the concurrency model used by TinyOS. Numerical sub-functions can be written entirely in C and included with the applications. Although TinyOS is widely used for WSSN applications, it is very challenging environment for non-programmers to develop network control and application software. This potential complexity of such software coupled with the uniqueness of the operating system/programming environment has severely limited the use of smart sensors for SHM applications. Chapter 3 presents a flexible software framework aimed at simplifying WSSN programming through the use of modular software services.

2.3 Data processing schemes

The ability to infer useful information on a structure's condition based on measured data is the purpose of SHM algorithms. The use of WSS for SHM applications requires that special attention be paid to the design and selection of an SHM algorithm. Many SHM strategies, such as those aimed at damage detection or high-fidelity modal analysis for example, require a dense array of sensors deployed through out the structure. To this end, the adopted SHM scheme must be scalable to a large number of sensors. Scalability refers to the ability to increase the size (number of nodes) of a network while maintaining proportionate increases in cost and preserving data communication and processing efficiency. To date, two primary approaches to data processing using wireless sensors have been employed to monitor civil structures, both of which impose limitations on network scalability and efficiency. Data processing, as it is used in the following paragraphs, refers to applying SHM algorithms such as modal analysis, system identification, damage detection, etc.

The first approach is to use a wireless smart sensor network as a direct substitute for a traditional monitoring system in a centralized data processing approach (see Figure 2.3). Achieving results similar to that of a wired system using wireless sensors introduces several potential problems that must be overcome, including accurate synchronization of the sensed data and effective power management. In terms of the data management of this approach, the wireless system emulates a wired system by sending all of the recorded data to a central processing station for post-processing. This approach can either be achieved in one of two ways. The first is by logging the data in the available memory on the sensor platform and then sending it wirelessly following measurement. The second is by streaming the data in a real-time manner. Once the data has been centrally deposited, traditional system identification and SHM algorithms can be applied, assuming adequate data quality that is sufficiently synchronized. The primary limitation to this approach is that as the network size increases, the amount of data to be wirelessly communicated becomes unmanageable. This communication may take many times the amount of time it took to sense the data and will rapidly deplete limited network resources. The approach does not take advantage of the local processing capabilities available on many smart sensor platforms - except perhaps to address synchronization, filtering and transmission of the sensed data - and is clearly a barrier to realizing a scalable wireless SHM system.



(a) Centralized data acquisition
Data processing
Data acquisition
(b) Independent data processing
Data acquisition & processing
(c) Hierarchical system
Coordinated and distributed data processing
Data acquisition

Figure 2.3 Smart sensor network topologies (Nagayama and Spencer 2007).

One method for that has been proposed for improving network bandwidth management is the introduction of some amount of higher power nodes with faster communication rates. Kottapalli et al. (2003) proposed a two-tiered wireless sensor network architecture with the goal of achieving the data rates required for SHM applications while maintaining reasonable power consumption levels. Chintalapudi, et al. (2006) utilizes a tiered approach, with lower tier nodes and powerful upper tier nodes. Assuming that the upper tier nodes have sufficient power, the limitation on the communication speed among upper tier nodes is removed; power consumption at lower tier nodes is moderate. While heterogeneous networks such as these may be a reasonable approach to power management especially on structures with readily available power sources, they introduce complexity that can inhibit network flexibility and reconfigurability. The introduction of more powerful, upper-tier nodes may not be practical and can reduce the advantages of smart sensors. Regardless, sensor networks employing a centralized data processing scheme limit the scalability of SHM smart sensor networks.

A second, albeit less common, approach to data processing for sensor networks independent data processing (see Figure 2.3), in which the majority of the data processing occurs at the sensor node without any data sharing, prior to sending critical results to the central base station (Sohn et al. 2002, Lynch et al. 2005). Single-node autoregressive, wavelet transform and FFT methods are potentially suited to such an approach. While such a scheme certainly exploits the computational ability of the smart sensor and drastically reduces the communication burden, it eliminates the ability to determine spatial information such as structural mode shapes. The lack of spatial information is expected to make damage detection less effective.

2.3.1 Decentralized SHM

Gao and Spencer (2008) proposed a hierarchical approach to data processing to resolve the issues associated with the above approaches. In the Distributed Computing Strategy (DCS) local sensor communities are formed in which data is shared and processed. This hierarchical approach takes advantage of the computational capacity of the smart sensors by pushing the computational burden out into the network while without sacrificing spatial information. This distributed approach significantly reduces the amount of required RF communication and is well suited for use with a scalable smart sensor network.

The DCS was developed for acceleration measurements and uses the Natural Excitation Technique (NExT, James et al. 1993) in conjunction with ERA (Juang and Pappa 1985) to perform system identification within the local sensor communities. The damage detection algorithm adopted for DCS is the flexibility-based Damage Locating Vector (DLV) method (Bernal 2002). In this method the results of the modal analysis provided by system identification within each sensor community are used to construct flexibility matrices of the undamaged and damaged structure, $\mathbf{F}_u$ and $\mathbf{F}_d$, respectively. The damage locating vectors (DLVs), **L**, are vectors such that when they are applied as static loads on the structure, they induce zero stress in the undamaged members. Equation 1.1 shows that the displacement vector is the same for the undamaged and damaged cases when the DLVs are applied which implies that the DLVs are in the null-space of the difference in the flexibility matrices.

$$\mathbf{F}_u\mathbf{L} = \mathbf{F}_d\mathbf{L} \text{ or } (\mathbf{F}_u - \mathbf{F}_d)\mathbf{L} = \mathbf{0} \qquad \textbf{(2.1)}$$

The DLVs are determined from the difference in the experimentally obtained flexibility matrices using singular value decomposition (SVD) and are in turn applied to an analytical model of the structure to locate the members with low induced stress (i.e. the candidate damaged members). Gao (2005) experimentally validated the DCS approach on a three-dimensional truss model (Figure 2.4) using wired sensors.

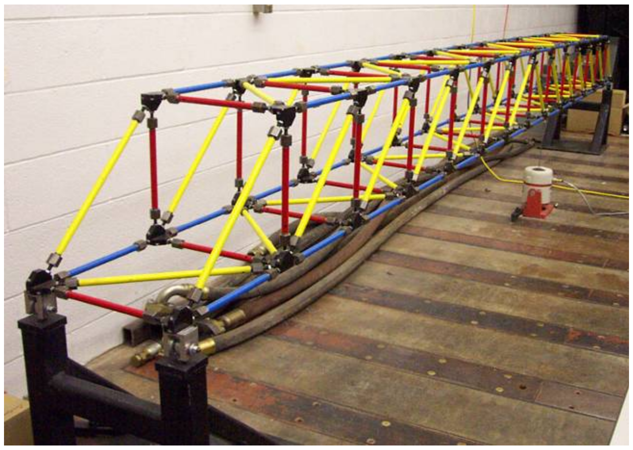


Figure 2.4 Three-dimensional truss model.

Nagayama and Spencer (2007) extended DCS for use with on a network of Imote2 smart sensors. The Imote2 platform was chosen for this application since it is the only commercially available smart sensor platform with the computational and data storage capacity to carry out this distributed SHM system. A homogeneous network architecture is chosen to provide system flexibility; however, the nodes are functionally differentiated as shown in Figure 2.5 and summarized in Table 2.4. This type of topology is tolerant of node failure since any node in the vicinity of the failed node can take over its responsibilities. In addition, node responsibilities may be shifted periodically to improve overall power consumption.

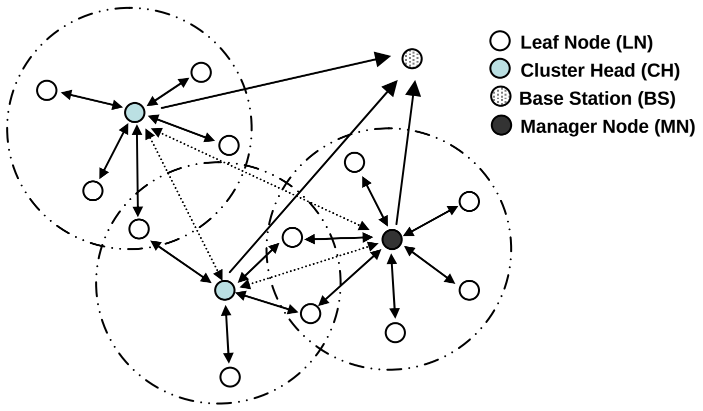
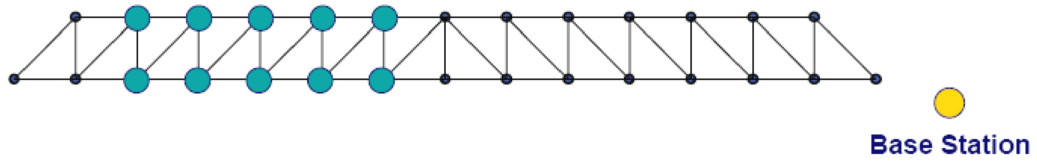


Figure 2.5 Network topology.

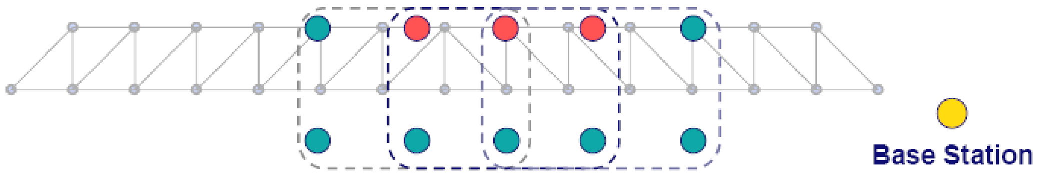
Table 2.4 Sensor network topology.

Sensor Node Role	Provided Functionality
Leaf Node	Sensing, some data processing, some communication
Cluster Head	Sensing, more data processing, more communication
Manager Node	Management of network-wide operations such as sensing and time synchronization
Base Station	Gateway to PC for command dissemination and data recording

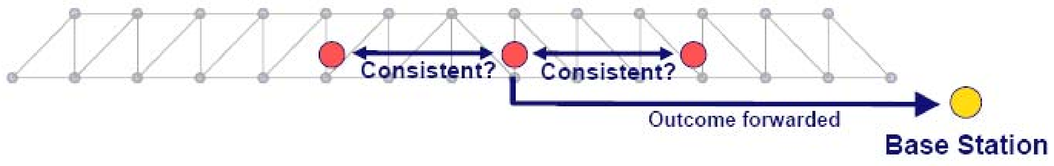
Several leaf nodes make up a local sensor community. One of the leaf nodes in the sensor community is designated as the clusterhead and one of the clusterheads in the network is designated as the manager node. Sensor communities overlap one another so that all members in a structure are shared by more than one community, as shown in Figure 2.6 (b).



(a)



(b)



(c)

Figure 2.6. DCS Implementation: (a) Leaf nodes on a truss structure, (b) overlapping sensor communities with designated cluster heads and (c) DCS logic between clusterheads.

Damage is represented in the truss by replacing one of the truss elements with one having a reduced cross-sectional area. The truss is excited vertically with a band-limited white noise and acceleration is measured in three overlapping sensor communities. The Stochastic DLV method (SDLV), an output-only version of the DLV method, is used as the damage detection algorithm and the damaged element, which is shared by two sensor communities, is successfully identified. Only when consensus is reached amongst the clusterheads regarding the damage detection is the outcome sent to the base station. This process is referred to as DCS Logic (Figure 2.6 (c)). Nagayama and Spencer (2007) provides a detailed discussion on the occurrence of false negative and false positive damage detections and other aspects of the experimental validation

2.4 Networking and implementation

Before any SHM algorithm can be implemented in a smart sensor network, critical network functionality, such as time synchronization, data aggregation, and reliable communication, must be addressed. Middleware services to bridge the gap between the hardware components and the application software are introduced to address these functionality concerns. This section discusses the necessary middleware services for SHM applications.

2.4.1 Time synchronization

Time synchronization errors in a smart sensor network can lead to inaccurate modal analysis and damage detection primarily due to incorrect phase response characterization leading to poor mode shape estimates and must therefore be addressed for SHM applications (Nagayama et al. 2007). Time synchronization is a commonly provided middleware service for wireless sensor networks and thus it has received widespread investigation. Each smart sensor has its own local clock, which is not synchronized initially with the clocks of the other sensor nodes and can drift over time. By communicating with the surrounding nodes, smart sensors can assess relative differences among their local clocks and implement the appropriate adjustments. Some of the more common synchronization methods include Reference Broadcast Synchronization (RBS, Elson et al. 2002), Flooding Time Synchronization Protocol (FTSP, Maroti et al. 2004) and Timing-sync Protocol for Sensor Networks (TPSN, Ganeriwal et al. 2003). Lynch et al. (2005) implemented the RBS and reported a maximum time delay of 0.1 s. TSPN was implemented on Mica2 motes (Ganeriwal et al. 2003) with a reported synchronization accuracy of 50 μs. Mechitov et al. (2004) implemented a modified FTSP on a Mica2 motes system for SHM to achieve synchronization better than 1ms. The same method was later implemented on the Imote2, incorporating drift estimation and correction, (Nagayama and Spencer, 2007) with the resulting time synchronization error reduced to about 10 μs.

While synchronizing the nodes in a network to a high degree of accuracy is critical, it does not, in itself, guarantee that the data acquired on a network is synchronized across the network. The following section addresses the issue of achieving synchronized sensing in addition to the network synchronization discussed in this section.

2.4.2 Synchronized sensing

Even when high levels of network time synchronization accuracy are achieved, a lack of synchronization between measured signals may remain. Assuming network time synchronization is addressed, the non-synchronicity of sensed data is the result of three primary remaining issues:

1) Inconsistent sample rates amongst the sensor nodes
2) Sample rate fluctuation on each node (jitter)
3) A random delay between the command to start sensing and the actual start of sensing

While real-time operating systems implemented for industrial applications with ample hardware resources can manage execution timing with more precision, achieving such operation in small embedded systems results in significant system size and complexity. Real-time control of wireless sensors in a network is particularly challenging; the situation is exacerbated for the high sampling rates required in SHM applications.

Even when a sensor node itself has near real-time control, i.e. the ability to precisely manage execution timing, the peripheral devices such as sensor components may have execution time delay or uncertainty in timing as has been observed in less-expensive COTS sensor components (Nagayama et al. 2006, Rice and Spencer 2007). Sample rate fluctuation, or jitter, can result from a number of sources. Lower quality digital sensors may also contribute to higher jitter levels. Kim et al. (2007) attributes jitter in high sampling rate applications (1 kHz) using wireless sensors to random delays in timer events caused by rapid posting of sensing and data logging tasks. While tasks in TinyOS are generally serviced in a FIFO manner, hardware even handlers (such as writing data to Flash) preempt the execution of task, thereby introducing random jitter. This problem is limited to platforms with onboard ADCs because of the tasks controlling the ADC are implemented in TinyOS and must wait in the queue with all other tasks being carried out by the microprocessor. External ADCs operate independently of the microprocessor and do not introduce the same jitter. Kim, et al. (2007) proposes reducing jitter by implementing higher-frequency flash writing and turning off all other components that could trigger preemptive tasks. The resulting sample rate fluctuation was reduced to less than 10 μs which corresponds to 0.09 degrees at 25Hz.

Nagayama and Spencer (2007) realized synchronized sensing on the Imote2 running TinyOS using post-processing. A polyphase implementation of resampling using the global time stamps addresses achieves synchronized sensing with accuracy better than 25 μs.

2.4.3 Communication protocols

Data loss is intrinsic to wireless communication due to a wide range of factors, including excessive communication range, physical interference, multi-path effects, and noise. Data loss can significantly degrade otherwise good quality sensor data and result in false damage detection and inaccurate estimates of modal parameters (Nagayama et al. 2007, Pei et al. 2007). Data loss can also occur during the transmission of commands which will result in compromised network functionality and potentially cause the network to fail. For these reasons, communication protocols must be implemented which achieve acceptable levels of data loss and reliable command dissemination.

Nagayama and Spencer (2007) developed acknowledgement-based reliable communication protocols for four distinct communication cases:

1) Unicast (node-to-node) of long data records

2) Broadcast (single node to multiple nodes) of long data records
3) Unicast of commands (single packet)
4) Broadcast of commands (single packet)

In the case of long data records, the entire record is sent before missing packets are addressed. This approach improves communication efficiency by reducing the number of queries and acknowledgement that must be sent. The short message (command) communication protocol immediately addresses dropped packets and resends until the data is received. While this reliable communication protocol is designed to achieve 100 percent packet reception rate, it has the potential to cause the network to become hung up in a never-ending cycle of packet retransmission. This behavior, however, was not observed by Nagayama and Spencer (2007).

Limitations on the transmission ranges of wireless transceivers require that multi-hop communication protocols be implemented to truly achieve a scalable sensor network for large civil infrastructure. *MintRoute* (Woo et al. 2003) is a TinyOS component that provides multi-hop routing for wireless networks and has been implemented successfully in a WSN deployed on the Golden Gate Bridge (Kim 2007).

2.4.4 Data aggregation

One of the most effective tools for power management in a smart sensor network is data aggregation because it can manage the amount of data communicated in the network, thereby conserving network resources. Data aggregation takes advantage of the local processing capabilities of the smart sensors to take data from a number of sensors and process it in such a way as to provide an aggregate value or set of values that eliminate the need to send raw data throughout the entire network. Data aggregation can be as simple as averaging measurement values between adjacent sensors.

Zimmerman et al. (2008) implement automated modal parameter in a wireless sensor network. The output-only Frequency Domain Decomposition (FDD) technique is used to determine the mode shapes of a historic theater balcony. In this method, each sensor node in the network calculates an FFT of its acceleration data and uses the Peak Picking (PP) method to estimate natural frequencies. These frequencies are sent back to a central processing station. With the estimated frequencies from all of the nodes in the network, global natural frequencies can be estimated with a high degree of accuracy. These global natural frequencies are then sent back out to the sensor nodes. Each of the sensor nodes determines the complex value of its calculated FFT at the frequencies dictated by the central node. Finally, the nodes in the network share these FFT values with an adjacent node to create a series of two-point mode shapes. These mode-shapes are sent back to the central node where they can be pieced together into global mode shapes. The results of this approach represent a high degree of savings in the communication load, as summarized by the authors, and appears to be a promising approach for decentralized system identification.

Nagayama and Spencer (2005) implement a model-based data aggregation approach to implement system identification and damage detection within local sensor communities. The Natural Excitation Technique (NExT) is a widely used modal analysis technique to obtain modal information from output-only measurement of structural vibration. This technique utilizes the correlation functions of structural response as the input for system identification algorithms such as ERA or Stochastic Subspace Identification (SSI). Correlation functions require response data from two sensor nodes. By carefully selecting the scheme by which data records and resulting correlation functions are communicated, the amount of raw data and calculation results that must be communicated can be kept to a minimum. Within a sensor community one node acts as the reference for correlation function estimation. Rather than all of the other nodes sending their data records to the reference node, the reference node broadcasts its data record to the community. After local calculations, the nodes in the community then send back much shorter correlation function estimates, thereby reducing the total communication burden. The authors provide an estimate of the communication savings resulting from this distributed approach.

Sim, et al. (2009) describes a decentralized method for determining global structural mode shapes in a two-part process. The first part of the process is to apply the NExT and ERA to data acquired and shared within overlapping subsets of sensor within the sensor network. Once the modal parameters from each subset of sensors is determined, they are communicated back to a central processing location. The global mode shapes are then estimated from the modal parameters calculated for each sensor subset using a least-squares method. The size of the of the sensor subsets determines the level of data aggregation and resulting communication reduction that occurs with the use of this method.

2.4.5 Application software enhancement

The application and middleware services software developed for the Imote2 to implement DCS on the Imote2 platform proved to be successful in experimental validations. The code, however took a significant amount of time and effort to develop and debug. The resulting application is comprised of 207 files totaling 1.8 MB. The core application logic software contains almost 9,000 lines of code while the numerical sub-functions contain a combined total of almost 54,000 lines of code. This complex code is specific only to the experimental validation carried out on the laboratory truss model and is not easily modified for other structures and applications, even for those with a background in embedded systems. The challenges associated with building application software for smart sensor networks have limited the advancement of the technology for SHM applications.

Given that many SHM applications utilizing smart sensors will use many of same components, a more modular approach to application software is beneficial. The middleware services can be adopted by a wide variety of applications as can many of the numerical sub-functions. Efforts to achieve an open-source, modular software architecture would encourage the development of more applications utilizing smart sensors for SHM applications. However, the task of creating such a software framework is not as simple as decomposing the code into smaller portions. Critical component interaction, efficient data transfer, built-in flexibility, and fault-tolerance measures must be addressed. Chapter 3 provides details on a software framework that addresses these concerns thus enabling a wide range of SHM application software development for WSSNs.

2.5 Full-scale implementations of wireless smart sensor networks for SHM

This section reviews some of the full-scale implementations of WSS networks for structural monitoring. In each case, the purpose of the monitoring system, the selected hardware, communication protocols, and data processing schemes are discussed. Table 2.5 at the end of this section summarizes this information for each case.

Some of the first efforts to deploy smart sensors in a full-scale monitoring system were done on the Alamosa Canyon Bridge, first by Straser and Kiremidjian (1998) and subsequently by Lynch et al. (2003). Both cases were short-term implementations aimed at validating their wireless sensor modules.

MicroStrain demonstrated the capabilities of their wireless strain and temperature modules with the installation of a network of sensors on the Ben Franklin Bridge in Philadelphia, PA (Arms et al. 2004). This continuous and autonomous real-time monitoring system included 10 sensor nodes and utilized a base station with a remote cellular phone interface allowing remote network interrogation and control. Each sensor node used rechargeable Li-Ion batteries and is enclosed in an environmentally sealed enclosure with magnetic mounts for quick installation on steel structural members and lead wires to strain sensors installed using a portable spot-welder. The system continuously measured strain at 1 Hz with the nodes in a sleep state between samples to conserve power. Train passage events triggered the system to increase the sampling rate to 32 Hz for the duration of the event. The radio channel and power could be reprogrammed remotely to optimize communication within a given environment. All of the programming of the network was achieved through a graphical user interface. The implementation on the Ben Franklin Bridge lasted several months and was removed once the cyclic strain performance of the bridge was found to be within normal ranges. No local processing was utilized in this application.

On two separate implementations, the Geumdang Bridge in South Korea was instrumented with 14 wireless sensor modules to measure the bridge's response to truck loading and to validate the performance of the wireless system in comparison to a wired system (Lynch et al. 2005 and Lynch et al. 2006). The wireless sensor used in the study is a prototype developed by the authors (Wang et al. 2004). The computational core of the sensor node includes the ATmega128 8-bit microcontroller with built-in 128 kB of flash and 4kB of RAM plus an additional 128 kB of external RAM for data storage. A 16-bit, 4-channel ADC is employed. The accelerometer noise floors during the initial Guemdang Bridge deployment ranged from 0.5 to 2.5mg. In an effort to improve the measurement resolution, additional signal conditioning for the sensor output was provided during the second deployment, consisting of an amplifier followed by a four-pole Bessel band-pass filter circuit with cutoff frequencies at 0.014 and 25 Hz, respectively. The noise floor in the second deployment improved such that it exhibited similar output as the wired system used simultaneously although the authors did not report a specific value. A beacon signal is used to implement time synchronization of the network with synchronization errors of up to 0.1 second reported.

The wireless tests were performed with forced excitation provided by truck loading. The in-network data processing approach employed in this application is an independent scheme. FFT and peak picking calculations are performed on each individual sensor node, and the results are sent back to the central control station along with time-history. The resulting information extracted from the network at the data repository are the modal frequencies and estimated operational deflection shapes, which are expected to provide some representation of the actual mode shapes. The results compare well with those obtained from post-processing of data obtained by the wired monitoring system.

Berkeley researchers installed 64 MicaZ motes with their customized accelerometer board (described in section 2.2) on the Golden Gate Bridge (Kim et al. 2007, Pakzad 2008 and Pakzad et al. 2008). Various combinations of available sensor channels (2 to measure larger accelerations and 2 to measure smaller accelerations and one temperature) and the 64 nodes were used in different data collection runs. The data channels were sampled at 1 kHz then averaged and downsampled prior to transmission. Each sensor node could accommodate 500 kB of data (~ 250,000 data points) which, for 60 active nodes, results in up to 30 MB of data to be transmitted in the network. In Pakzad, et al. (2008) it is reported that 20 MB of data from a full network of sensors took 9 hours to transmit to the base station. They implement a reliable communication protocol (Straw) in conjunction with a pipelining data transmission technique to achieve reliable, multi-hop communication. Pipelining allows nodes within the network to transmit data simultaneously, thereby more effectively utilizing network bandwidth. The data is not processed in network but rather uses a centralized strategy for data processing.

Whelan et al. (2007) implemented a 20 node WSS (introduced in section 2.2) network on a single-span bridge in St. Lawrence County, NY to validate the use of a high-sampling rate, real-time dense sensor array for SHM. A total of 40-sensor channels were employed: 29 acceleration and 11 strain, to perform ambient vibration and quasi-static measurements. The sensor data was oversampled digitally filtered and downsampled using a 55-tap digital filter to 128 Hz and the data was streamed to the central collection station in near real-time. An acknowledgement-based communication protocol ensured that over 99-percent of the data was successfully communicated with an effective data rate of 97–126 kbps. The peak excitation levels produced by the traffic loading ranged from 2 to 10 mg and were clearly captured by the accelerometer channels. The data was post-processed to determine the modal properties of the structure, which corresponded well to a FEM model. While not utilized in this study, the authors propose that the strain measurements can be used for load rating tests. Beyond digital filtering and downsampling, no in-network processing occurs in this application. The near real-time transmission of data that in this implementation emulates a wired network is not scalable to higher-frequency sampling rates and or/a larger number of sensors. While numerical estimates are presented to predict the battery life of this network under various duty cycles, no field validation of the power consumption is discussed.

The full-scale applications presented in this section illustrate the rapid advances seen in the implementation of WSS for civil infrastructure monitoring. However, most of these applications emulate a wired network and don't exploit the computational capacity of the nodes to realize scalable SHM. Even when some local processing is employed as in the system used on the Alamosa Canyon and Guemdang Bridges, the information provided by the network is limited to basic modal analysis. To date, there has not been a successful full-scale WSSN implemented with the intention of deploying distributed data processing on a large array of sensor to characterize structural performance.

2.6 Summary

The background provided in this chapter shows the steady progression of SHM research towards integrated, global approaches to monitoring and damage detection. Many traditional monitoring systems have been used in recent years with a wide range of goals. The introduction of wireless smart sensors has opened the door to achieving distributed SHM in dense sensor networks. The components smart sensor platforms and their implications on the performance of the WSSN have been discussed. The available smart sensor platforms and the applications that employ them show the wide variety of technology and approaches that have been used for SHM applications. These applications have either emulated a wired network by using central data collection and post-processing or they have done limited and independent processing on the sensor nodes without any data sharing. In the few cases that in-network data processing is explored, the implementations were not evaluated for their long-term viability. Additionally, prior to the Golden Gate Bridge deployment, the largest network deployed was limited to 20 sensor nodes.

The advancement of WSSNs has been limited by several factors, including the lack of flexible sensor hardware that works in combination with a smart sensor platform to provide high-fidelity data acquisition at high data rates and the processing capabilities required to implement decentralized SHM strategies. In addition, the software running on the smart sensor platforms presented in this chapter is very complex and not easily modified by other users. Finally, there have been very few WSSN deployments in which the network operates autonomously, e.g. the short term Ben Franklin Bridge deployment (Arms et al. 2004), and there have been no deployments in which an autonomous network has operated on a large scale with the ability to implement distributed SHM data processing.

The research presented in this report seeks to fill the gaps identified herein by providing sensor hardware designed for SHM applications utilizing a smart sensor platform suited to high-frequency, high-fidelity data acquisition and demanding data processing algorithms. Additionally, a flexible and modular software framework will be created which provides the foundation for a wide range of SHM applications, including the support of network functionality, data processing, power management, and autonomous monitoring. Most importantly, the hardware and software will be open-source and made available to the broader research community in an effort to truly advance research in the field of smart sensors for SHM, ultimately improving infrastructure maintenance and enhancing public safety.

Table 2.5 Summary of full-scale implementations of wireless sensor networks for bridge monitoring.

Implementation	Purpose	Sensor platform	# of Nodes	Sensor types (channels)	Sampling rate	Test duration	In-network processing	Results
Alamosa Canyon (Lynch et al. 2003)	Proof-of-concept for wireless system	WiMMs (prototype)	7	Accel. (14)	976 Hz	Short-term	Independent FFT	Modal frequencies
Ben Franklin (Arms et al. 2004)	Demonstration of wireless system with remote reprogramming capability	SG-Link	10	Strain + Temp. (10)	1 – 32 Hz	Continuous, medium-term	None	Streaming real-time strain time histories
Guemdang (Lynch et al. 2005 and 2006)	Validation of wireless sensing unit prototype, embedded computing and time synchronization	Author's prototype	14	Accel. (14)	14 sensors @ 50 Hz 4 sensors @ 200 Hz	Two short-term tests (2004 & 2005)	Independent FFT and peak picking	Modal frequencies, operational deflection shapes
Wright Road (Whelan et al. 2007)	Full-scale validation of wireless sensing unit for real-time monitoring	Tmote Sky	20	Accel. (29) Strain (11) Temp. (20)	128 Hz	Short-term	None	Modal analysis after central data collection
Golden Gate (Pakzad et al. 2008)	Test wireless sensor network requiring multi-hop communication	MicaZ	64	Accel. (128 - 256) Temp. (64)	50 – 200 Hz	Short-term (2006)	None	Modal analysis after central data collection

Chapter 3

FLEXIBLE SOFTWARE DEVELOPMENT FRAMEWORK

3.1 Software development for wireless smart sensors

SHM applications implemented on wireless smart sensor networks (WSSN) require complex programming, ranging from network functionality to algorithm implementation. Software development is made even more difficult by the fact that many smart sensor platforms employ special-purpose operating systems without support for common programming environments. The extensive expertise required to develop SHM applications has severely limited the use of smart sensing technology for monitoring of civil infrastructure.

A common approach to the issue of software complexity is to divide the software system into smaller, more manageable components. Service-oriented architecture (SOA) has recently been proposed as a way to use this design philosophy in building dynamic, heterogeneous distributed applications (Singh and Huhns 2005, Tsai 2005).

SOA design principles are focused on how services are defined and the manner in which data is passed from service to service. Services, in SOA terminology, are self-describing software components in an open or modifiable distributed system. The description of a service, called a contract, lists its inputs and outputs, explains the provided functionality, and describes non-functional aspects of execution (timeliness, resource consumption, cost, etc.). Data is passed among the services in a common format. An application built using SOA consists of a composition of a number of linked services within a middleware runtime system that provides communication and coordination among them. Services do not need to know who provides the required input data or from where it comes. Different applications can be built from the same set of services, depending on how they are linked and on the execution context (Gu et al. 2005). This approach provides support for dynamic, highly adaptive applications without the need to revisit and adapt the implementation of each service in a particular application context.

Smart sensor networks consist of numerous independent nodes, each an embedded computing platform with a processor, memory, and a radio transmitter. As such, WSSN applications are by definition distributed and thus require communication and coordination for parts of the application running on different nodes. SOA has been proposed to address the inherent problems in designing complex and dynamic WSSN applications (Liu and Zhao 2005, Mechitov et al. 2007). Building an application from a set of well-defined services moves much of the complexity associated with embedded distributed computing to the underlying middleware. This approach also fosters reuse and adaptability, as services for a given application can be employed by many other applications.

Another attractive aspect of SOA is that it provides a separation of concerns in application development. Application designers can focus on the high-level logic of their application, service programmers can concentrate on the implementation of the services in their application domain, and systems programmers can provide middleware services (reliable communication, time synchronization, data aggregation, etc.) that enable the services to interact. In sensor networks, which at this stage are principally used by scientists and engineers, the application designer is likely to be the user of the application as well, having expertise in SHM applications and the desired output of the network, but limited knowledge on network programming and the hardware-software interface. This situation makes it especially important for the less complex high-level design of the application and the domain-specific algorithms used by the services to be separated from the often more complex low-level infrastructure necessary to make the system work. SOA in WSSNs makes it possible to compose and deploy, on-the-fly, complex applications through a web-based user interface suitable for non-programmers. User-driven WSSN programming holds the promise to lower the barriers to entry in sensor network application development and to accelerate their use in structural health monitoring applications.

The Illinois Structural Health Monitoring Project (ISHMP), a collaborative effort between researchers in civil engineering and computer science at the University of Illinois at Urbana-Champaign, has sought to tackle the complexity associated with creating WSSN applications by developing a framework for structural health monitoring using the design principles of SOA described above. This framework provides a suite of services implementing key middleware infrastructure necessary to provide high-quality sensor data and to transport it reliably across the sensor network, as well as a broad array of numerical algorithms. By leveraging this framework, engineers may focus their attention on the advancement of SHM approaches and the development SHM systems without having to concern themselves with low-level networking, communication and numerical sub-routines. This software is open-source and available for public use at http://shm.cs.uiuc.edu/software.html.

The primary contributions of the research in this report toward the framework development include providing the application specific requirements of the framework, the development of key services, the creation of test applications for the numerical services, extensive testing and refinement of additional services to improve the contents of the framework, and extensive documentation for using its components. The effort to create the software framework was not simply a matter of decomposing existing code into smaller components, it also involved the challenge of managing the interaction between the components, including efficient data transfer and memory utilization, as well as extensive measures to ensure its robust, fault-tolerant operation. While the development of the framework represents a collaborative effort, its entire contents are presented in this chapter for completeness.[1]

3.2 Application software enhancement

The service-based software framework (http://shm.cs.uiuc.edu/software.html) presented in this section provides an open-source software library of customizable services for, and examples of, SHM applications utilizing WSSNs. SHM middleware services and distributed damage detection algorithms reported in Nagayama et al. (2007) and Nagayama and Spencer (2007), along with a rich array of tools, utilities, and algorithms, have been implemented to enable efficient development of robust, extensible, and

[1] Unless otherwise noted, the software presented in this chapter was developed in collaboration with Kirill Mechitov from the Department of Computer Science at the University of Illinois at Urbana-Champaign.

flexible structural health monitoring applications on WSSNs. Additional services that enable autonomous network operation have also been developed, as described in Chapter 4.

The components of the service-based framework can be divided into three primary categories: (1) foundation services, (2) application services, and (3) tools and utilities. In addition, a library of supporting numerical functions that are common to many SHM algorithms is provided including fast Fourier transform (FFT), singular value decomposition, Eigenvalue analysis, etc.

3.2.1 Foundation services

In SOA terminology, services are high level, self-describing building blocks for distributed computing applications. The foundation services implement functionality needed to support the application and other services. In the context of this research, they include gathering synchronized sensor data, reliably communicating both commands and long data records, and providing accurate and precise timestamps to collected data. When used together, one of the primary purposes of these services is to be used by applications to achieve synchronized sensing from a network of sensors. The following paragraphs provide more detail on each of the foundation services.

- A *Time Synchronization* service (Mechitov et al. 2004) provides consistent, network-wide global timestamps for sensor data, making it possible to meaningfully compare data collected from multiple sensors.
- The *Unified Sensing* service (Rice et al. 2008) provides a convenient, general-purpose application programming interface, replacing the standard TinyOS sensing interface for the Imote2 and extending its functionality to include precise timestamping of the data and providing transparent support for a variety of sensor boards. The existing TinyOS sensing interface does not support the collection of data from remote nodes and is difficult to modify for a variety of sensing parameters. In the *Unified Sensing* service, data for all sensor channels, together with a single set of associated timestamps, is returned to the application in a single, shared data structure. A compact data representation format is used, which encapsulates all information necessary to recreate the sensor values, yet is memory-efficient for storage and transportation across the wireless network. This complete and self-contained data representation makes it easy to pass around and modify the data without hard-coding connections between components that use only parts of this data. This approach facilitates data being passed directly to the application services described below.
- Since sensor data loss is intrinsic to wireless systems and undermines the ability to perform system identification and detect damage (Nagayama et al. 2007), a *Reliable Communication* service (Nagayama and Spencer 2007), which eliminates data loss, is needed for sending commands and data between sensor nodes. The *ReliableComm* service employs four distinct reliable communication protocols, chosen automatically based on the type of communication, to eliminate data loss in an efficient manner.

3.2.2 Application services

These services provide the numerical algorithms necessary to implement SHM applications on the Imote2s and may also be used independently. For each application service, an application module to test the algorithm on both the PC and the Imote2 has been developed by the author. The application services are as follows:

- *SyncSensing*: Resamples timestamped sensor data from a node in a synchronized sensor network (provided by the Unified Sensing application service) so that the output for each node in the network has a common sampling rate with a common start time. The service takes raw sensor data and a sparse set of associated global timestamps as arguments and applies the resampling filter. This resampling is accomplished in a memory-efficient way with a by applying the filter to the data one block of at a time so that additional memory requirements for the service are independent of the size of the input data.
- *CFE*: Returns the Correlation Function Estimate (CFE) via FFT calculation. CFE takes two synchronized discrete-time signal vectors as input and outputs their CFE employing a user-specified number of FFT points and spectral window.
- *ERA*: Performs the Eigensystem Realization Algorithm (ERA). This time-domain system identification service uses the impulse-response function, or in the case of the NExT algorithm (James et al. 1993), the correlation functions, to determine the modal characteristics of the structure (damped natural frequencies, damping ratios, mode shapes, modal participation factors, EMAC values and the state-space matrices defining the identified model of the structure).
- *SSI* (Sim 2009): Performs the covariance-driven Stochastic Subspace Identification (SSI) algorithm. This time-domain system identification method uses the cross correlation functions to determine the modal characteristics of the structure (damped natural frequencies, damping ratios, mode shapes, and the matrices which define the state-space model of the structure).
- *SDLV*: Performs output-only, model-based damage detection using the Stochastic Damage Locating Vector (SDLV) method. The inputs of SDLV are the modal characteristics determined by one of the system identification service. More detailed information on this method can be found in Nagayama and Spencer (2007).
- *FDD* (Sim et al. 2009): Performs the Frequency Domain Decomposition (FDD) algorithm (Brinker, et al. 2001). This frequency-domain system identification method uses the cross spectra to determine the modal characteristics of the structure (damped natural frequencies and mode shapes). Because the natural frequencies are selected by a peak-picking method, some modes may not be reliably found.

Documentation has been provided for each service and test application within the directory that the software is located, giving more detail on requirements and formats of the inputs and outputs for the service. The documentation and test applications created for the numerical services as a part of this research are essential for allowing a broad audience to use the software framework.

3.2.3 Tools and utilities

This section describes application tools and utilities for basic testing and debugging. These tools and utilities are necessary in any large scale or long-term WSSN deployments to evaluate network the conditions at the structure, determine appropriate values of adjustable system parameters, and assess power consumption and longevity issues. Included are utilities for resetting nodes remotely, listing the nodes within communication range of the local node, and changing the radio channel and power for local and remote nodes.

The application tools can be categorized as either those operating on a single node or those operate on multiple nodes distributed in the network. The single node application tools include:

- *LocalSensing*: This tool allows sensor data to be collected while a single Imote2 is connected directly to the PC (i.e., no radio communication is required). It allows developers to test the functionality of sensor boards and develop driver software for new boards.
- *imote2comm*: A basic terminal program for interfacing with the Imote2 through the Imote2 Interface Board's USB port. It uses the serial port UART interface to open a telnet-like connection with the mote.
- *TestServices* (Sim 2009): A numerical service that combines application services: CFE, ERA, and SDLV. It uses acceleration signals as input in the CFE service to calculate the correlation functions that is used in the ERA service. The estimated modal characteristics of the structure are then used in the SDLV service to identify potential damage locations.

The application tools that involve multiple nodes are given below. The distributed nature of these tools require careful scheduling and coordination of network tasks and are therefore more susceptible to the failure if any of the nodes in the network malfunctions. For this reason, significant effort has been made to ensure that the applications continue to operate even when one or more of the nodes in the network exhibit unexpected behavior.

- *TestRadio* (Linderman et al. 2009): Tests the raw bidirectional communication between a sender node and a group of receiving nodes, and outputs the packet loss rate (in each direction, and round-trip).
- *RemoteSensing:* A network-wide distributed application, this tool is used to collect sensor data from multiple sensors and provides the basis for most distributed SHM applications. *RemoteSensing* provides a high level of flexibility in the choice of network and sensing parameters. The first step in the application is network synchronization followed by sensing with concurrent collection of timestamps. Depending on the command that is given at run time, this service can output either the raw timestamped data or resampled synchronized data. If the resampling option is selected, the data is resampled locally using the *SyncSensing* service to account for any jitter or non-uniform delay in the start of sensing for each node. All data and commands in *RemoteSensing* are sent between nodes using the *ReliableComm* service, eliminating data loss.
- *DecentralizedDataAggregation* (Sim 2009): This sample application illustrates use of the framework for data acquisition and processing in a decentralized,

hierarchical sensor network. This application supports multiple sensor clusters, in which data processing is conducted independently to other clusters. The main outputs of the application are sensor data and their correlation functions in each sensor cluster.

The *RemoteSensing* and *DecentralizedDataAggregation* application tools employ a distributed state machine to determine the timing and control flow of the application across a network of sensors. A state machine is a formal method for defining how an application behaves or responds when it is in a particular *state* and the *transitions* required to move between states. The flowchart given in Figure 3.7 illustrates the state machine for the *RemoteSensing* application. Table 3.2.6 summarizes each state and transition associated with *RemoteSensing*.

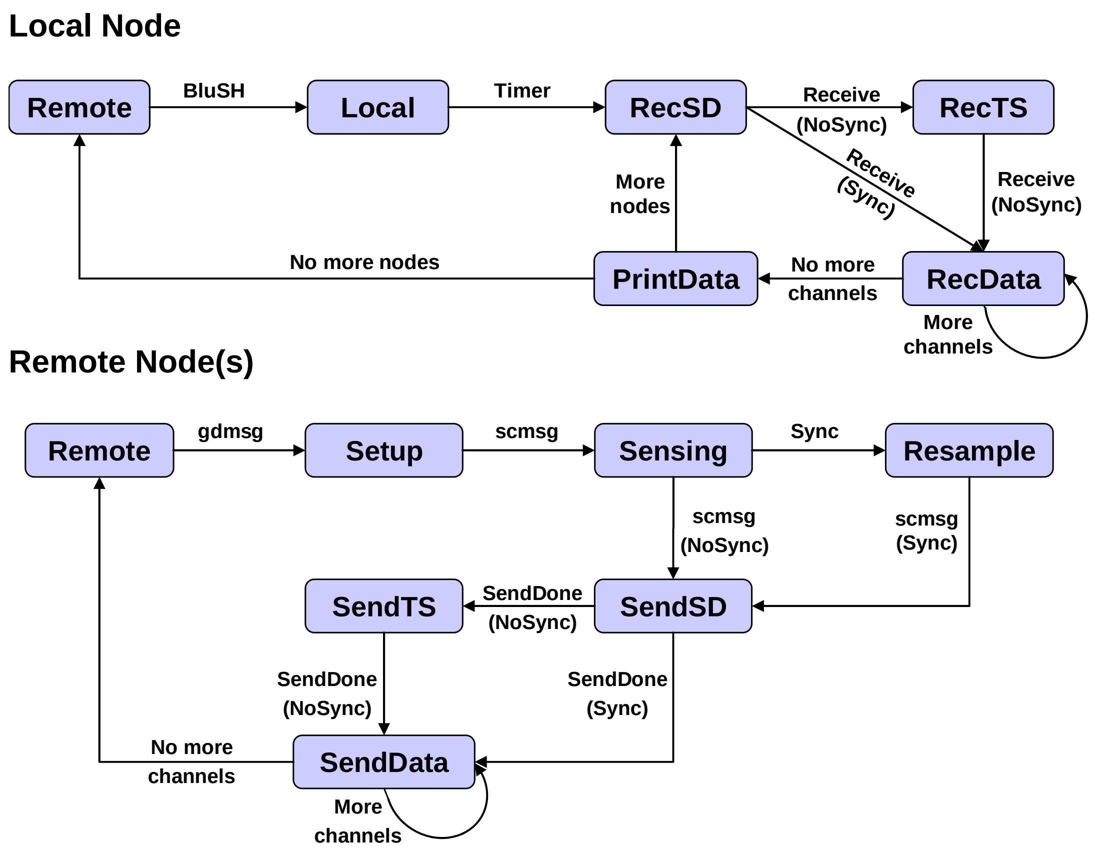


Figure 3.7 *RemoteSensing* state machine for the local node (top) and remote nodes (bottom). Boxes represent states, arrows represent transitions, and arrow labels indicate conditions or actions needed for the transition to occur.

Table 3.2.6 State and transitions for *RemoteSensing* application.

State	*Description*
Remote	Initial state
Local	Initial local node state
Setup	Receive and store sensing parameters
Sensing	Data acquisition
Resample	Resample of acquired data based on timestamps and initial delay
SendSD	Send sensor data structure
RecSD	Receive sensor data structure
SendTS	Send timestamps (if data is not resampled)
RecTS	Receive timestamps (if data is not resampled)
SendData	Send sensor data
RecData	Receive sensor data
PrintData	Write data to PC
Transition	
BluSH	Application initialized by user through the Blue Shell interface
gdmsg	*GetData* message containing sensing parameters received
Timer	Timer set to wait for remote node(s) to acquire data
scmsg	*StartCollection* or request for data message received
Sync	Resampling flag set
NoSync	Resampling flag not set
sendDone	Previous message sent successfully
receive	Data successfully received

3.2.4 Extensibility

The modularity and flexibility of the components of the service-oriented architecture described above lend themselves to the exploration of new approaches to solve specific problems. As a simple example, Figure 3.8 illustrates how the system identification method can be swapped out in an SHM application. In keeping with the SOA framework, these interchangeable services share the same input and output parameters. Other application examples that can benefit from the modular services provided in the framework include distributed damage detection algorithms that rely only on the parameters derived from the correlation function estimates (Castaneda et al. 2008) or methods for distributed modal parameter estimation in a WSSN (Sim et al. 2009).

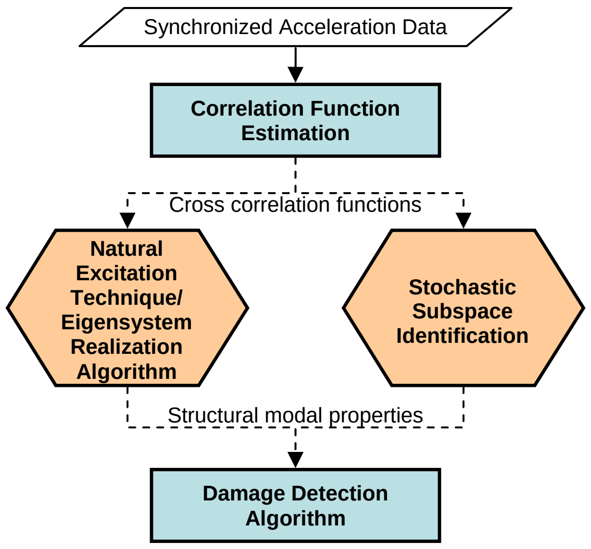

Synchronized Acceleration Data
Correlation Function Estimation
Cross correlation functions
Natural Excitation Technique/ Eigensystem Realization Algorithm
Stochastic Subspace Identification
Structural modal properties
Damage Detection Algorithm

Figure 3.8 Alternate services for SHM application development.

In its current form, the software development framework presented in this chapter requires application programmers to provide the code necessary to interconnect the services and tools in a way that makes sense for their applications. In general terms, this code serves the following functions:

- Run service *X* at node **A**
- Send a control message to node **B** to run service *Y*
- Send results from **B** to **A**
- Run service *Z* taking as inputs the outputs of services *X* and *Y*

3.3 Summary

This chapter has described an open-source framework developed using the design principles of service-oriented architecture (SOA). The creation of the framework facilitates efficient memory utilization and data transfer between the various components and ensures robust, fault-tolerant application performance. This SOA-based approach creates an enabling framework that manages the complexity inherent in the use of WSSNs. As a result, researchers and application engineers can design and deploy efficient SHM systems without worrying about how the underlying middleware and application services are implemented. The service-based framework described herein will ensure that smart sensor technology sees more widespread use in SHM applications, ultimately driving the technology forward to improve infrastructure maintenance and enhance public safety.

In addition to the services presented in this chapter, services have been added that enable autonomous network operation with limited user interaction. These services will be introduced in Chapter 4, and their validation presented in Chapter 7.

Chapter 4

AUTONOMOUS MONITORING

Three critical deployment issues drive the smart sensor software that is presented in this chapter: 1) Continuous and autonomous monitoring, 2) efficient power management, and 3) data inundation mitigation. While these may appear to be conflicting goals, careful application design can meet the requirements for all three. The solution is a network that is only minimally active during non-critical structural response, but becomes fully active to measure higher response levels. The software presented in Chapter 3 lays the groundwork for full-scale, autonomous monitoring of civil infrastructure however it does not address these concerns that arise when moving from a laboratory setting to a full-scale deployment.

Ideally, full-scale smart sensor network deployments should require minimal external interaction. After some initialization involving the establishment of network operation parameters, the network should run autonomously unless instructed otherwise by the network administrator. Special care must be taken in the design of the application software to ensure a continuous and autonomous operating scenario is achieved while maintaining power efficiency. These measures can be divided into three categories: schedule-based operations, trigger-based operations, and safe-guard features. This chapter presents software developments in each of these categories that, when integrated, enable full-scale, autonomous network operation.

4.1 Sleep cycling

In a traditional wired sensor implementation, power management is of little concern. The sensors can remain active at all times and thus have the ability to be interrogated at any time to acquire data. Unlike such wired systems, one of the most critical features of a successful WSSN deployment is the implementation of careful power management strategies. The Imote2 allows the processor to be put into a deep sleep mode, whereby only the clock component of the processor is supplied power; all other components are powered down. The deep sleep mode lasts for a set period of time (thus the need for the clock to be powered) and results in significantly reduced power consumption. When the node is in the deep sleep state it cannot send data or receive via the radio or the serial ports and the LEDs do not function. Effectively, the node has no power until the sleep time expires.

While it may seem advantageous to keep the nodes in the deep sleep mode for extended periods of time to save power, this approach limits the ability of the base station node to access the network at random to send inquiries or initiate network operations. To take advantage of the power savings of the deep sleep mode, while still allowing the base station node access to the remote nodes, a sleep/wake cycle service called *SnoozeAlarm* has been developed. When *SnoozeAlarm* is operating on the remote nodes (i.e. they are in the *SnoozeAlarm* mode), they sleep for a period of time, SLEEP_TIME and then wake up for a short period of time, WAKE_TIME, during which they can listen and receive message. This process is illustrated in Figure 4.9. The ratio between WAKE_TIME and the sum of WAKE_TIME and SLEEP_TIME is the *SnoozeAlarm* duty cycle. The duty cycle should be minimized while still allowing the listen time to be long enough to receive and process commands (>500 ms). The impact of the duty cycle on power management is further discussed in Chapter 6.

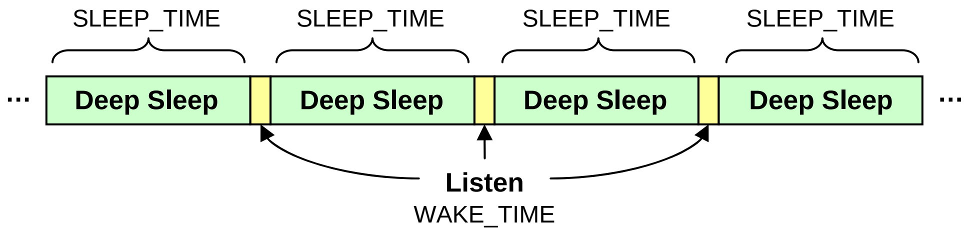


Figure 4.9 *SnoozeAlarm* timing.

SnoozeAlarm provides three interfaces to the application:

- `SnoozeAlarm.wakeup(targets, tcount)`: Command given to local node to wake up targets (tcount is the number of targets).
- `SnoozeAlarm.awake(targets, tcount)`: Event signaled on local node with the node IDs and number successfully woken up nodes.
- `SnoozeAlarm.stayawake()`: Command given on remote node to stay awake (i.e. stop wakeTimer).

If a message is received during the wake period, the remote node stays awake until it is placed back in the sleep/wake cycle. *SnoozeAlarm* leverages the fact that *ReliableComm* continuously resends packets to the destination node until it receives an acknowledgement packet, thus allowing the base station node to send a wake up (or other command) to the remote node, even if it is in the deep sleep mode. *ReliableComm* will continue to send the message until the remote node wakes up or the message is withdrawn. Upon reception of the command, the remote node stops its wake timer, sends an acknowledgement packet, and awaits the next command. The node resumes the *SnoozeAlarm* cycle upon being put back to sleep or being reset (either by software or a hard reset). Figure 4.10 illustrates how *SnoozeAlarm* operates on the remote nodes.

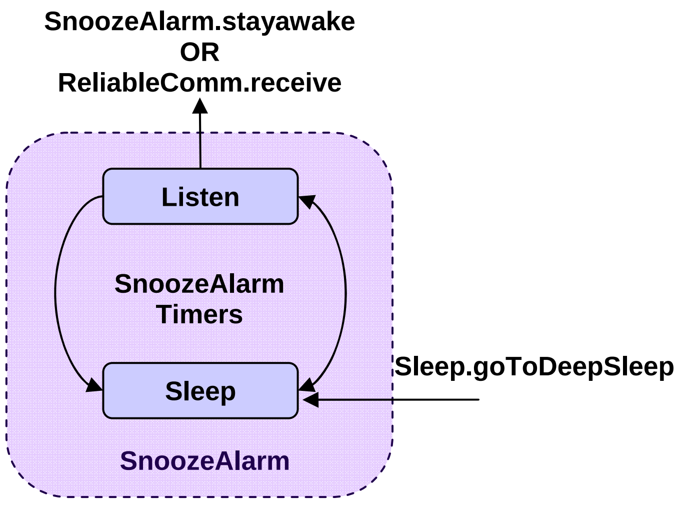


Figure 4.10 Remote node *SnoozeAlarm* interfaces and operation.

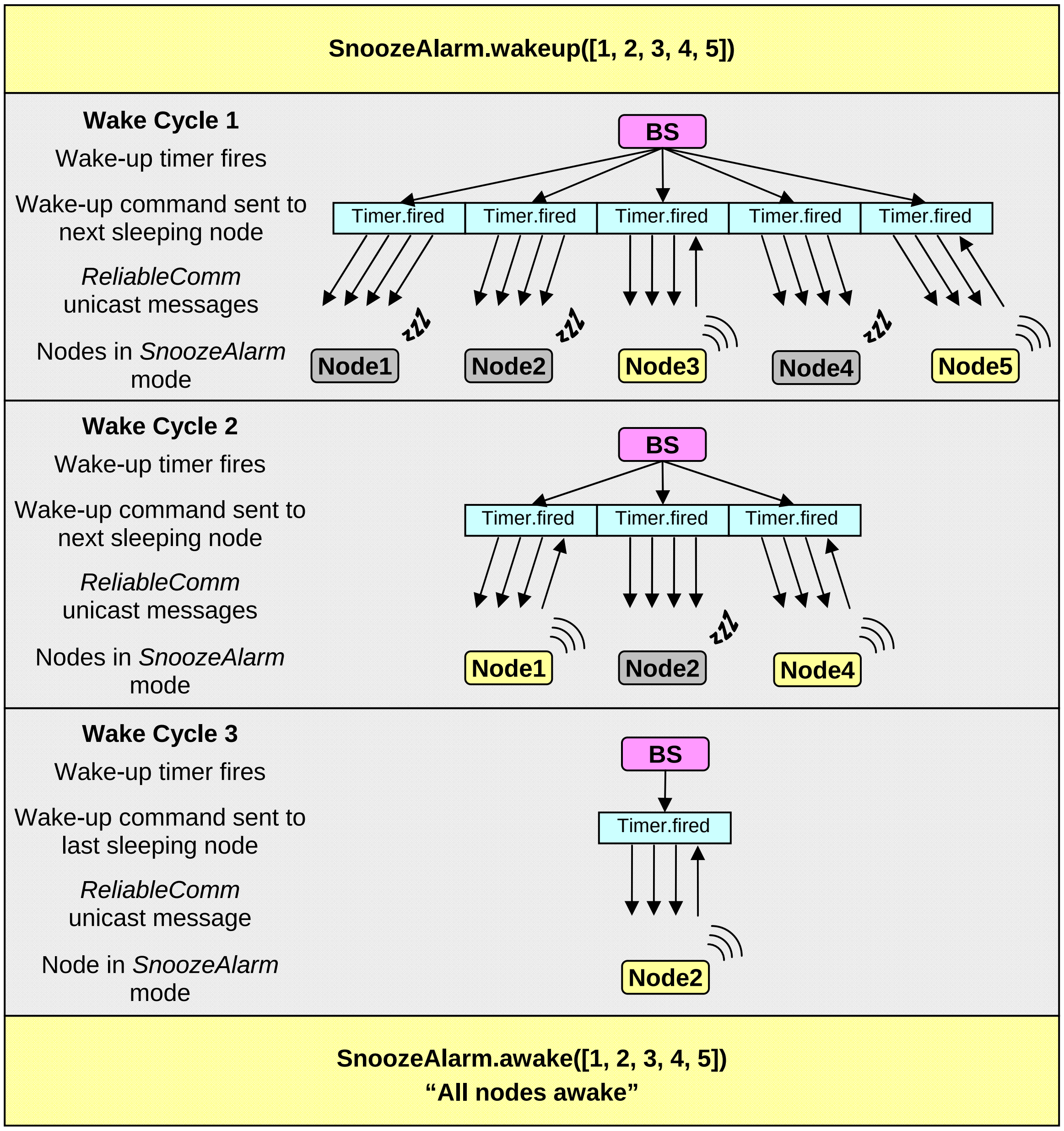

SnoozeAlarm.wakeup([1, 2, 3, 4, 5])
Wake Cycle 1
Wake-up timer fires
Wake-up command sent to next sleeping node
ReliableComm unicast messages
Nodes in SnoozeAlarm mode
BS
Timer.fired
Node1
Node2
Node3
Node4
Node5
zzz
Wake Cycle 2
Wake-up timer fires
Wake-up command sent to next sleeping node
ReliableComm unicast messages
Nodes in SnoozeAlarm mode
Wake Cycle 3
Wake-up timer fires
Wake-up command sent to last sleeping node
ReliableComm unicast message
Node in SnoozeAlarm mode
SnoozeAlarm.awake([1, 2, 3, 4, 5])
"All nodes awake"

Figure 4.11 *SnoozeAlarm*.wakeup command

The `SnoozeAlarm.wakeup` command provides an efficient method for waking a network of nodes in *SnoozeAlarm* mode. The *ReliableComm* protocol for broadcasting messages to a group of nodes is only successful if all of the destination nodes respond with an acknowledgement in a set period of time, thus limiting its use for waking the network. Instead the network is woken in a serial manner using successive unicast commands from the base station node to individual nodes in the network. The base station node cycles through the list of sleeping nodes, sending a wakeup command to one node in the list each time the wake-up timer fires. When a node sends back an acknowledgement, thus indicating it received the message and is remaining awake, it is removed from the list of nodes to wake up and added to the list of nodes that have been successfully woken up. This process continues until all nodes are awake or until a time-out timer expires. In both cases, a list of the nodes successfully woken up is signaled in the `SnoozeAlarm.awake` event. The wake up process is illustrated in Figure 4.11.

4.2 Threshold triggering

The *ThresholdSentry* tool allows a subset of the network to act as "sentry" nodes that are woken up periodically to sense data for a short period of time, determine if a set threshold has been exceeded in the measured data, and send a flag back to the base station. The base station node signals a `ThresholdSentry.exceeded` event upon reception of a flag indicating the threshold has been exceeded. This event can be used by the higher level application to make decisions on whether to wake the network and initiate network sensing or distributed modal analysis, etc. The current implementation of *ThresholdSentry* utilizes acceleration measurements; however triggers, such as strain levels or wind speed, could be incorporated into the application.

ThresholdSentry is setup on the base station node by specifying the nodes that comprise the sentry network, the interval at which they will be asked to sense data, the duration of the data check on each sentry node, the sampling parameters for the data check, and the threshold value used for comparison in the data check. Once *ThresholdSentry* is initiated, a timer is started that fires after the check interval is reached. When this timer fires, the base station node sends a wakeup command and sentry request to the first node in the list of sentry nodes. Upon reception of the sentry request, the remote node senses for the prescribed period of time. When the data collection is complete, the remote node checks the absolute maximum normalized value for each channel that collected data. The maximum peak of all the channels is then compared to the threshold value. If the threshold is exceeded, the remote node sends a flag, with the peak measured value, back to the base station and remains awake to wait for the next command. If the base station receives the flag indicating the threshold has been exceeded, it signals the `ThresholdSentry.exceeded` event. If the threshold is not exceeded on the sentry node, it sends a message back that indicates the threshold was not exceeded and puts itself back into *SnoozeAlarm* mode. If the base station node receives a message that says that the threshold was not exceeded, the base station node restarts the check timer and moves on to the next sentry node in the queue. *ThresholdSentry* operation on the local and remote nodes is illustrated in Figure 4.12 and the states and transitions are described in Table 4.7.

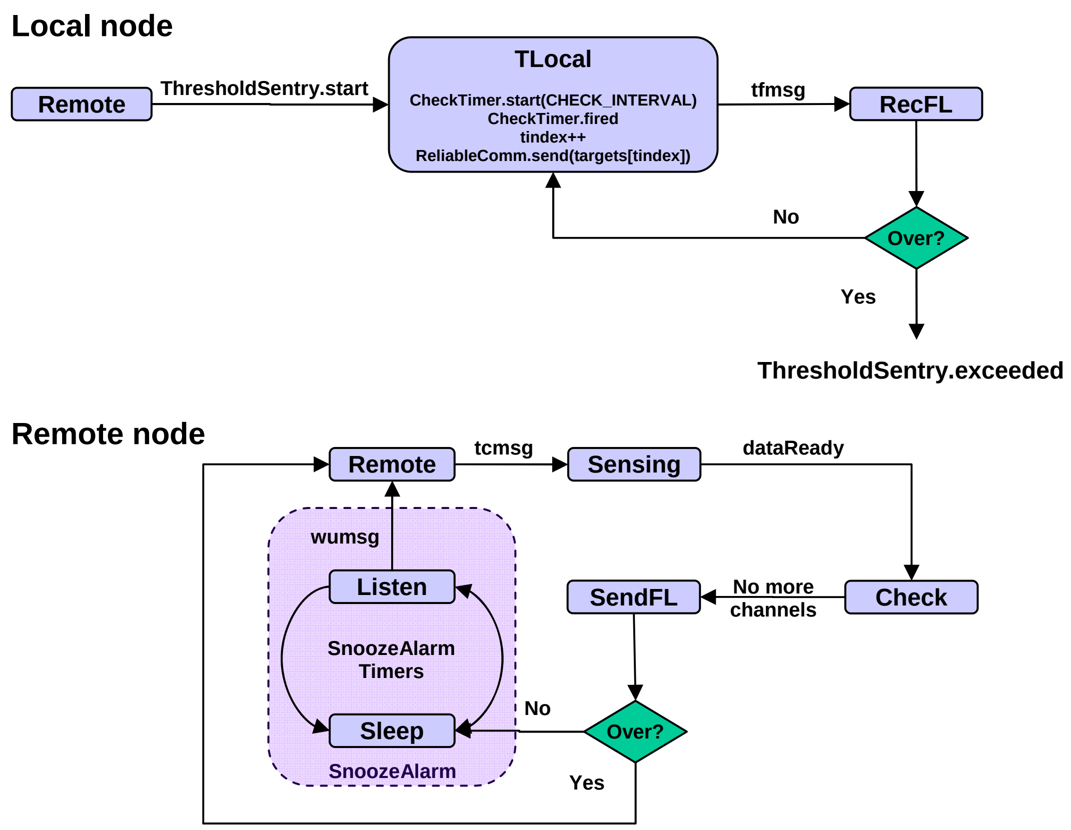


Figure 4.12 *ThresholdSentry* operation on local node and remote nodes with *SnoozeAlarm*.

Table 4.7 State and transitions for *ThresholdSentry* tool.

State	*Description*
Remote	Initial state
TLocal	Initial local node state
Sensing	Data acquisition
Check	Calculation of maximum value of normalized sensed data
SendFL	Send threshold flag
RecFL	Receive threshold flag
Transition	
ThresholdSentry.start	Application initialized by user
tcmsg	Threshold check message sent to next sentry node in queue
dataReady	Sensing on sentry node is complete
no more Channels	Peak for all channels calculated and checked against threshold
tfmsg	Threshold flag message sent from sentry node to remote node

The selection of the threshold value and the sentry nodes within the network should be made such that the threshold is exceeded often enough for adequate structural monitoring, but not an excessive number of times at the risk of data inundation and higher power consumption levels. Because a single threshold value is used for all sentry nodes, the nodes selected as sentry nodes should measure similar levels of vibration to ensure consistency in the events that trigger the network. For example, on a long-span bridge, the nodes located near the support piers are expected to experience much lower vibration levels than those near the mid-span of the bridge. Sentry nodes in each of these locations and would exceed the threshold under very different loading circumstances. Because of additional sensing duties, sentry nodes will consume more power than non-sentry nodes. However, if more nodes make up the sentry network, the burden of increased power consumption on each sentry node will decrease because the base station will call on each sentry node fewer times within a day. If only a few sentry nodes are selected, a larger power source may be required for those nodes. Also, the check time interval must be carefully selected so that important events are captured, while power management is still considered. The effect of the selection of these parameters on power consumption is discussed more in Chapter 6.

Several safeguards have been built into *ThresholdSentry* to ensure its continuous operation in spite of potentially unexpected network behavior. When a sentry request is sent to a sentry node, a timer is started on the local node. If the sentry node does not respond before the time expires, the sentry node prints a message that the node was not responsive and moves onto the next sentry node. This measure ensures that if a sentry node dies or becomes unavailable for some reason, the application will continue. Carefully monitoring the debug output is important to diagnose problems within the network. A node that is consistently skipped indicates that it requires attention.

On the remote sentry nodes, timers are also implemented to reset the node if it does not carry out its duty within the time allotted. This measure ensures that a node does not stay awake in an unexpected state for a long period of time draining power. In the case that the reset does take effect, the Watchdog timer (as described in the next section) will ultimately reset the node, thus ensuring that no remote node hangs indefinitely.

The current implementation of *ThresholdSentry* used in conjunction with *RemoteSensing* allows the network to record the response of longer-duration, lower-frequency events such as high wind; however, it does not support capturing short-term, transient events such as an earthquake. The time required to wake the network and perform time synchronization prior to the collection of data would cause such events to be missed. In future network development, the network wakeup time could be reduced using a propagating wakeup message with optimized communication parameters and the order of data collection and synchronization could be switched to facilitate faster initiation of sensing.

4.3 Watchdog timer

The issue of network stability is one that has plagued long-term applications of wireless sensor networks. Nodes within a network can fail due to power depletion, physical damage, and a number of other known and unknown reasons. At times, an otherwise "healthy" node may become hung-up during its operation and can only resume operation upon being reset. The reset can come in the form of some type of software reset or a hard reset where power is temporarily removed from the node. The network must continue to function even with the loss of one or more nodes or in cases when the expected operations stall.

One approach to limiting network hang-ups is to implement a Watchdog timer (*WDT*, Murphy 2000) on the sensor nodes. Such a system triggers the node to reset in the case that it behaves unexpectedly (hangs) or does not receive an external signal (i.e. from the base station) within a given timeframe. As illustrated in Figure 4.13, the processor (controlled by a higher level application) periodically restarts the *WDT*. If the *WDT* is allowed to expire, i.e. does not get restarted by the processor (due to a command from the application), it resets the processor. This process ensures that if the processor gets hung up, it will eventually be reset and return to a refreshed/operable state. A TinyOS *WDT* module has been adapted for a network of Imote2s. In addition to the *WDT*, the network can be designed to automatically preemptively reset on a timer. This functionality has been validated on the full-scale bridge implementation, the Stawamus Chief Pedestrian Bridge, which is presented in Chapter 7.

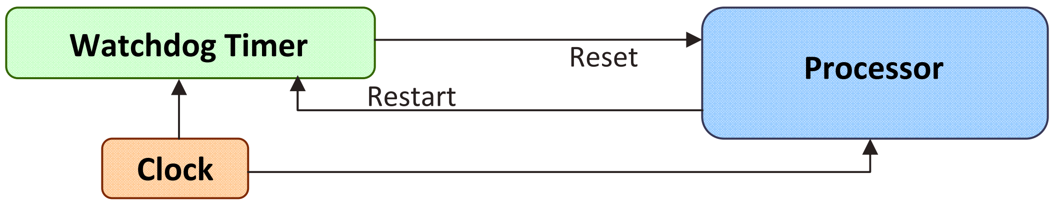


Figure 4.13 *WDT* implementation.

4.4 Autonomous operation

Achieving an autonomous SHM implementation on a network of smart sensors requires a high-level application to coordinate each of its components in response to various events. *AutoMonitor* has been developed to provide this functionality and is described in this section.

AutoMonitor is present on the base station node and coordinates the following primary tasks:

- Define the *RemoteSensing* network and sensing parameters
- Define the *ThresholdSentry* sentry network
- Setup the *ThresholdSentry* parameters
- Start the *ThresholdSentry* component
- Wakeup the network and initiate *RemoteSensing* when the threshold value has been exceeded on a sentry node
- Automatically generate data files when *RemoteSensing* occurs
- Automatically generate log files of the local node debug output
- Enforce the maximum number of allowed network sensing events in a specified time period.

AutoMonitor is initiated via an input file that sets the parameters for each of the tasks it coordinates. Once started, it requires no additional input from the user. *AutoMonitor* can be stopped via a BluSH command (AutoMonitorStop) at which point *RemoteSensing* or other network operations can be carried out manually (with BluSH commands). *AutoMonitor* is restarted again with the input file. The input parameters are defined in Table 4.8. The selection of most of these parameters is highly application-dependent and will take a period of adjustment and refinement to optimize for each case. Many of these parameters have power consumption implications; their effect on power management is addressed in Chapter 6.

Table 4.8 *AutoMonitor* input parameters

Input Line	Input Line Description	Argument	Description
RSSSetup	Remote synchronized sensing setup	nodeIDs	Node IDs of nodes in entire network
GDSetup	Sensing parameters for *RemoteSensing*	channelMask	channels involved in network-wide sensing
		numSamples	number of samples requested in network-wide sensing
		samplingRate	sampling rate for network-wide sensing
SentrySetup	Setup for *ThresholdSentry*	channelMask	channels involved in threshold check on sentry node
		samplingRate	sampling rate for threshold check on sentry node
		checkTime	duration of sensing for threshold check on remote node
		checkInterval	time between sentry checks
		threshold	value checked against in *ThresholdSentry* (in mg)
		rsmax	maximum number of *RemoteSensing* events allowed in a given time period
		period	time period outputting a debug log and resetting the number of sensing events
THSentryStart	Start of *ThresholdSentry* timer	nodeIDs	Node IDs of sentry nodes

Figure 4.14 provides a simple illustration of how the local node manages network operations in *AutoMonitor*. After the input file containing all of the parameters listed in Table 4.8 is read, *AutoMonitor* initiates *ThresholdSentry*. *ThresholdSentry* continues operating (moving through the list of sentry nodes at the specified interval, *checkTime*) until the threshold is exceeded on one of the sentry nodes. When the base station node receives the flag that the threshold has been exceeded, it first checks whether the maximum number of *RemoteSensing* events, *rsmax*, in a set time period has been reached.

If *rsmax* has been reached, *ThresholdSentry* is resumed. Otherwise *AutoMonitor* sends a command to wake the network. Once all nodes are awake, or the wakeup command times out, *AutoMonitor* initiates *RemoteSensing* with the successfully woken nodes. After *RemoteSensing* completes, when all data is finished being written, *ThresholdSentry* is reinitiated. A timer runs in the background to keep track of the set time period specified in the input file. When this time period has elapsed, the count of *RemoteSensing* events, *rscount*, is set back to zero and the debug output for the last time period is saved to a file. For example, the application may be limited to two *RemoteSensing* events within 24 hours, with the debug output being written to a file once a day.

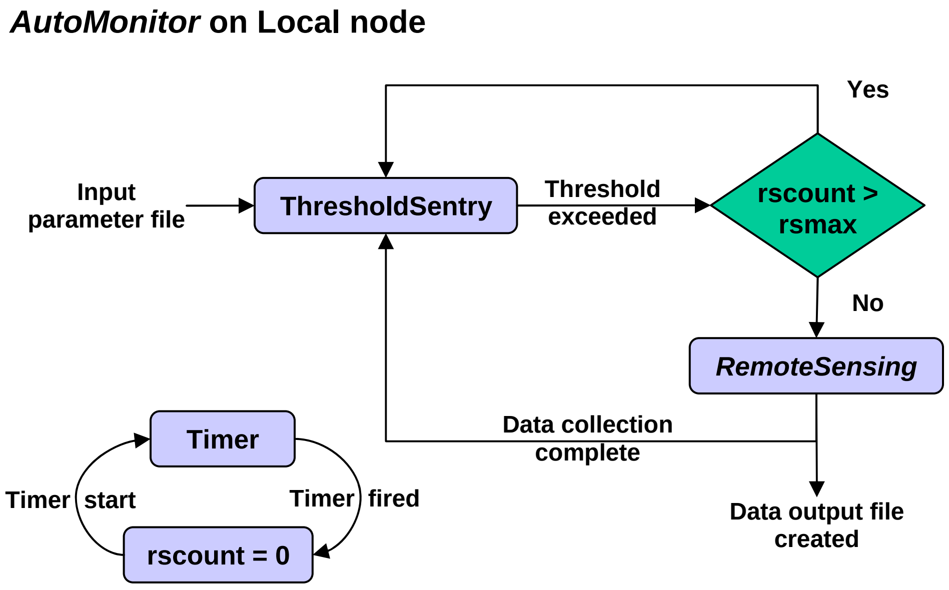


Figure 4.14 *AutoMonitor* operation on base station node.

One of the safeguards built into the *AutoMonitor* applications takes advantage of the `SnoozeAlarm.awake` event. This event is signaled after the `SnoozeAlarm.wakeup` command is executed. The arguments of the `awake` event are the nodes that were successfully woken up. *AutoMonitor* initiates network sensing with the nodes that were responsive to the wakeup command, ensuring greater probability of successful network sensing. In this way, the wakeup command acts to establish the nodes that should be included in network-wide operations.

A second safeguard feature of the *AutoMonitor* application is that the periodic node reset associated with the *WDT* has been disabled. It is not desirable for the base station node to periodically reset as it must maintain the input parameters in volatile memory throughout its operation. For example, if the time interval between *ThresholdSentry* checks is 20 minutes, the base station node will be idle during that time, not sending or receiving any messages. The *WDT* is only reset by the application when a task is performed or the node sends or receives messages. If the *WDT* is set to an interval less than 20 minutes, it will reset the node between the sentry check events, causing all of the network parameters to be lost and *AutoMonitor* to halt operation until it can be restarted again externally.

The validation of the *AutoMonitor* network management application on the Jindo Bridge in South Korea is described in detail in Chapter 7.

4.5 Summary

This chapter has presented key software components that enable continuous operation of WSSNs for SHM applications outside of the laboratory setting. This software allows critical structural response to be captured while maintaining low-power network operation the majority of the time. The *AutoMonitor* network management application coordinates the operation of *ThresholdSentry*, *SnoozeAlarm* and *RemoteSensing* to ensure autonomous and continuous functionality of the network. Chapter 6 illustrates how the software presented in this chapter impacts effective power consumption of the network, even when more power hungry sensor components are utilized (as presented in the following chapter). The safeguard measures built into the software help ensure that it runs with minimal external interaction and can maintain its functionality despite unexpected events or network behavior. The Jindo Bridge implementation discussed in Chapter 7 illustrates the effectiveness of each of the software components presented in this chapter.

HIGH-FIDELITY SENSOR BOARD DEVELOPMENT

This chapter presents the development of sensor hardware designed specifically for a broad range of SHM applications. This sensor hardware interfaces with the only commercially available smart sensor platform that is well suited to the demands of such applications. Until now, the vibration monitoring sensors that have been widely available for smart sensor platforms, and in particular the Imote2, have lacked user-selectable anti-aliasing filters, flexibility in the choice of sensing parameters, sample rate accuracy, and temperature correction. The multimetric sensor board described in the following chapter addresses these issues.

The smart sensor platform used in this research is the Imote2 (Figure 5.15). A detailed description of the Imote2 can be found in Chapter 2. The onboard memory of the Imote2 (refer to Table 2.2 for specifications) is one of the features that sets it apart from other wireless sensor platforms and allows its use for the high-frequency sampling and computationally intense data processing required for dynamic structural monitoring.

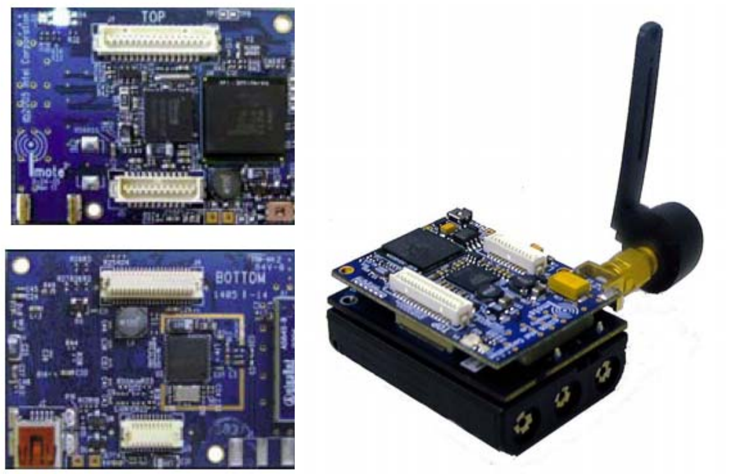


Figure 5.15 Imote2 top and bottom view (left) and stacked on battery board with antenna (right).

Vibration-based SHM requires sensed data that well represents the physical response of the structure both in amplitude and phase. The measurements must have ample resolution to characterize the structural response and must be recorded with a consistent sample rate that is synchronized with other sensed data from the structure. Whether the data is used to perform modal analysis, system identification or vibration-based damage detection, these aspects of the data quality must be met so that reasonable results may be achieved (Nagayama et al. 2007). To be used in SHM applications, the sensor hardware that interfaces with the Imote2 must provide such high-fidelity data.

The only commercially available accelerometer sensor board that interfaces with the Imote2 (ITS400C sensor board, Crossbow 2007b) has been evaluated to determine its appropriateness for SHM applications. The results of this evaluation demonstrate the need for a more flexible sensor board designed specifically for SHM applications. The design and testing of a newly developed Structural Health Monitoring Accelerometer (SHM-A) board is presented and experimentally verified in this chapter. The sensor board provides user-selectable sampling rates and anti-aliasing filters for a broad range of applications. The components of the sensor board have been selected to meet the requirements of vibration-based SHM applications, specifically with respect to data quality and the demands of achieving synchronized sensing.

5.1 Imote2 sensor interface

The Imote2 does not possess intrinsic sensing capabilities, but rather provides a flexible platform for a range of sensing applications. The sensors used with the Imote2 are interfaced to the main board via two connectors in a stackable configuration (see Figure 5.16). The Imote2 does not have an onboard analog-to-digital converter (ADC) and therefore is only compatible with digital inputs. The options available for communication with the Imote2 are I2C (which allows interface to an unlimited number of channels), 3 SPI ports (serial data ports limited to one channel per port), and multiple GPIO (general purpose I/O) pins (Intel 2005). The flexible sensor interface on the Imote2 allows its users to tailor sensor boards to their application.

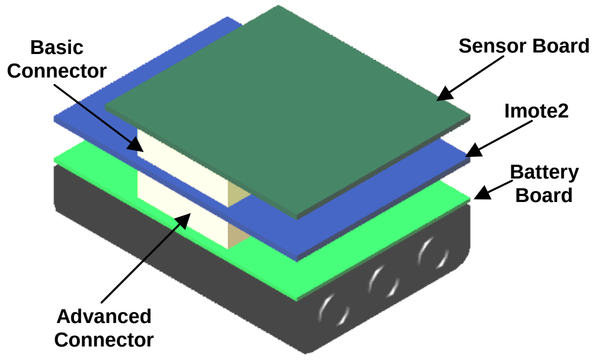


Figure 5.16 Stackable configuration of Imote2.

5.2 Evaluation of the ITS400C Imote2 sensor board

Until now, the only option for vibration sensing with the Imote2 is to utilize the sensor board developed by Intel, now available from Crossbow. This ITS400C has a three-axis digital accelerometer (LIS3L02DQ, ST Microelectronics 2005a), a light sensor (TSL2561), a temperature and relative humidity sensor (SHT15), a second temperature sensor (TMP175), and incorporates a 4-channel, 12-bit ADC (Maxim 1363).

The LIS3L02DQ has a built in ADC with digital filters with four selectable cutoff frequencies and corresponding sampling rate options which are selected by setting a decimation factor as shown in Table 5.9. (STMicroelectronics 2005a). A potential drawback of the ITS400C sensor board for SHM applications could be its lack of flexibility in selecting the cutoff frequency and sampling rate for data acquisition. The lowest sampling rate is 280 Hz which may be too high and therefore a waste of resources in systems deployed on civil structures with frequency responses less than 20 Hz. The specifications for the accelerometer state that when one of the given decimation factors is specified, the cutoff frequency and the sampling rate will be within 10 percent of the selected values. This is a potentially large variation in sampling rates which would undermine the ability to acquire synchronized data from a network of sensors. Subsequent tests to evaluate this behavior are described in the following paragraphs.

Table 5.9 LIS3L02DQ user specified sampling rates and cutoff frequencies.

Decimation factor	Cutoff frequency (Hz)	Sample rate (Hz)
128	70	280
64	140	560
32	280	1120
8	1120	4480

Calibration testing of the Imote2 with the ITS400C sensor board was conducted on a bench-scale shake table (Quanser 2006). The purpose of these tests was to determine the performance of the Imote2s and the ITS400C sensor board, including the noise characteristics, low frequency response, sampling rate accuracy, etc.; however, this paper will only discuss the results associated with the sampling rate accuracy as it is the most critical to successful vibration based SHM.

The Imote2s were fixed to the shake table along with a reference accelerometer and the response signals were compared. Several input types were used to fully characterize the performance of the Imote2 and the ITS400C sensor board, including band-limited white noise and periodic square waves. To facilitate the synchronization of the reference sensor and the Imote2, the reference sensor was sampled at 2 kHz while the Imote2 decimation factor was set to 64 to obtain a sampling rate of 560 Hz. By sampling the reference sensor at such a high rate, the reference sensor could be easily matched to the Imote2 signal and then down-sampled to the lower sampling frequency.

The results of a single sensor referenced to the reference sensor show excellent agreement. Figure 5.17(a) is the power spectral density functions (PSD) of an Imote2 and the reference sensor both resampled to 300 Hz (with the first 50 Hz shown to represent the frequency range excited by the shake table).

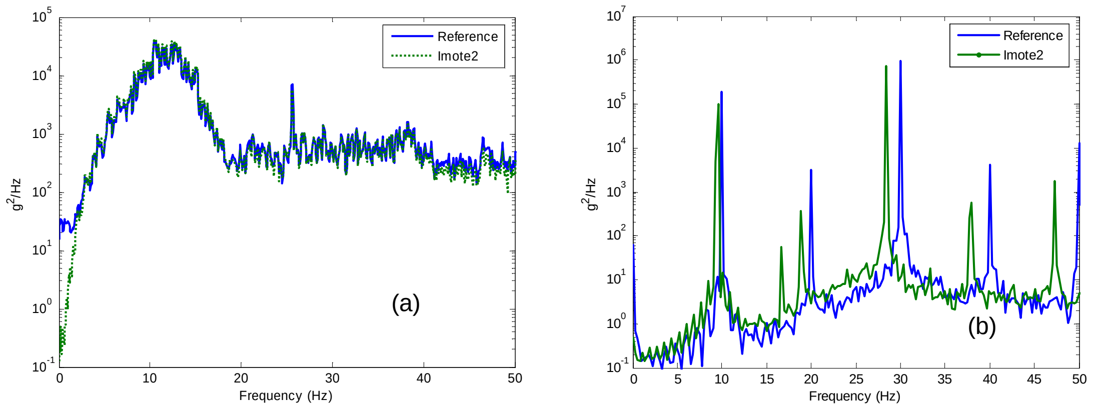


Figure 5.17 PSD of Imote2 and reference sensor subjected to: (a) 10 Hz BLWN, and (b) 10 Hz square wave.

Although the results of single Imote2s compared to the reference sensor proved to be very good, further investigation showed that the sample rate of the Imote2 sensor boards was inconsistent between each sensor board when the same decimation factor was specified. Figure 5.17(b) shows the PSD response of the reference sensor and one of the sensor boards to a 10-Hz square wave when both were assumed to have the expected sampling rate of 560 Hz. The peaks in the plots of the PSDs do not match along the frequency axis. As expected, the peaks of the reference sensor represent the harmonics of the 10-Hz square waves and appear at 10 Hz, 20 Hz, 30 Hz, etc.; however, the peaks of the data measured on the Imote2 are not aligned to the correct frequency values when the PSD is calculated assuming a sampling rate of 560 Hz. This result indicates that the sampling rate of the Imote2 deviates from the expected 560 Hz.

In total, 14 sensor boards were tested to determine the sampling frequencies of each board to an accuracy of 0.1 Hz. The sample rate was found to be different for each sensor board, varying from 537 Hz to 605 Hz for the expected sampling rate of 560 Hz as illustrated in Figure 5.18.

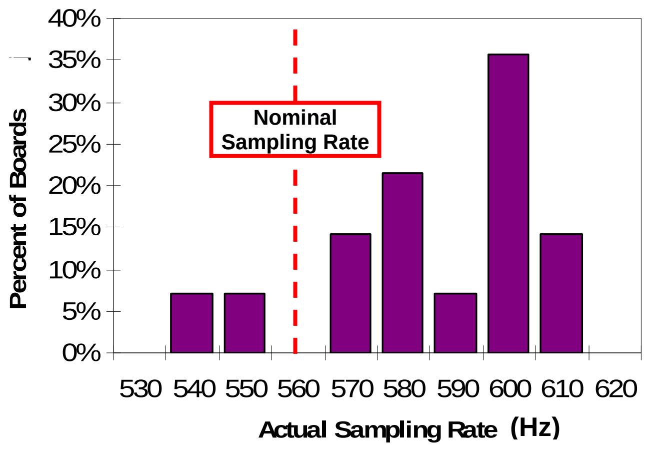


Figure 5.18 ITS400CA sample rate variation.

This range is within the limits of the specified maximum variation given by the manufacturer (± 10%, ST Microelectronics 2005c), but is truly problematic for sensing applications where accurate sampling and phase information are critical. For example, if signals from sensors with non-uniform sampling frequency are used for modal analysis, one physical mode may be identified as several modes spread around the true natural frequency. Additionally, tests conducted by Nagayama et al. (2006) showed that there was non-stationary fluctuation in the sample rate on each sensor of up to 0.1%.

5.3 Imote2 sensor board development

This section addresses the lack of Imote2 sensing hardware suitable for SHM applications that was demonstrated in the previous section. The majority of SHM applications are based on measured ambient vibration response and require high-fidelity data to ensure that modal analysis and damage detection algorithms, among other processing techniques, provide reasonable results. To encompass a wide range of potential applications the sensor hardware must allow flexibility in the sample rate as well as provide user-selectable anti-aliasing filters. Finally the hardware must be available to a wide audience to drive the advancement of WSSN for SHM.

The first phase in developing an effective sensor board for SHM using the Imote2 was to create a high-quality accelerometer board. This board incorporates three-axes of high-sensitivity accelerometer measurements with a high-resolution ADC that provides user-selectable sampling rate and anti-aliasing filters. The next phase in SHM sensor development for the Imote2 was to incorporate temperature, humidity, and light sensors. The temperature sensor allows the signals from the sensors to be calibrated to account for temperature changes as well as give more insight to the structural response under varying environmental conditions.

The following sections provide the details of each stage of the sensor board development and the solutions that were implemented in each subsequent revision. At each stage of development, the sensor boards undergo calibration and noise performance testing to evaluate and validate the designs.

5.3.1 SHM-A basic design

The key component of the Structural Health Monitoring Accelerometer (SHM-A) board described herein is the Quickfilter QF4A512, a versatile, 4-channel ADC and programmable signal conditioner with user-selectable sampling rates and programmable digital filters (Quickfilter 2007a). The board interfaces with the Imote2 via SPI I/O and has a 3-axis analog accelerometer for vibration measurement. A block diagram of the components of the SHM-A sensor board is given in Figure 5.19. Figure 5.20 shows three views of the board. The details of each component on the board will be discussed in subsequent paragraphs.

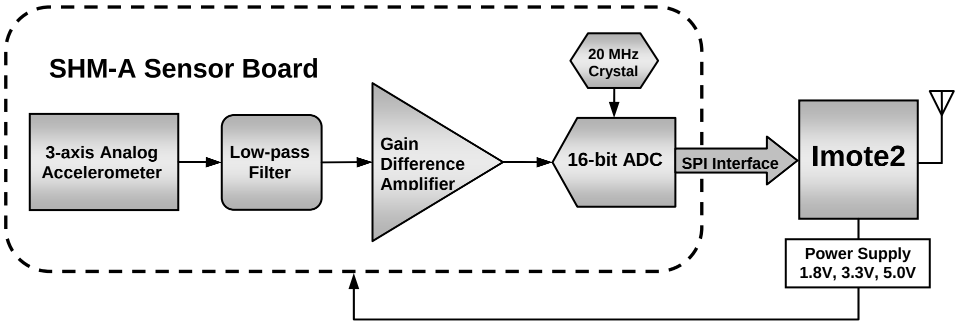

SHM-A Sensor Board
3-axis Analog Accelerometer
Low-pass Filter
Gain Difference Amplifier
20 MHz Crystal
16-bit ADC
SPI Interface
Imote2
Power Supply 1.8V, 3.3V, 5.0V

Figure 5.19 Block diagram of first SHM-A sensor board.

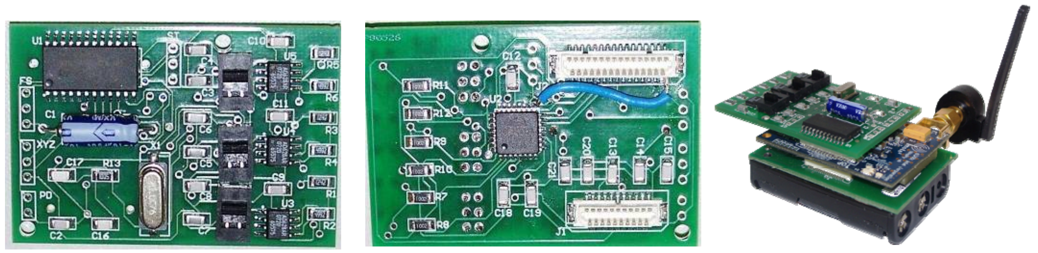


Figure 5.20 Top view of the first SHM-A board (left), bottom view (middle) and stacked on Imote2 and battery board (right).

The ST Microelectronics LISL302AS4 capacitive-type MEMS accelerometer with DC to 1500 Hz measurement range is chosen for the SHM-A board (ST Microelectronics 2005b). This type of accelerometer utilizes the motion of a proof mass to change the distance between internal capacitive plates, resulting in a change of output voltage in response to acceleration. Though MEMS accelerometers are available with lower noise levels, the ST Micro accelerometer offers an excellent price/performance ratio. In addition, it offers 3-axes of acceleration on one chip. For these reasons, this accelerometer was selected for the SHM-A sensor board. The specifications for the LIS3L02AS4 accelerometer are given in Table 5.10. If lower noise characteristics are required for a specific application, a higher-cost accelerometer, such as the SD1221 (Silicon Design 2007) or the Si-Flex SF1500S (Colibrys 2007), could be incorporated into the board design with appropriate measures to accommodate higher power requirements.

Table 5.10 LIS3L02AS4 accelerometer specifications.
(ST Microelectronics 2005b)

Parameter	Value
Axes	3
Measurement Range	±2 g
Resolution	0.66 V/g
Power Supply	2.4 V to 3.6 V
Noise Density	50 μg/√Hz
Temperature Range	-40 to 85˚C
Supply Current	0.85 mA

A design limitation of the ST Micro accelerometer is that it has a high output impedance with a large margin of error in the specified resistor value. Special care must be taken to compensate for the high output impedance and avoid the introduction of error into the output signal. These measures will be discussed in the following paragraphs.

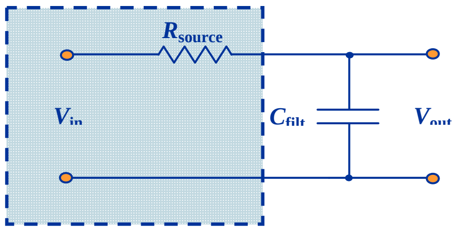


Figure 5.21 Diagram of the RC filter created by the internal resistor and external capacitor at the interface of the accelerometer output.

An internal resistor on the LIS3L02AS4 accelerometer is in series with an external user-selectable capacitor to form a single-pole low-pass RC filter. The value of the internal resistor (R_{source}) is 110 kΩ (±20%). According to the LIS3L02AS4 specifications, the minimum capacitor value that can be used corresponds to a cutoff frequency of approximately 1500 Hz (STMicroelectronics 2005b). The cutoff frequency of the filter is defined by the 3 dB roll-off point. This type of filter is inadequate as an anti-aliasing filter, because of its very slow roll-off (6 dB per octave). For example, the filter gain does not reach the level of the theoretical noise floor of the ADC (81 dB), until f = 11,000*f_c due to the very slow roll-off of the filter. If a measurement bandwidth of 20 Hz is desired, the data would have to be sampled at 2*11,000*20Hz = 440 kHz, to ensure that no higher frequency energy is aliased into the signal. Additionally, the filter has non-linear phase distortion. Figure 5.22 illustrates the gain and phase response of a single-pole RC filter.

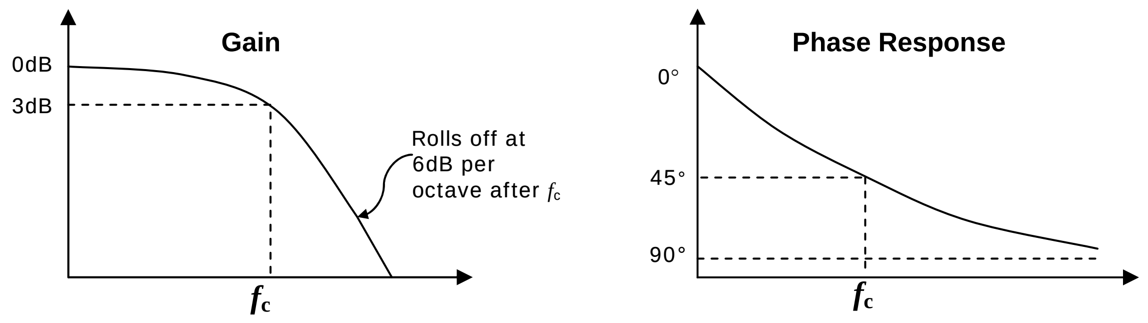


Figure 5.22 Single pole RC filter response typical amplitude response curve (left) and phase response (right).

Beyond the limitations of a single-pole RC filter to act as an effective anti-aliasing filter, the accuracy of the amplitude and phase response is subject to the accuracy of the series resistor and capacitor which comprise the filter. The potential error in the accelerometer's internal resistor (±20%) must be addressed to avoid error in the signal amplitude and phase over the bandwidth of interest. Assuming that the capacitor is known precisely, the variation in the resistance can result in a variation in the cutoff frequency which can range from -17% to +25% of the nominal value. The potential phase mismatch between channels at the 50-Hz cutoff frequency can be as high 11.5 degrees.

Figure 5.23 (a) shows the attenuation over a 100-Hz bandwidth associated with three different cutoff frequencies, and Figure 5.23 (b) shows the phase responses of the filter. Figure 5.23 (c) shows the maximum potential phase mismatch between measurement channels for three different cutoff frequencies which could result from the error in the accelerometer's internal resistor. The maximum mismatch at 100 Hz for the 500-Hz nominal cutoff frequency is 4.9 degrees while the maximum mismatch at 100 Hz for the 1500-Hz cutoff frequency is 0.5 degrees.

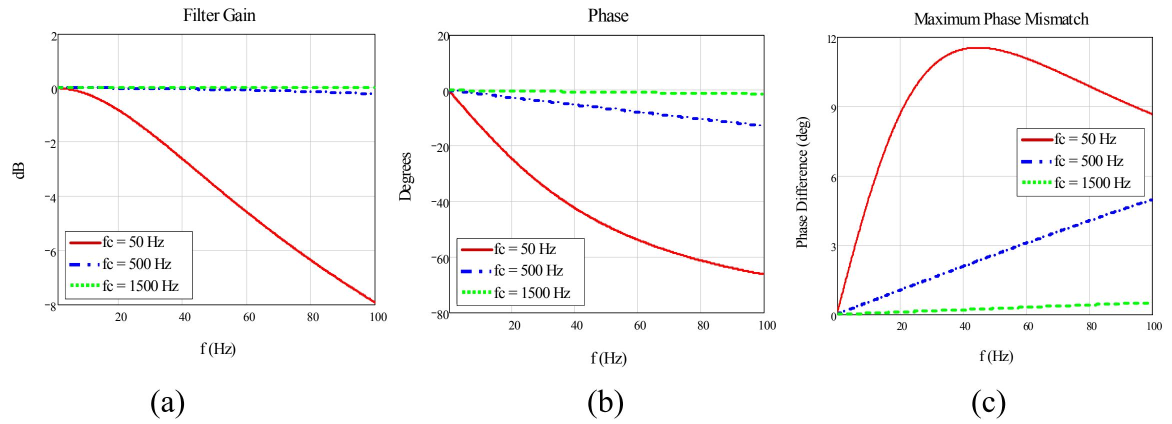


Figure 5.23 (a) Single-pole RC filter transfer function, (b) phase response and (c) maximum phase mismatch due to errors in the accelerometer's internal resistor.

To minimize phase and amplitude errors, the highest possible cutoff frequency should be selected for the accelerometer. The initial design of the SHM-A board configuration allowed the user to select between the 50 Hz and the 500-Hz cutoff frequency. Subsequent board revisions fixed the cutoff frequency to the highest possible value (1500 Hz). Methods for addressing aliasing will be discussed in subsequent paragraphs.

A gain difference amplifier (AD628, Analog Devices 2007) follows the accelerometer and low-pass filter to compensate for the high output impedance of the accelerometer. A high output impedance (R_{source}) can be problematic if the device that the output signal feeds into has a much lower input impedance (R_{load}) because the signal is attenuated in the following way:

$$V_{load} = V_{source} \frac{R_{load}}{R_{load} + R_{source}} \tag{5.2}$$

where V_{source} is the signal from the accelerometer and V_{load} is the attenuated signal. The output impendence of the accelerometer is 110 kΩ while the input impedance of the subsequent ADC is 10 kΩ. Without the amplifier, the amplitude of the resulting signal would be less than 10 percent of the original signal. The input impedance of the amplifier is 100 kΩ which results in the accelerometer signal being approximately cut in half.

The key component of the SHM-A board is the Quickfilter QF4A512 Programmable Signal Conditioner (Quickfilter 2007a). The QF4A512 employs a versatile 4-channel, 16-bit resolution ADC. Each channel has a selectable gain (up to 8x), an analog anti-aliasing filter with a 500-kHz cutoff frequency, individually selectable sampling frequencies and individually programmable digital FIR filters (up to 512 filter coefficients). A block diagram of the QF4A512 is shown in Figure 5.24.

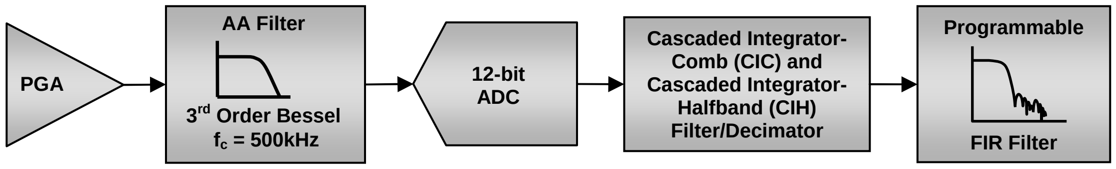
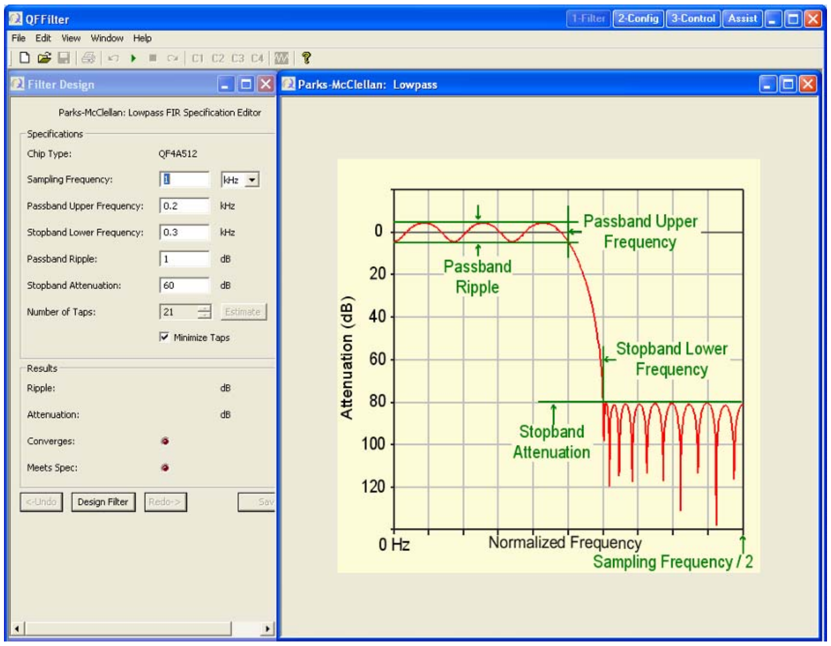

PGA
AA Filter
3rd Order Bessel
fc = 500kHz
12-bit ADC
Cascaded Integrator-Comb (CIC) and Cascaded Integrator-Halfband (CIH) Filter/Decimator
Programmable
FIR Filter

Figure 5.24 Block diagram for Quickfilter Programmable Signal Conditioner.

The QF4A512 performs oversampling, filtering, and decimation to achieve two purposes in the digitization of the measured signal. The first purpose of oversampling is to improve the resolution of the output by decreasing the noise from quantization error. The resolution of the ADC dictates the smallest measurable increment that can be resolved. Quantization introduces a constant level of noise energy which is uniformly distributed over the measured bandwidth. The higher the sampling frequency, the wider the frequency range over which the noise energy is distributed. Because the energy of the noise is constant, increasing the Nyquist frequency lowers the amplitude of the noise. When a digital decimation filter is applied to the oversampled signal, the noise energy above the new Nyquist frequency is eliminated, thereby improving the resolution of the signal. A 4-times oversampling rate lowers the quantization noise floor by 6 dB or the equivalent of achieving one additional bit in resolution.

The QF4A512 provides variable anti-aliasing filters by following the unaliased, oversampled signal with digital filtering and decimation. The analog anti-aliasing filters are 3rd order Bessel filters with a cutoff frequency of 500 kHz. The digital decimation filters are Cascaded-Integrator-Comb (CIC) filters, working in combination with the Cascaded-Integrator Halfband (CIH) filters to ensure that the integrity of the signal is maintained upon decimation to the final user-specified sampling frequency. This combination of filters provides excellent amplitude response and while preserving a linear phase response (Hogenauer 1981). This method of oversampling, filtering, and decimation to remove aliasing is common for PC-based analyzer systems such as those offered by Siglab (Spectral Dynamics 2007).

The gain, sampling rate, and user designed FIR filters are all set with the use of software provided by Quickfilter Technologies, Inc. (2007b) The user first selects the desired FIR filter type. The available filter types are Basic Parks-McClellan, Window Sync Blackman, and Window Sync Blackman-Harris with low-pass, high-pass, band-pass and band-stop options. The user then selects the final sampling rate and filter characteristics in the FIR Specification Editor as shown in a screen shot in Figure 5.24. The sampling rate can range from 6 Hz to over 100 kHz, however the maximum sampling rate is limited by the Imote2 to ~5000 Hz. The analog gain is then selected and the filter is assigned to the measurement channels. Finally, the results of the filter design and configuration are exported to a header file which is included when the sensing application is loaded onto the Imote2. Multiple configuration files can be created and stored on the Imote2. When a sensing application runs, the Imote2 then loads the requested configuration file onto the QF4A512.



Parks-McClellan: Lowpass FIR Specification Editor

Specifications

Chip Type:	QF4A512	
Sampling Frequency:	1	kHz
Passband Upper Frequency:	0.2	kHz
Stopband Lower Frequency:	0.3	kHz
Passband Ripple:	1	dB
Stopband Attenuation:	60	dB
Number of Taps:	21	

☑ Minimize Taps

Results

Ripple:		dB
Attenuation:		dB
Converges:	●	
Meets Spec:	●	

Figure 5.25 Screen shot of Quickfilter FIR filter design software interface.

The master clock of the QF4A512 uses an external signal provided by a surface mounted 20 MHz crystal oscillator (Citizen HCM-49, 2007). A phase locked loop (PLL) circuit provides the control system to generate a clock signal that tracks (or locks into) the frequency provided by the oscillator. All internal clocks are derived from the master clock through the use of dividers. The default PLL clock frequency is equal to the oscillator frequency divided by 10, or 200 MHz; the clocks that controls the ADC and FIR filters are derived from the PLL clock with additional dividers. The clock used to drive the ADC has a default frequency of 100 MHz (divider = 2) and the system clock, which runs the FIR filters, has a default frequency of 200 MHz (divider = 1). The accuracy in the effective sampling rate of the output signal depends on the accuracy of the external crystal, which in this case is specified as ±30 ppm (0.003%). The processor clock on the Imote2 is independent of the QF4A512 clocks and runs at 3.25 MHz. During sensing, the Imote2 clock only affects the time-stamps assigned to each data point coming from the QF4A512 and is assumed to be accurate, although it could also be a source of sampling rate error.

A software driver for the SHM-A board was developed in TinyOS. The purpose of the driver is to control the functions of the QF4A512 such as loading the filter coefficients, allocating memory, timestamping, writing data, etc. The driver was adapted from driver code provided by Intel and implemented for the SHM-A board.

The driver first initializes the ADC and then triggers the sampling to start. One limitation of the driver which is derived from an inherent limitation of TinyOS is the inability to accurately control the time delay between the command to begin sampling and the actual start of sampling. During sampling, the samples are released from the QF4A512 and written to the Imote2 buffers as two-byte integers (16-bit). Timestamping occurs at multiples of the sampling time. If timestamping is requested, the timestamps are written with the ADC data at this lower specified frequency (e.g. every 10 samples).

Tests were conducted to calibrate each channel of the accelerometer. The SHM-A board mounted on an Imote2 was placed on an accelerometer calibration frame which ensured a level measurement surface. Measurements were taken with the board oriented so that signals corresponding to –1 g, 0 g and +1 g were measured for each of the measurement axes. The results provided the necessary calibration constants (DC offset and scale) which can be directly implemented in the sensing application. The following paragraph describes the calibration testing conducted to evaluate the sensor boards at frequencies above the DC (static) response; the scale factors from the dynamic tests were consistent with these results from the static tests.

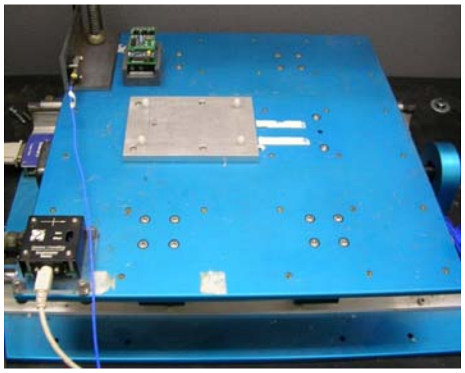


Figure 5.26 Reference sensor and SHM-A board with Imote2 mounted on a bench-scale shake table for calibration testing.

The SHM-A board was also tested on a bench-scale shake table against a wired reference sensor (Figure 5.26). The reference sensor is a capacitive accelerometer (PCB Model 3701G3FA3G, PCB 2007) with 1 V/g sensitivity and DC to 150 Hz measurement range. Several types of excitations were used to test the performance of the sensor board. Because lower frequency performance of the sensor is important for civil structures, focus was placed on the 0 to 20 Hz range. In an effort to compensate for the limitations of the shake table in the lower frequency range, a "shaped" band-limited white noise (BLWN) with a 10-Hz cutoff frequency was used. The results given in Figure 5.27 show excellent agreement between the wired sensor and the SHM-A board in the time and frequency domain. Additionally, the higher frequency range was excited by a BLWN with a 50-Hz cutoff frequency as shown in Figure 5.29. In both cases, there is some discrepancy in the lower frequency ranges, <2Hz for the 10-Hz shaped BLWN excitation (Figure 5.28) and <5Hz for the 50-Hz BLWN (Figure 5.30). The reason for this error is that it is difficult for the shake table to excite the very low frequency range due to the limitation of its stroke, especially when attempting to excite a broad range of frequencies. The result is signals in these ranges are too small to achieve a good comparison. The results of the static tests presented earlier validate the DC response of the sensors.

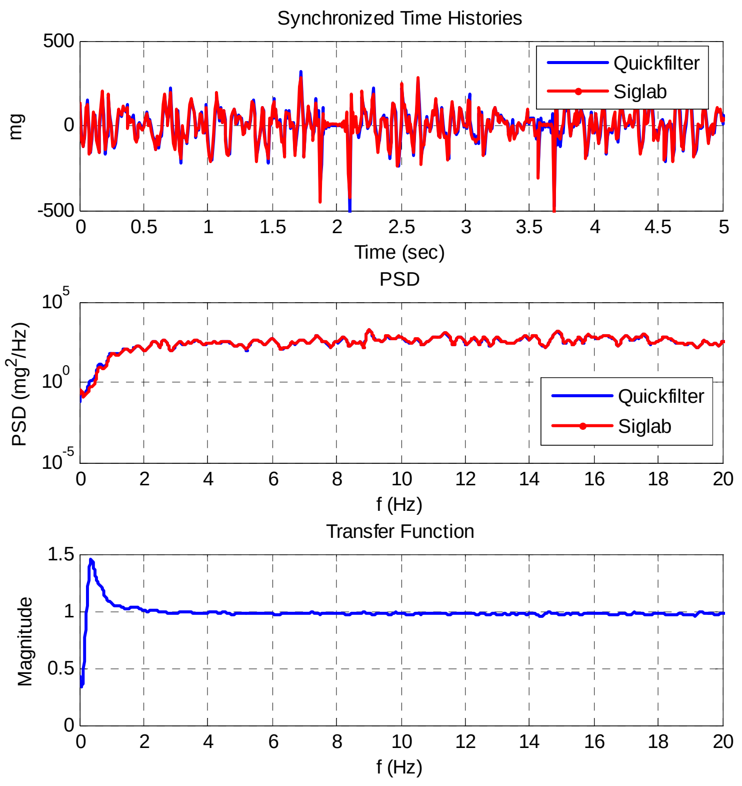


Figure 5.27 Time history and power spectrum plots of shake table tests for a "shaped" 10-Hz band-limited white noise excitation.

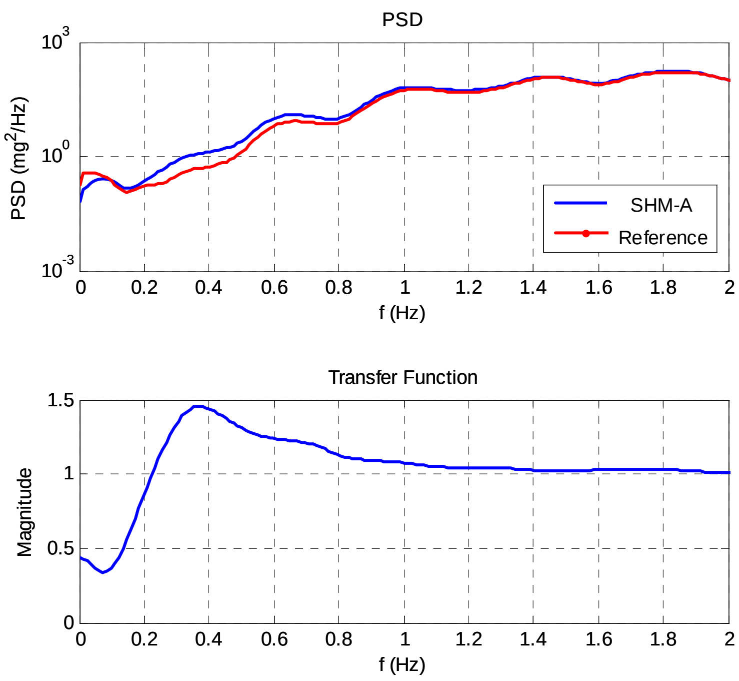


Figure 5.28 Zoomed PSDs (top) and transfer function (bottom) between the SHM-A sensor board and the reference sensor for the "shaped" 10-Hz band-limited white noise excitation shake table tests.

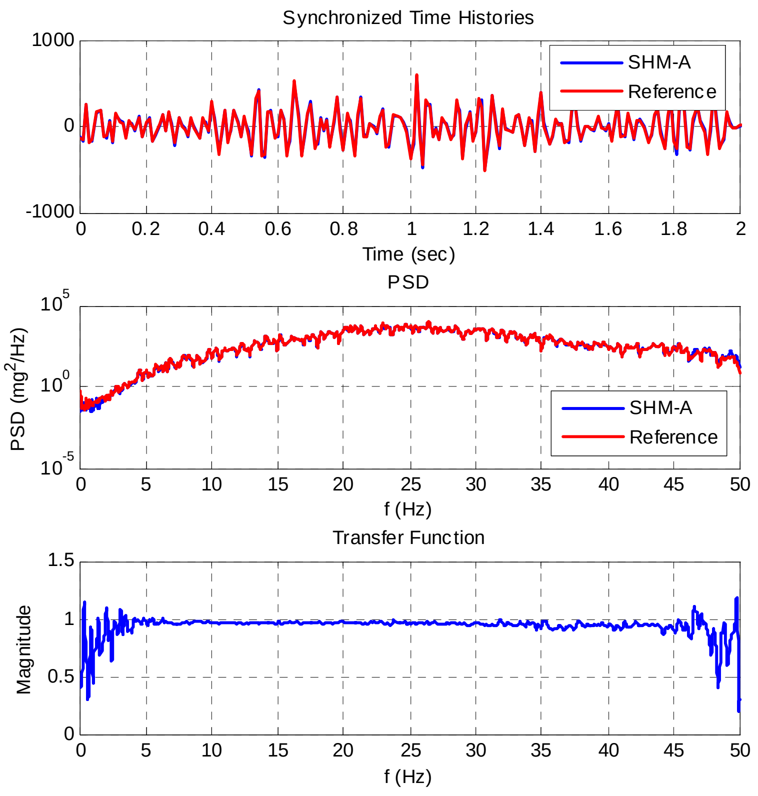


Figure 5.29 Time history, power spectrum, and transfer function plots of shake table tests for a 50 Hz band-limited white noise excitation.

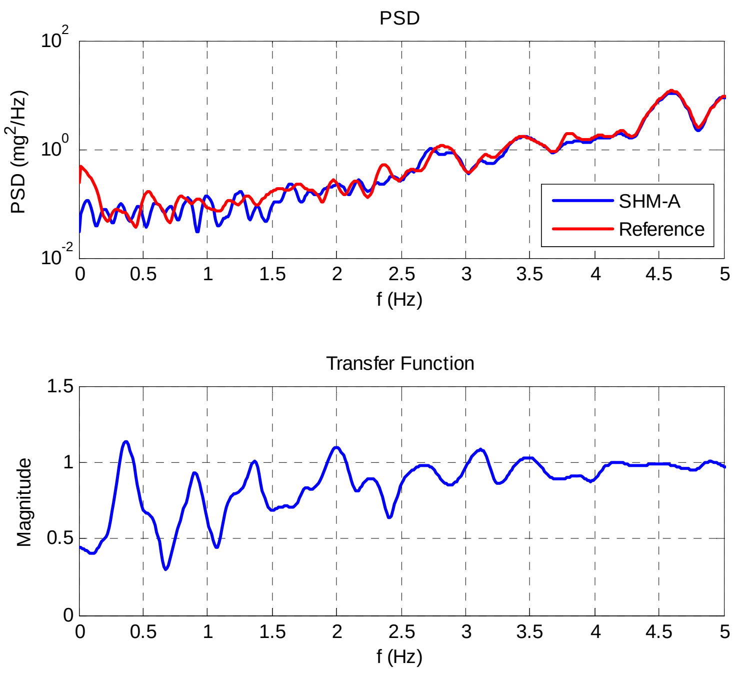


Figure 5.30 Zoomed PSDs (top) and transfer function (bottom) between the SHM-A sensor board and the reference sensor for the 50-Hz band-limited white noise excitation shake table tests.

Tests were conducted to quantify the noise floor and resolution of the initial SHM-A sensor board. The Quickfilter ADC has a nominal resolution of 16 bits. The acceleration range of the ST Micro accelerometer is ±2 g at a sensitivity of 0.66 V/g and a zero-g offset equivalent to half of the supply voltage. The gain difference amplifier used in the first SHM-A design results in a halving of the output of the accelerometer; therefore the full range is 2g*0.66 V/g = 1.33 V. Each channel input to the QF5A512 has two input pins. The input on each pin must be between 0.2 and 2.5V and the difference between the pins must be between 0 and 1V. Series resistors directly prior to the input pins for each channel serve to shift and scale the input signals so that they satisfy these requirements. The values of the series resistors used in this first SHM-A board revision were not optimally sized to take advantage of the full range of the ADC; the full range of the accelerometer output (±2g = 4g) only takes up 40% of the ADC range. Based on this design, the theoretical nominal resolution of the output signal is 0.15 mg for a 40% of a 16-bit ADC. However, the signal-to-noise ratio (*SNR*) of the ADC specified in the QF4A512 data sheet that results from noise within the device is given as 81 *dB*, which corresponds to 13.2 effective number of bits (*ENOB*). Using 40 percent of the 13.2 bit ADC range yields a resolution of 1.1 mg. Eq. 5.2 gives the relationship between *ENOB* and the noise floor (in *dB*). As discussed previously, oversampling can result in an increase in the number of bits achieved beyond those realized in hardware.

$$ENOB = \frac{SNR - 1.76}{6.02} \tag{5.3}$$

Inherent noise is present in the accelerometer and other components, in addition to the noise resulting from the ADC quantization. The noise density of the accelerometer is given as 50 μg/√Hz. The relationship between the noise density, N_d and the *RMS* noise level, N_{RMS}, measured for a particular measurement bandwidth (*BW*) is given in Eq. 5.3.

$$N_{RMS} = 0.707 \cdot N_d \cdot \sqrt{BW} \tag{5.4}$$

Over a 128-Hz bandwidth the specified accelerometer noise density corresponds to an *RMS* noise level of 0.4 mg.

The actual *RMS* noise level of the accelerometer was determined by conducting static tests with the SHM-A board resting on a rubber-backed aluminum plate and placed on the concrete strong floor of the Smart Structures Technology Laboratory in the basement of Newmark Civil Engineering Laboratory. The measured value for all three channels was determined to be approximately 1.3 mg over a bandwidth of 128 Hz. Eq. 5.4 relates this *RMS* value to the noise floor for the same bandwidth.

$$SNR = 20\log\left(\frac{FSR}{N_{RMS}}\right) \tag{5.5}$$

where *FSR* is the full scale measurement range. In the case of the LIS3L02AS4 accelerometer it is 4g. The resulting equivalent noise floor is 70 dB and (from Eq. 5.2) 11.3 *ENOB*. Table 5.11 summarizes the theoretical and measured acceleration RMS noise values for the first SHM-A board design.

Table 5.11 Theoretical and measured *RMS* noise values for the first SHM-A board.

Source Component	*RMS* noise (BW = 128 Hz)
QF4A512 ADC (theoretical)	1.1 mg
LIS3L02AS4 Accelerometer (theoretical)	0.40 mg
SHM-A Rev. 1 Sensor Board (measured)	1.3 mg

The measured *RMS* noise matches well with the expected noise based on using 40 percent of the range of the QF4A512 ADC with 13.2 *ENOB*. The noise floor of the acceleration output can be improved in a few ways. One improvement is to eliminate the difference gain amplifier and replace it with an operational amplifier with a very high input impedance to reduce signal attenuate prior to the QF4A512. A second improvement is to correctly size the series resistors on the input to the QF4A512 to take full advantage of the ADC range. These design changes are explored in the subsequent revisions presented in the following section. Also, as previously mentioned, a different accelerometer with lower noise characteristics could be used which would be expected to result in lower noise levels in the final output.

The specified supply currents and typical supply voltages for each of the components or the SHM-A sensor board are given in Table 5.12. The values shown for the QF4A512 are assuming three-channel operation at a 1000-Hz sampling rate. The QF4A512 utilizes both 3.3 V and a 1.8 V power supplies for various functions within the chip, both supplied by the Imote2. According to the specifications (Quickfilter 2005), the power consumption of the QF4A512 is a function of the number of active channels and the sampling rate; the number of FIR filter coefficients has negligible effect.

Table 5.12 Specified power consumption of original SHM-A components.

Component	Active Current (mA)	Supply Voltage (V)	Power (mW)
Accelerometer (LIS3L02AS4)	0.85	3.3	2.8
Gain Difference Amplifier (AD628)	1.6	5.0	8.0
QF4A512 PGA	12.7	3.3	41.9
QF4A512 anti-aliasing filters, ADC, clocks	72.8	1.8	131.0
QF4A512 FIR filters	6.8	1.8	12.3
QF4A512 SPI operation	0.06	3.3	0.2
Total Power			196.2

The total estimated current draws on the various supply lines during sensing are 13.6 mA on the 3.3V supply, 79.6 mA for the 1.8V supply, and 1.6 mA for the 5V supply. Tests were conducted to measure the actual power consumption of the SHM-A board while data acquisition is taking place. A 4.5 V DC power source was connected to the battery board to represent the nominal voltage from 3-AAA batteries. The Imote2 was turned on and the current draw from the Imote2 and battery board was measured. The sensor board was then attached to the Imote2 and the current was measured again. Data acquisition was initiated and the current was observed during sensing. In this way, the incremental power consumption of the SHM-A board during sensing was estimated. The results are shown in Table 5.13 for two different sampling rates with either one active channel or three active channels. The table shows that the inactive SHM-A board attached to the Imote2 requires in an additional 10 mA of current beyond the Imote2 operating in an idle state with no sensor board. In the data acquisition mode the Imote2 and the SHM-A board acting together require approximately 110 mA (500 mW) for one active channel and 155 mA (700 mW) for three active channels. No difference was observed between the two sampling rates tested. During data acquisition, the Imote2's operating frequency is switched to 104 MHz which results in increased power consumption.

Table 5.13 Estimated SHM-A power consumption when powered by 4.5V DC power source.

		Imote2 On – No Sensor Board Attached		Imote2 + SHM-A Board – Inactive		Imote2 + SHM-A Board – Sensing	
Sampling Rate (Hz)	Active Channels	Current (mA)	Power (mW)	Current (mA)	Power (mW)	Current (mA)	Power (mW)
500	1	60	270	70	315	110	495
500	3					155	698
1000	1					110	495
1000	3					155	698

5.3.2 SHM-A improvements

A second revision of the SHM-A board was created to achieve higher resolution, a lower signal-to-noise, ratio and a simpler design and layout. These goals were achieved with some design changes and careful consideration of the component and trace layouts.

First, the accelerometer used in the previous design (LIS2L02AS4) was made obsolete and thus was replaced by its new equivalent (LIS3L02AL, ST Microelectronics 2008). As discussed in the previous section, the original design used two capacitors and switches to provide single-pole filters following the accelerometer with two possible cut-off frequencies. The switches were eliminated and the capacitor used in the second revision puts the cutoff frequency at 1500 Hz, thereby limiting error associated with the filter. The removal of the AD628 differential amplifier and addition of an operational amplifier (OPA4344, Texas Instruments 2008) eliminated the significant signal attenuation exhibited by the first board revision and thereby improved the resolution and noise performance of the board. A block diagram of the components of the Rev. 2 board is shown in Figure 5.31.

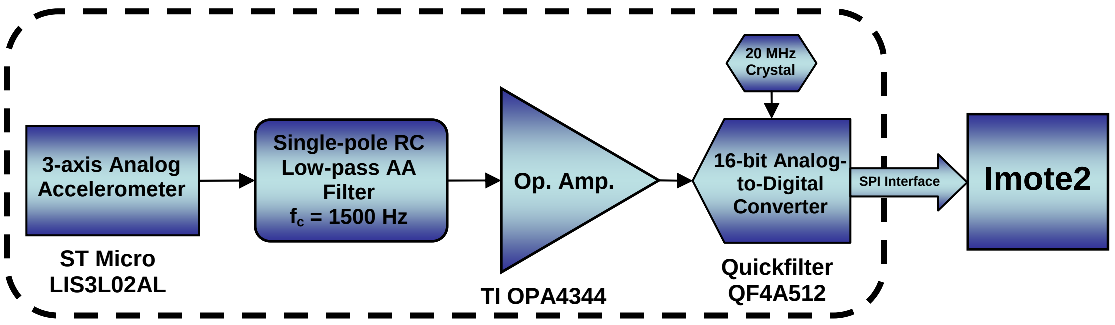

3-axis Analog Accelerometer
ST Micro LIS3L02AL
Single-pole RC Low-pass AA Filter f_c = 1500 Hz
Op. Amp.
TI OPA4344
20 MHz Crystal
16-bit Analog-to-Digital Converter
Quickfilter QF4A512
SPI Interface
Imote2

Figure 5.31 Block diagram of the SHM-A Revision 2 sensor board.

Very careful attention was paid to separating the digital and analog components of the board, avoiding ground loop interference and ensuring short and parallel digital traces, all with the intention of minimizing noise. The use of a two-layer board in this design does not allow separate power and ground planes, which help minimize signal noise and interference, however the revised board created a solid ground region on the bottom side of the board. All of the ground signals on the board connect directly to the ground region which provides a clear path for the return current. The top and bottom of the SHM-A Revision 2.0 board are shown in Figure 5.32.

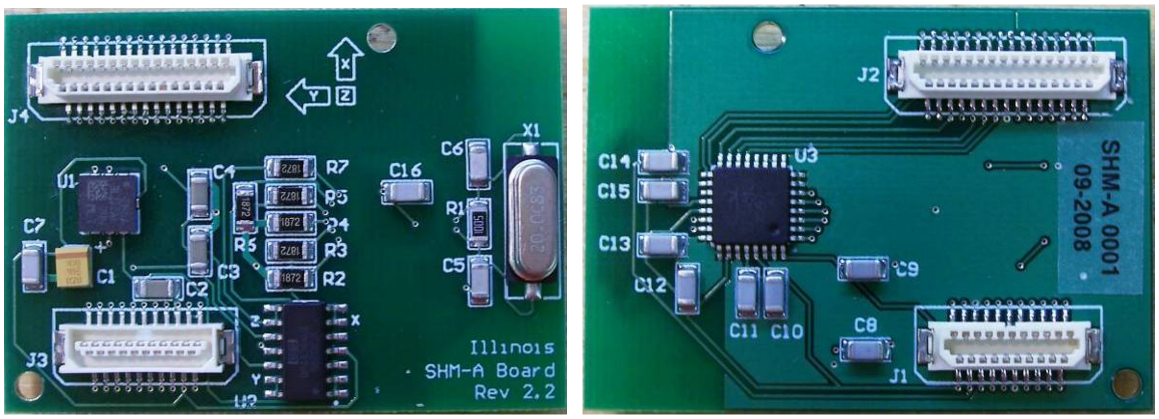


Figure 5.32 SHM-A Revision 2 top (left) and bottom (right).

Finally, the series resistors on the input to the QF4A512 were sized to take advantage of more of the range of the ADC. With the input utilizing 90-percent of the ADC range and a theoretical noise floor of 81 dB the expected *RMS* noise level of the acceleration is 0.48 mg. Similar calibration and noise performance tests were conducted on the revised sensor board as presented in the previous section. The LSB (with 1x PGA setting) is approximately 0.16 mg and the *RMS* noise level over 500 Hz is approximately 0.5 mg. The resulting SNR is 13.0 *ENOB* and the equivalent SNR is 80 dB which corresponds well to the predicted values.

The sampling rate accuracy of the QF4A512 was tested to determine how closely the actual sampling rate matches the requested sampling rate and to observe any fluctuation in the sampling rate over time. Tests were conducted by creating four QF4A512 configuration files for four different sampling rates (corresponding to the sample rates available on the ITS400C sensor boards). For each configuration file two SHM-A sensor boards and two Imote2s in different combinations were tested by acquiring 1000 timestamped data points using the *LocalSensing* application. Assuming the timestamps from the Imote2 processor are accurate they can be used to determine the average sample rate of the data and evaluate whether the sample rate fluctuates in time over the course of data acquisition.

When QF4A512 configuration files are created, the design sample rate may vary slightly from the requested nominal sample rate as the software automatically optimized the configuration to meet the filter requirements and maintain a reasonable file size. Table 5.14 shows the design sample rates corresponding to the requested nominal sample rates as well as the measured average sample rates on four combinations of Imote2s and SHM-A sensor boards. The measured sample rates are consistent for all combinations of Imote2s and sensor boards meaning all sensors within a network are expected to have the same sampling rate. Whereas the ITS400C sensor boards were observed to have up to 10 percent variation in sample rate from sensor board to sensor board, the SHM-A sensor board has no observed variation. In addition, the measured sample rates are very close to the design sample rates produced by the QF4A512 configuration software.

Table 5.14 Measured sample rate accuracy from the QF4A512.

Nominal f_S (Hz)	Design f_S (Hz)	Measured f_s (Hz)				% Error, Design vs. Measured
		Imote2(a) SHM-A(a)	Imote2(a) SHM-A(b)	Imote2(b) SHM-A(a)	Imote2(b) SHM-A(b)	
280	280.018	280.06	280.06	280.06	280.06	0.015
560	560.036	560.12	560.12	560.12	560.12	0.015
1120	1120.072	1120.2	1120.2	1120.2	1120.2	0.011
4480	4489.94	4490.8	4490.8	4490.8	4490.8	0.019

In addition to the mean sample rate, the fluctuation in the sample rate over time may be determined by taking the difference between each time stamp over the data acquisition period. Figure 5.33 shows plots for each of the nominal sampling rates of the difference between consecutive time stamps plotted versus the sample number. Very little fluctuation is observed primarily within the 1 μs resolution of the timestamp values. For the 280-Hz sample rate the temporal *RMS* variation is less than 0.02%. Since this fluctuation is centered on a mean value, the error in the sampling rate is not expected to be additive over time. This is a vast improvement over the ITS400C sensor boards which were measured to have a non-constant mean temporal fluctuation of 0.1%.

One of the observed limitations of the SHM-A sensor board that was identified in the Rev. 2 design was drift in the mean value of the acceleration output. Figure 5.34 shows the drift in the mean value of the accelerometer output of the SHM-A board over one hour. Converting the voltage drops to equivalent acceleration values, the *x* and *y* axes drift ~6 mg while the *z* axis drifts ~16 mg. According to the accelerometer datasheet, the sensor board has the potential for significant zero-g level drift due to temperature changes. The *x*-axis has the potential to drift up to -0.5mg/°C, the *y*-axis can drift up to -0.75mg/°C and the *z*-axis has the most potential to drift, up to -1.75mg/°C. While environmental heating would affect the measured values, the ambient temperature is not expected to change significantly during the course of data acquisition. However, given that the Imote2 and sensor board both generate a certain amount of heat when they operate, the relationship between the board temperature and the mean value drift was further investigated.

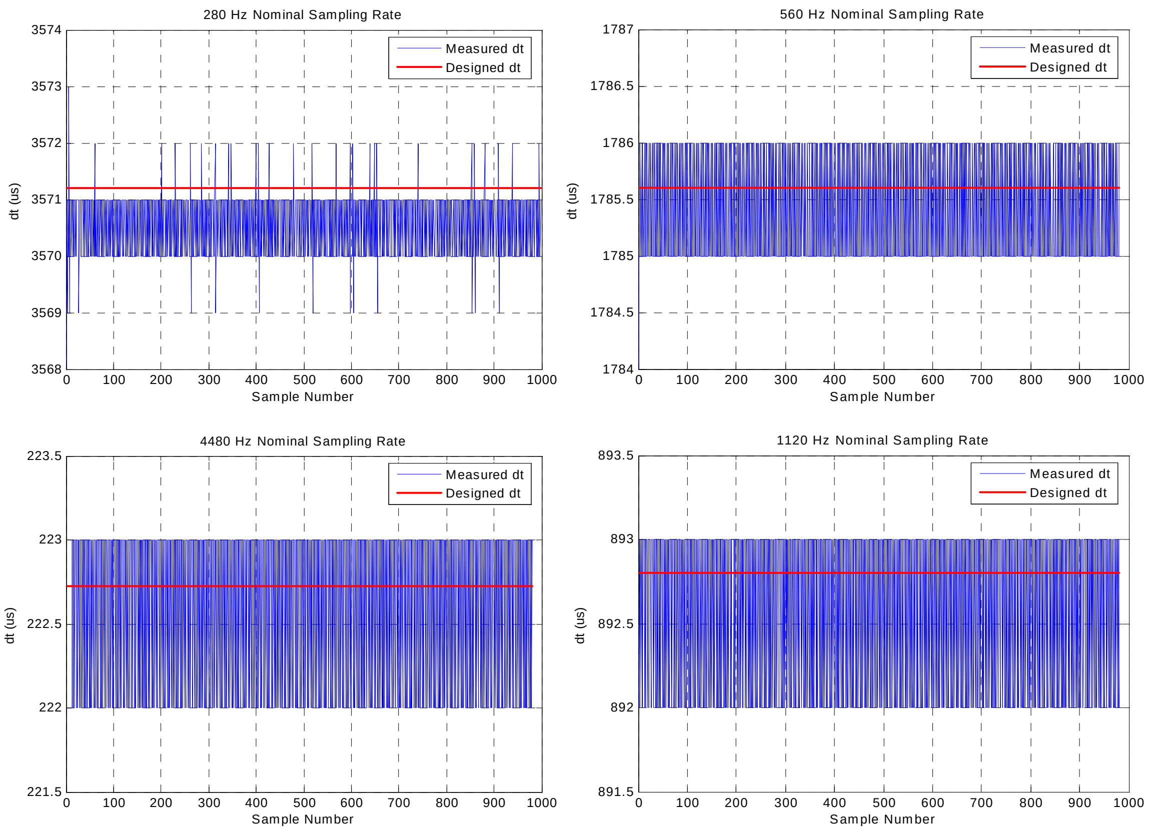


Figure 5.33 Sample rate fluctuations shown as the time between consecutive time stamps at various sample rates.

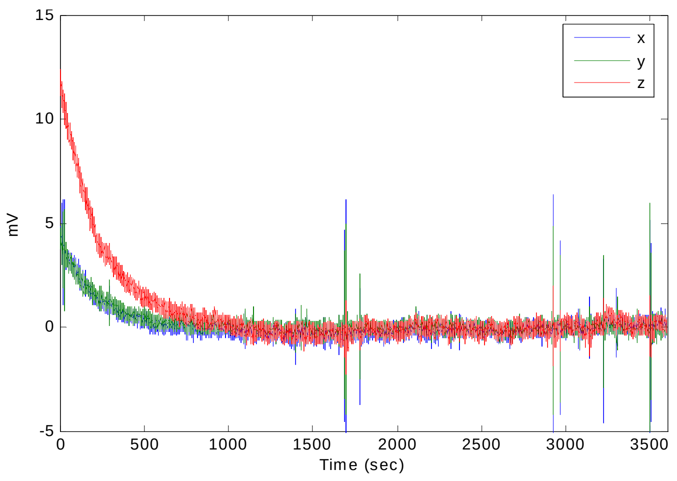


Figure 5.34 Normalized measured voltage output of still accelerometer on SHM-A board over one hour.

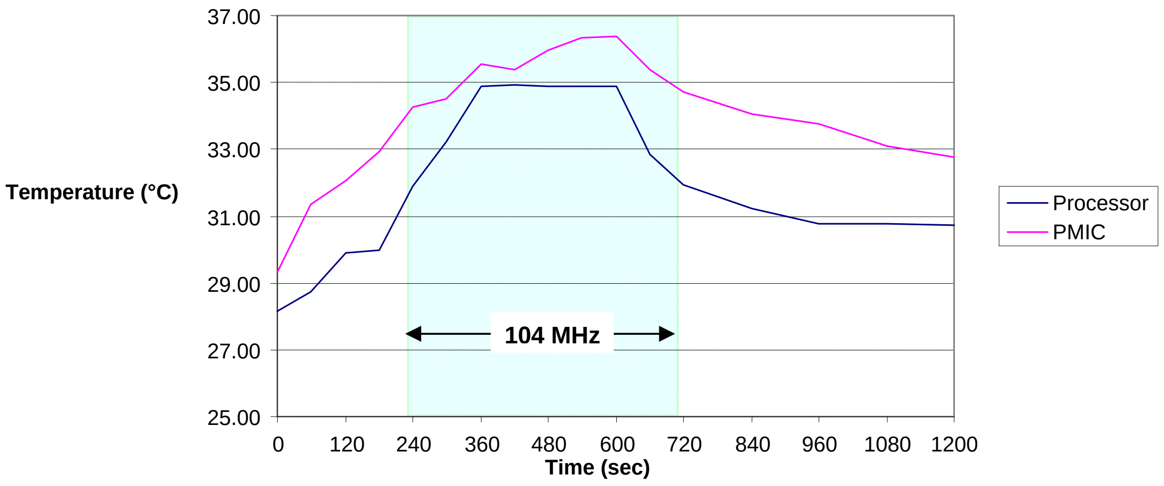


Figure 5.35 Imote2 component heating during processor speed increase.

During sensing, the processor speed of the Imote2 is increased from 13 to 104MHz. This increase is expected to result in an increase in the temperature of the processor. The processor temperature change, as well as the temperature changes in the power management IC (PMIC) component, were measured with an IR probe to determine their surface temperature; the results are shown in Figure 5.35. Both components see a temperature increase of approximately 7°C in 5 minutes.

The QF4A512 also generates heat during sensing when it does the oversampling, analog-to-digital conversion and applies digital filters. The same IR probe was used to measure the surface temperature of the QFA512, the op-amp and the accelerometer on the SHM-A board during sensing over a 20 minute period. The results are shown in Figure 5.36. The QF4512 experiences 15°C increase in temperature. The op-amp, which is closer to the QF4A512 than the accelerometer is a few degrees warmer than the accelerometer during the test. Both the accelerometer and the op-amp increase in temperature by approximately 9°C. According to the data sheet, the resulting change in the mean value of the accelerometer out put could be as high as -5mg in the *x*-axis, -6.75 mg in the *y*-axis and -15.75 mg in the *z*-axis. These estimates are in the same range as the observed drifts.

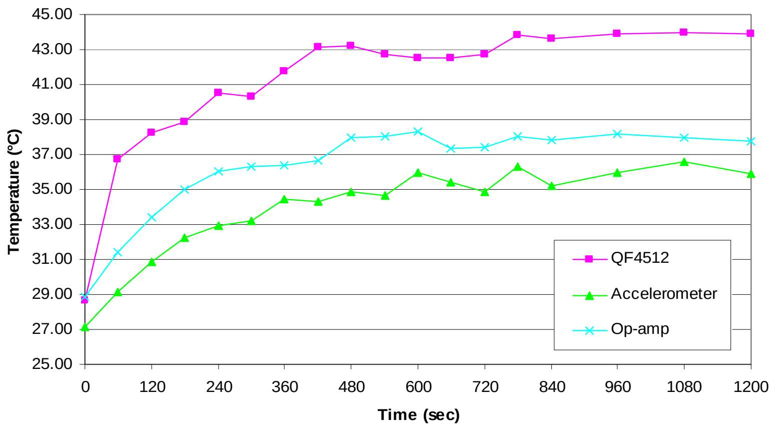


Figure 5.36 SHM-A component temperatures during sensing.

To help mitigate some of the self-heating temperature effects, the accelerometer was moved further away from heat-generating components in the final board layout. The evaluation of this change and further investigation of the temperature effects are addressed in the following section when the onboard temperature sensor is incorporated onto the SHM-A sensor board.

5.3.3 SHM-A multimetric sensor board

The third revision of the SHM-A sensor board was aimed at the addition of environmental sensors to the second revision to create multimetric sensor board for use in a broad range of SHM applications. A temperature sensor has the potential to lend additional insight into the observed behavior of a structure. Additionally, humidity measurements are of interest on structures that are susceptible to corrosion due to environmental factors. The anticipation of incorporating solar power options in the future also makes light measurements appealing. This board incorporates digital light, temperature, and humidity sensors to the existing SHM-A Rev 2 board layout. The sensors selected for this are as follows:

- *Light*: Texas Advanced Optoelectronic Solutions TSL2561 light-to-digital-converter which low active power and (~0.75 mW) and 16-bit Digital I2C Output
- *Temperature and Relative Humidity*: Sensiron SHT11 Humidity and Temperature with low power consumption (~30 mW), 14-bit temperature output and 12-bit humidity output.

Additional software updates were made to allow commands to remotely read single points of light, temperature and humidity as well as continuous temperature measurements when acceleration data is collected.

The fourth and final revision of the SHM-A sensor board added some additional features to minimize some of the observed temperature effects on the mean value of the acceleration output and to expose external access to the remaining fourth channel of the 4-channel Quickfilter QF4A512 signal conditioner. Additionally, minor changes to the scaling of the input to the ADC were also carried out to further improve the sensitivity/noise performance of the board.

As with Rev. 2, the accelerometer had to be updated once again due to the LIS3L02AL becoming obsolete and being replaced with the LIS344ALH (ST Micro 2009). This new accelerometer is ultra-compact but otherwise, according to the data sheet, expected to have equivalent performance. Also, a dedicated voltage regulator was incorporated to ensure that the accelerometer power supply is kept constant.

A block diagram of the resulting board is shown in Figure 5.37 and the board is pictured in Figure 5.38.

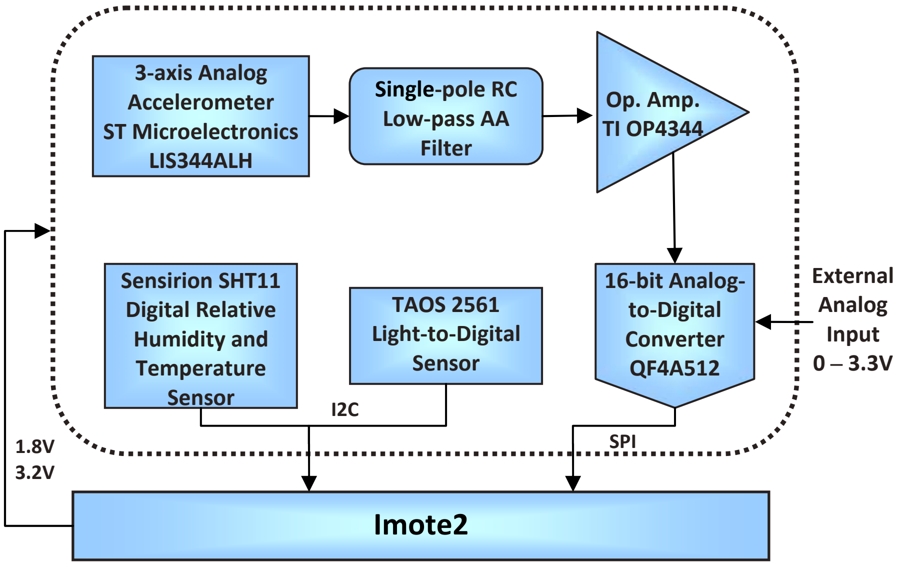

3-axis Analog Accelerometer ST Microelectronics LIS344ALH
Single-pole RC Low-pass AA Filter
Op. Amp. TI OP4344
Sensirion SHT11 Digital Relative Humidity and Temperature Sensor
TAOS 2561 Light-to-Digital Sensor
16-bit Analog-to-Digital Converter QF4A512
External Analog Input 0 – 3.3V
I2C
SPI
1.8V
3.2V
Imote2

Figure 5.37 Block diagram of SHM-A Rev 4.0

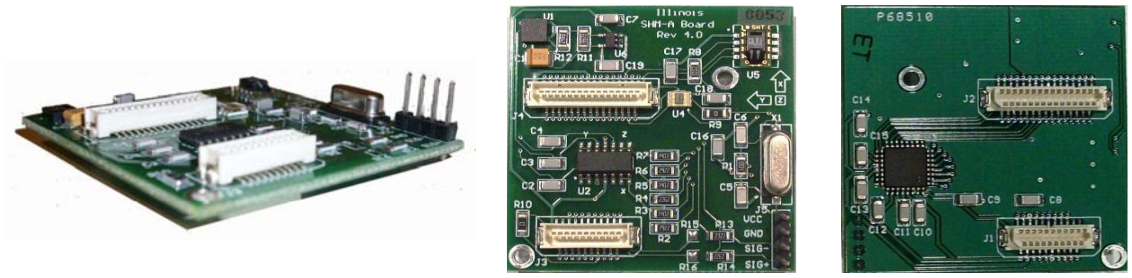


Figure 5.38 SHM-A sensor board: perspective (left), top (middle), and bottom (right).

According to the data sheet of the LIS344ALH accelerometers (ST. Micro), the expected noise floor of the *x* and *y* axes is in the range of 22 - 28 μg/√Hz while the *z*-axis is expected to have a noise floor in the rage of 30 – 60 μg/√Hz. Over a 20-Hz bandwidth, this corresponds to an *RMS* noise of 0.10 – 0.13 mg for the *x* and *y* axes and 0.13 – 0.26 mg for the *z*-axis.

One hundred SHM-A boards were tested to determine their noise performance and their calibration constants (scale and offset). The sensors were placed flat on a desk and data was collected at 50 Hz (with a cutoff frequency of 20 Hz) for 1.5 minutes with no external excitation. Additional tests were conducted with 8 sensor boards at a 1000-Hz sample rate with a cutoff frequency of 250 Hz to asses the higher frequency performance. To ensure that the desk was not vibrating, simultaneous measurements were taken with a low-noise seismic accelerometer (PCB 393C).

The average measured *RMS* vibration level of the level was 0.29 mg for the *x*- and *y*-axes and 0.67 mg for the *z*-axis. Although these values are higher than the values predicted by the accelerometer datasheet, they represent an improvement over the previous board revisions in the *x*- and *y*-axes. The higher noise levels in the *z*-axis appear to be intrinsic the most recent ST Micro accelerometer revision (LIS344ALH) which exhibits higher 1/*f* noise characteristics (higher noise at lower frequencies) along that axis; this noise was not observed in the previous accelerometers. The *x*- and *y*-axes should be used as the primary measurement axes when possible.

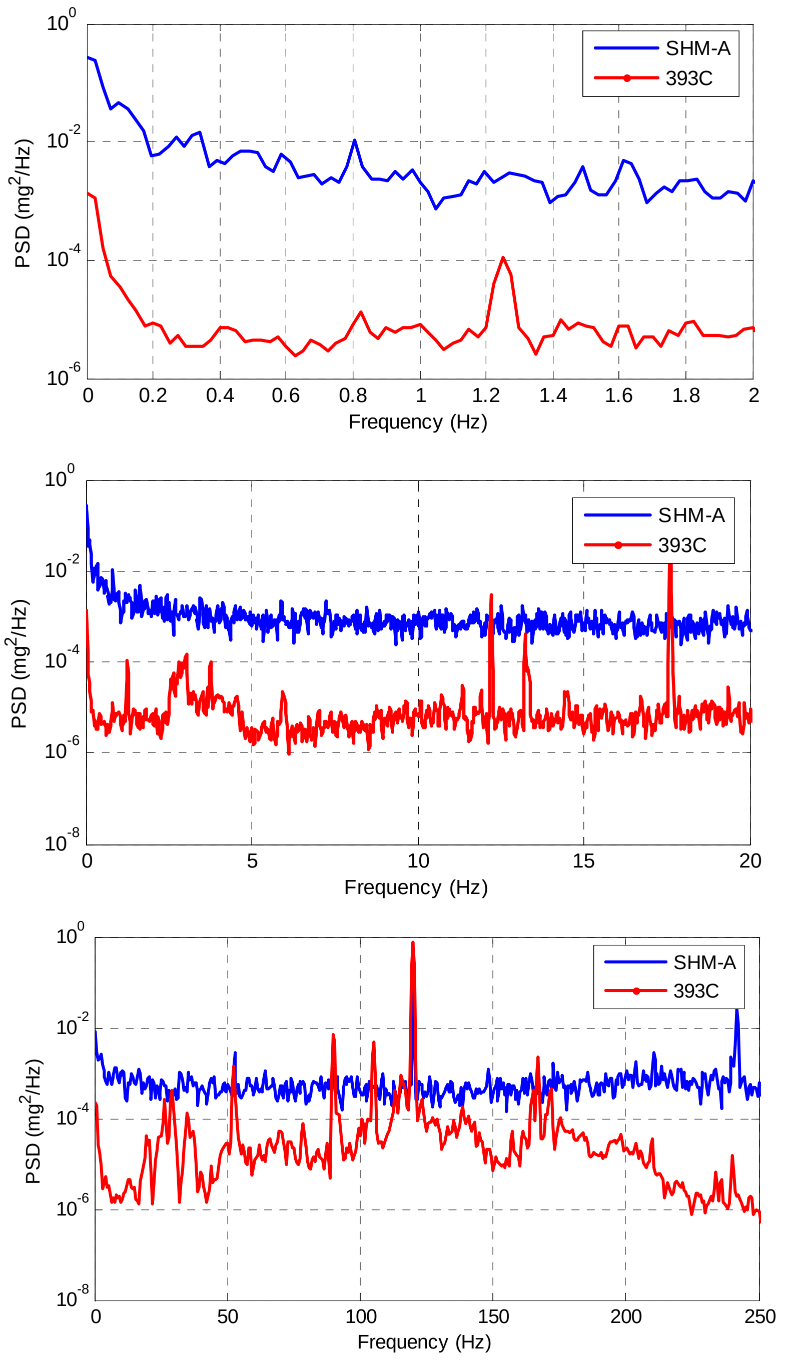


Figure 5.39 Example noise measurements of the *x* axis over a 20-Hz bandwidth [(a) and (b), zoomed to 2 Hz in (a)] and over a 250-Hz bandwidth (c).

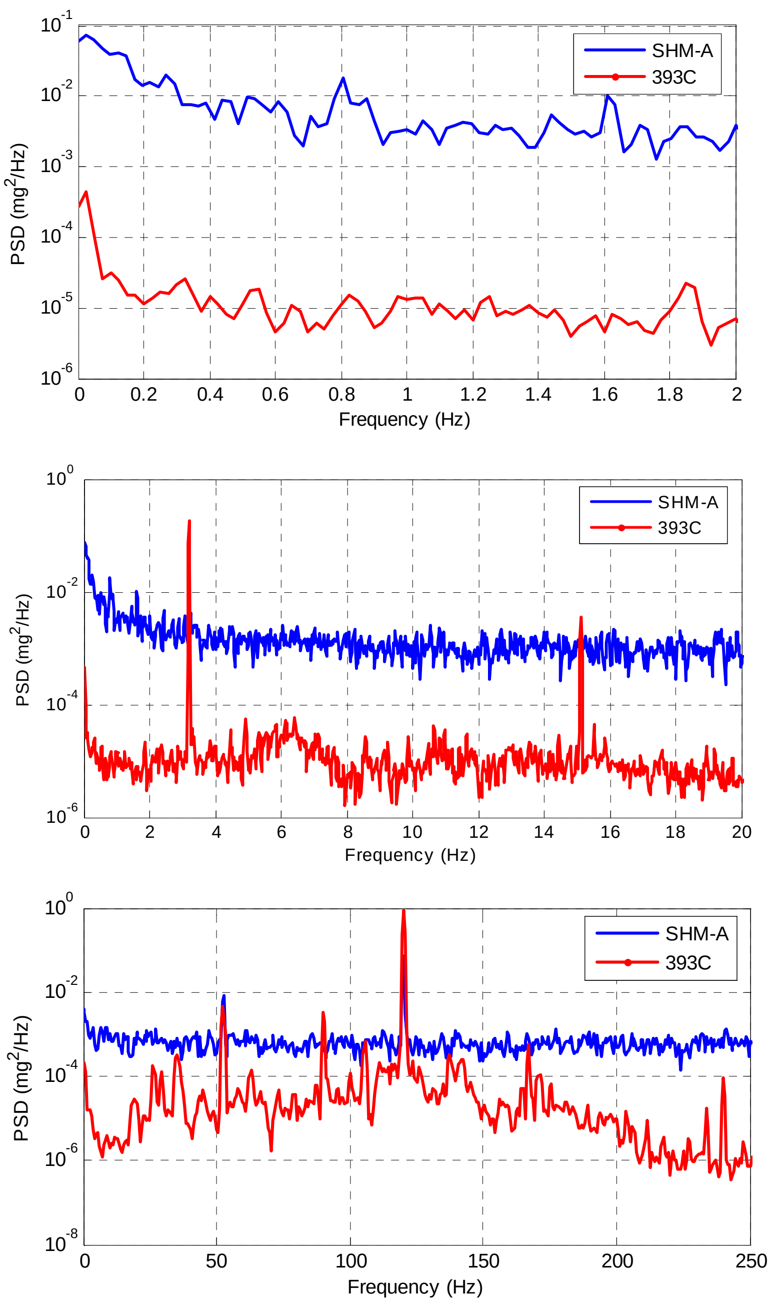


Figure 5.40 Example noise measurements of the *y* axis over a 20-Hz bandwidth [(a) and (b), zoomed to 2 Hz in (a)] and over a 250-Hz bandwidth (c).

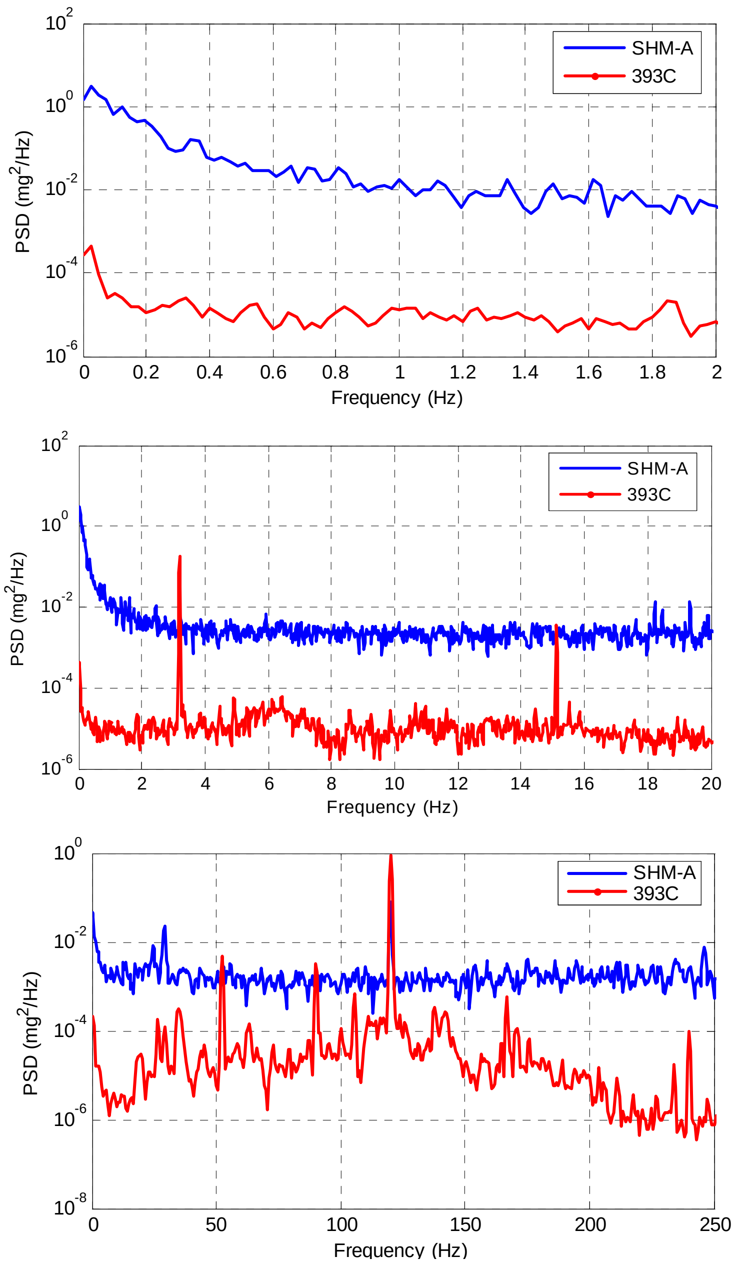


Figure 5.41 Example noise measurements of the *z* axis over a 20-Hz bandwidth [(a) and (b), zoomed to 2 Hz in (a)] and over a 250-Hz bandwidth (c).

The power consumption of the original sensor board was addressed in the previous section however with the hardware changes in the final revision, the power consumption of the SHM-A sensor board is revisited here. In addition to component changes, modifications were made to the operating and power management software to improve the power consumption. The theoretical power consumption of the final SHM-A sensor board may be determined from the data sheets of its various components.

Table 5.15 Specified power consumption of final SHM-A components.

Component	Active Current (mA)	Supply Voltage (V)	Power (mW)
LDO (LTI1761)	0.02	3.2	0.06
Accelerometer (LIS344ALH)	0.85	3.2	2.7
Op. Amp (OPA4344)	1.0	3.2	3.2
QF4A512 PGA	12.7	3.2	40.6
QF4A512 anti-aliasing filters, ADC, clocks	72.8	1.8	131.0
QF4A512 FIR filters	6.8	1.8	12.3
QF4A512 SPI operation	0.06	3.2	0.19
Light (TSL2561)	0.24	3.2	0.77
Temperature and humidity (SHT11)	0.94	3.2	3.0
Total Power			193.8

The actual power consumption of the SHM-A sensor board was determined by measuring the current draw from 3 D-cell batteries with a combined voltage of 4.5V while the sensor board was idle on the Imote2 (also idle) and during sensing (when the Imote2 is operating at 104MHz).

Table 5.16 Measured power consumption for the final SHM-A sensor board design.

Imote2 + SHM-A Board – Inactive		Imote2 + SHM-A Board – Sensing	
Current (mA)	Power (mW)	Current (mA)	Power (mW)
46	207	168	756

The relatively high power consumption of the SHM-A sensor board (primarily due to the QF4A512) must be considered in the overall power management of the sensor network. Chapter 6 gives more detail on how *SnoozeAlarm* works together with the sensing applications to minimize the overall power consumption despite the higher draw of the SHM-A sensor board.

The SHM-A Datasheet in Appendix A gives a detailed description of the electrical and mechanical characteristics and the typical performance of the Rev. 4.0 board.

5.3.4 Temperature compensation

To address the mean value drift of the accelerometer output resulting from temperature changes, onboard temperature compensation has been implemented in software. The availability of the onboard temperature sensor allows the temperature to be measured simultaneously with the acceleration. The result is that the direct relationship between the self heating of the board and the accelerometer can be determined. The accelerometer and the temperature sensor are located similar distances from the most heat-generating components (the QF4A512 and the Imote2 Processor as indicated in Figure 5.42) so the temperature sensor is expected to read temperatures very similar to those experienced by the accelerometer.

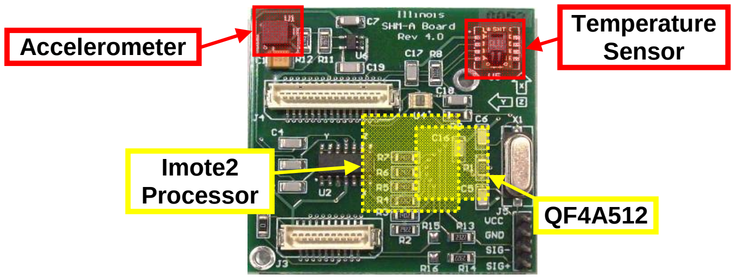


Figure 5.42 Location of components on or near the SHM-A sensor board.

Figure 5.43 (a) shows an example of the temperature measured on board the SHM-A sensor board over a 2-minute period. In this case, the temperature increases by 2°C. Figure 5.43 (b) shows the drift in the mean value of the *x*- and *y*-axes over the same time period. As the temperature increases, the mean value of the acceleration decreases. The correlation between the temperature and the zero-g drift (mean value) can be determined by plotting the mean value versus temperature and using linear regression to fit a line to the data as shown in Figure 5.43 (c). The slope of the line, α, is the zero-g offset drift as a function of temperature. In the example data shown in the figure below, the *x*-axis mean value temperature sensitivity, α_x, is 0.55 mg/°C and the *y*-axis mean value temperature sensitivity , α_y, is 0.43 mg/°C.

Two sets of temperature tests were conducted on SHM-A sensor boards to evaluate their performance in a variety of temperatures. The first set of tests was conducted on 8 sensor boards in an oven with temperatures ranging from 25 to 50°C. The second set of tests was conducted with two sensor boards in a cold setting at 0 and -20°C. The core voltage of the Imote2 was increased to 1.1V (from a default of 0.85V) for the cold temperature tests. In colder temperatures the conductivity of the semiconductor material within processor decreases. Increasing the core voltage helps overcome the decreased conductivity. The tests showed that the Imote2 and sensor board performed well at all temperatures.

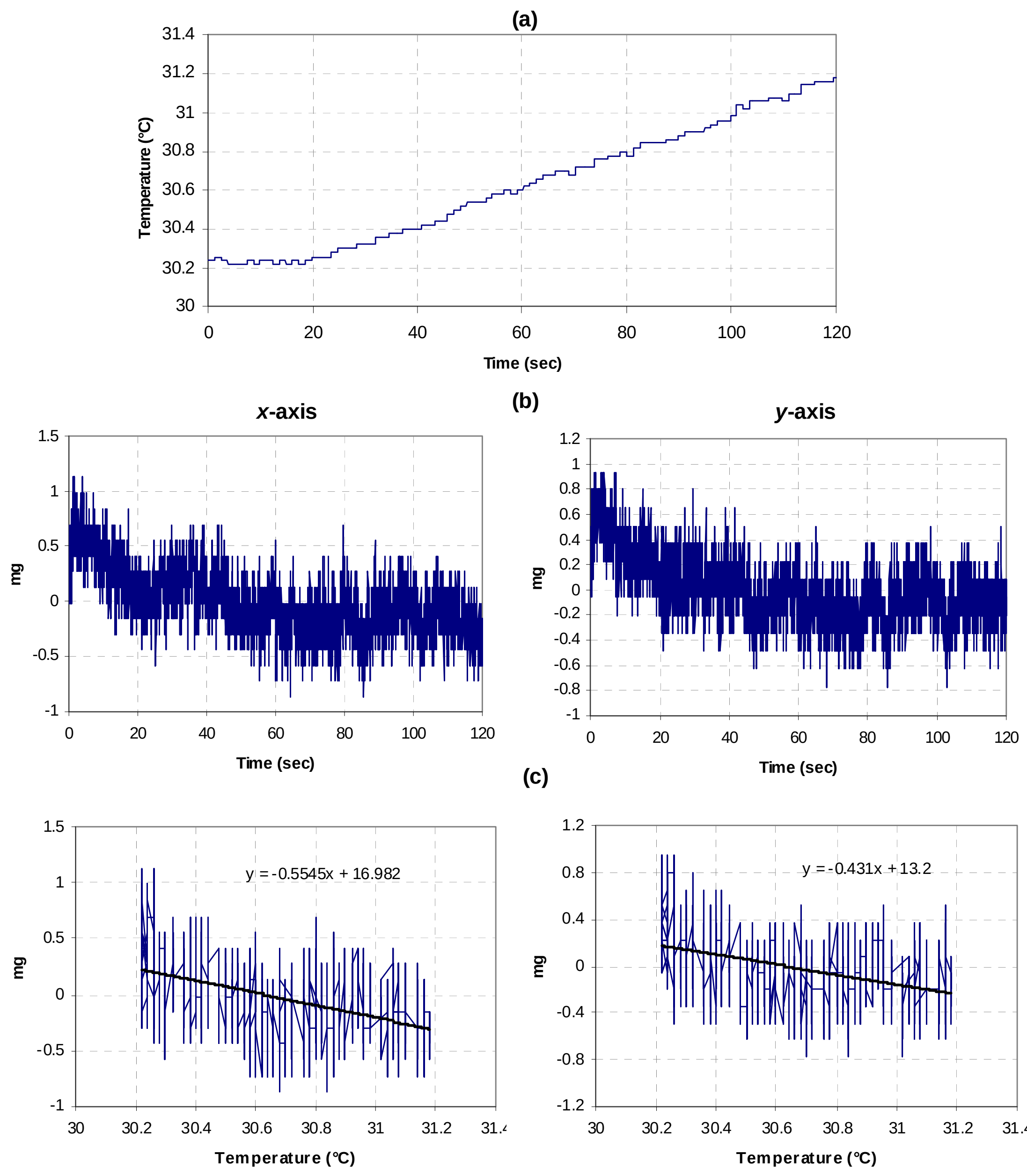


Figure 5.43 Measured temperature on SHM-A sensor board (a), uncorrected accelerometer readings (b) and acceleration vs. temperature plots (c).

At each temperature test point (every 5 degrees between 25 to 50°C) two relationships were determined: 1) the mean value temperature sensitivity, α, and 2) the amplitude sensitivity, β. The mean value temperature sensitivity, in mg/°C was determined by the linear relationship between the measured mean value of the acceleration measurement at zero g and the temperature during each test. The amplitude sensitivity, expressed as a percentage, was determined by the difference in the mean value temperature sensitivities measured at zero g and 1g divided by 1g as shown in Eq. 5.6 and illustrated in Figure 5.44. For the eight boards tested in the oven, the distribution of the results is shown in Figure 5.45, Figure 5.46, and Figure 5.47.

$$\beta = \frac{(\alpha_{1g} - \alpha_{0g})}{1g} \tag{5.6}$$

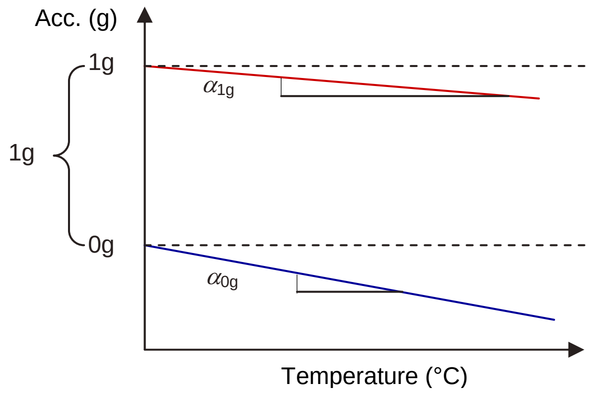


Figure 5.44 Acceleration amplitude sensitivity determination.

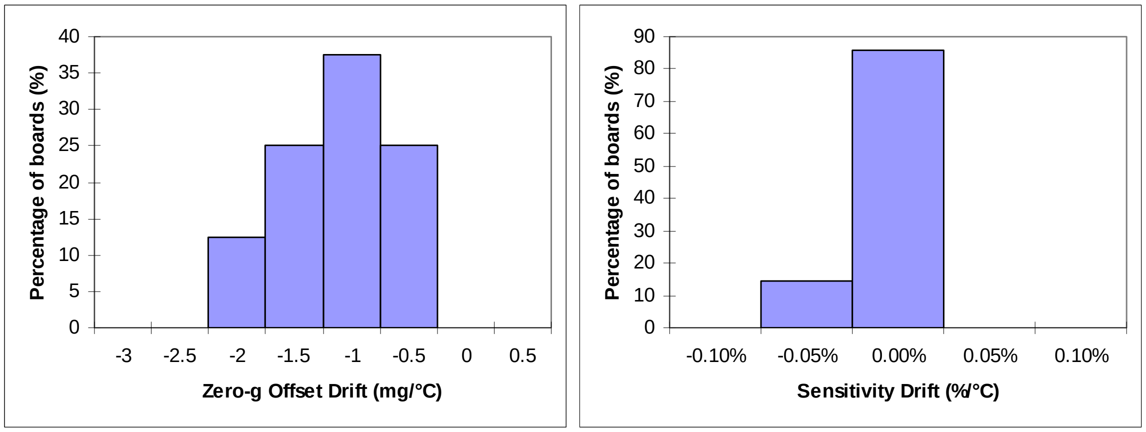


Figure 5.45 *x*-axis mean value and amplitude temperature sensitivity distributions.

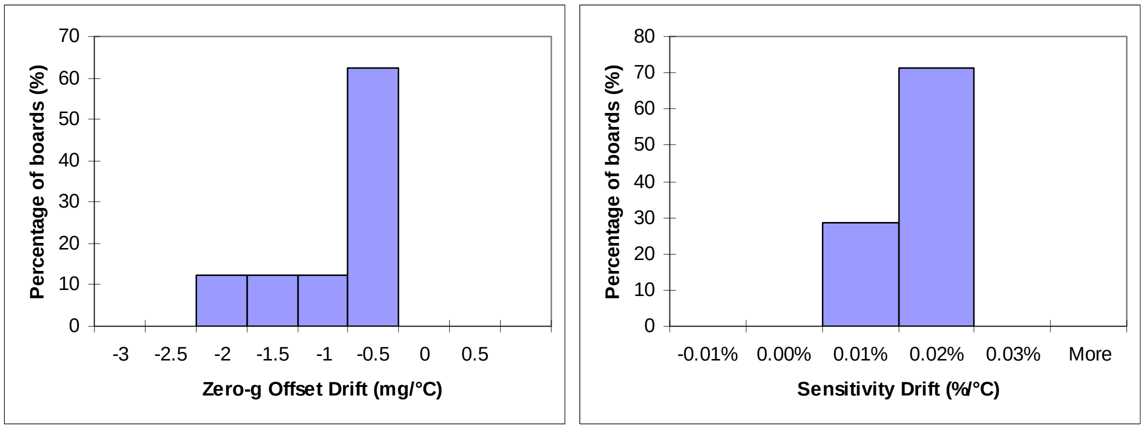
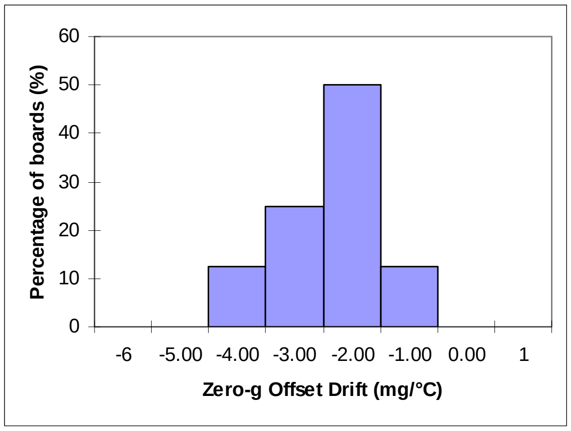


Figure 5.46 *y*-axis mean value and amplitude temperature sensitivity distributions.



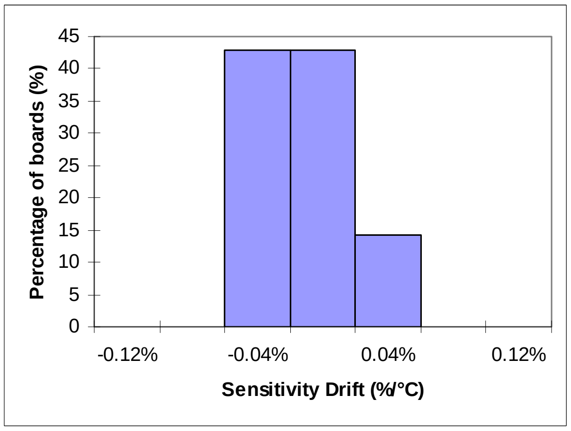


Figure 5.47 *z*-axis mean value and amplitude temperature sensitivity distributions.

The amplitude sensitivity drift is quite small – within ±0.04 percent for all axes on all boards – and is not expected to have a significant effect on the measured data. For example, in the worst case, if the temperature changes by 20°C, the amplitude sensitivity would change by 0.8 percent. At 30 mg, the resulting error introduced would be 0.24 mg, which is less than the resolution of the SHM-A sensor board. Due to its limited effect on the SHM-A measurements, the amplitude temperature sensitivity is neglected and not corrected with the measured temperature data. If larger temperature swings or higher vibration levels are expected, amplitude sensitivity correction could be incorporated into the software.

The measured mean value drifts are significant and must be corrected using the measured temperature data. To use the temperature measured on board the SHM-A sensor board to correct the acceleration data, there must be a two part approach: 1) determine the mean value temperature sensitivity of each channel on the sensor board and 2) implement onboard correction in software with the sensitivity coefficient. A utility, *SHMATempCal,* has been created to address the first part of the problem, and temperature correction functionality has been added to the sensor board driver for the SHM-A sensor board. The following paragraphs give the details of these approaches.

The *SHMATempCal* utility can be run on up to 40 remote nodes at one time, facilitating efficient calibration constant determination for each channel on many sensor boards. To run the calibration utility, the sensors should be placed on a still surface at room temperature. When the command is sent to the remote nodes, sensing is initiated on each of the three accelerometer channels. Once five minutes of acceleration and temperature data are collected, the linear relationship, or mean value temperature sensitivity, S_T, between the change in temperature and the change in the mean acceleration value is estimated according to the following equation:

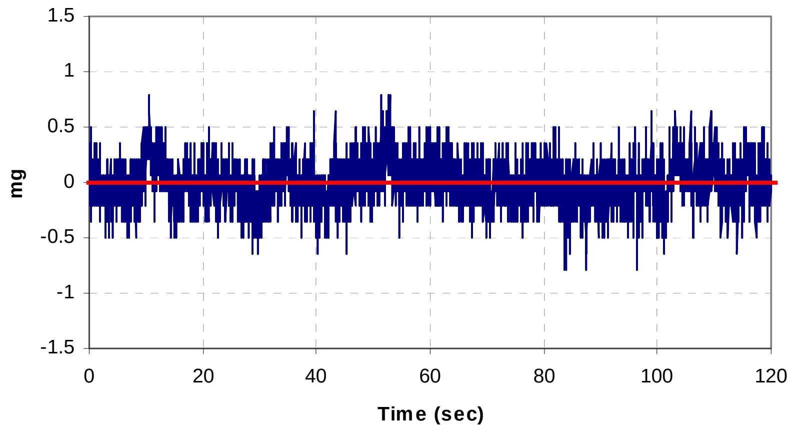
$$S_T = \frac{\sum_{i-1}^{n} \frac{a_i - a_{init}}{T_i - T_{init}}}{n} \tag{5.7}$$

where: n is the number of measured data points
a_i is the ith accelerometer measurement

a_{init} is the mean value of the first 10 data accelerometer data points
T_i is the ith temperature measurement
T_{init} is the mean value of the first 10 temperature data points

If the calibration utility is run more than once, the results may vary slightly from test to test due to inherent signal noise and temperature measurement error (up to ±0.4°C according to the SHT11 datasheet) so the sensitivity values returned by the utility are approximate.

Figure 5.48 shows the measured acceleration data after the mean value temperature sensitivity constants have been added to the software and the onboard temperature correction is integrated into the driver. These figures show that the mean value of the SHM-A sensor board output do not drift in time when the temperature correction is applied.



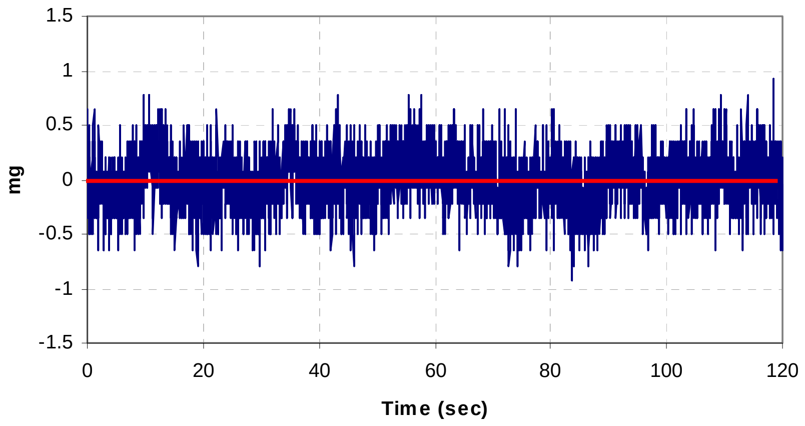


Figure 5.48 Scaled mean value measurements of *x*-axis (top) and *y*-axis (bottom) collected after onboard temperature correction.

5.3.5 Summary

This section has presented the design of a multimetric sensor board for SHM applications. This sensor board measures three axis of acceleration with a nominal noise floor of 0.3 mg in the *x*- and *y*-axes and 0.67 mg in the *z* axis (over 20Hz). There is an additional external analog input available (0 – 3.3V) as well as temperature, humidity, and light measurements. The acceleration mean value may be corrected for temperature drift using the onboard temperature measurements. The SHM-A sensor board has versatile signal processing capabilities and provides user-selectable anti-aliasing filters with variable cutoff frequencies, while maintaining high sampling rate accuracy. The SHM-A sensor board has been validated through a series of shake table and other calibration tests. The performance of the sensor board on full-scale structures can be found in Chapter 7.

IMPLEMENTATION CONSIDERATIONS

The design of a smart sensor network for autonomous, full-scale implementations requires the consideration of many critical issues that are not encountered in a laboratory or small-scale test bed. A large-scale implementation introduces challenges associated with communication quality/range that must be addressed, and power consumption becomes a critical issue for the success of a long-term deployment. This chapter addresses these implementation considerations, starting with a thorough assessment of the radio communication characteristics of the Imote2. The second section addresses the need for environmental enclosures for outdoor networks. The final section of the chapter illustrates how the power consumption and battery life of the network is affected by the design of the application and the choice of hardware. The results presented in this chapter are intended to provide guidelines on the design of a full-scale, autonomous network of Imote2 smart sensors.

6.1 Radio communication evaluation

Successful WSSN implementations rely heavily on achieving effective communication within the network however the wireless communication hardware typically used by smart sensing platforms in SHM applications has had limited experimental characterization. This section provides experimental characterization of smart sensor wireless communication hardware by tailoring the analysis towards the end user, civil engineers, and researchers, with a focus on SHM applications. Three key elements of wireless transmission are addressed within the framework of the Imote2 smart sensor platform: (1) a qualitative understanding of wireless communication and packet delivery, (2) a quantitative characterization of the performance of the Imote2's onboard antenna and the performance of an optional external antenna performance, and (3) the impact of these factors on full-scale implementation.

6.1.1 Communication hardware

The radio chip used on the Imote2 is the Chipcon CC2420 2.4 GHz IEEE 802.15.4 RF transmitter. The chip is a byte-level radio ideal for low-voltage, low-power wireless applications. The radio supports multiple transmission options that can be tailored to the application to optimize network performance. This study explores the effects of varying two of the primary transmission options: (1) transmission channel (or frequency) and (2) transmission power. The selection of the appropriate transmission frequency is important when other wireless devices operating in the 2.4GHz frequency band are within range of the sensor network. The selection of the optimal transmission power is critical to power management. Higher transmission power allows the sensors to achieve longer communication distances but results in higher current consumption on the sensor node. Limiting the current consumption will reduce the amount of battery power used by RF communication over the life of the network.

The CC2420 transmission power output ranges from -25 dBm to 0 dBm and the corresponding power level and current consumption are shown in Table 6.17. In receiver mode, the typical current consumption is 19.7 mA.

Table 6.17 Output power options and corresponding current consumption (Chipcon 2004).

Power Command	Output Power (dBm)	Current Consumption (mA)
31	0	17.4
27	-1	16.5
23	-3	15.2
19	-5	13.9
15	-7	12.5
11	-10	11.2
7	-15	9.9
3	-25	8.5

The onboard antenna included on the Imote2 is the Antenova Mica 2.4 GHz SMD pictured in Figure 6.49. The antenna is designed to use the board to which it is mounted as a ground plane; thus, the entire board acts as the antenna. The antenna offers a peak gain of 1.8 dBi (dBi is a measure of the gain of the antenna with respect to a hypothetical isotropic antenna with a 0 dB gain). A more thorough discussion of the antenna behavior with the Imote2 will be given in a subsequent section.

The optional external antenna used selected for use with the Imote2 in this study is the Antenova Titanis 2.4 GHz Swivel SMA antenna pictured in Figure 6.49. The half-wave dipole antenna has a peak gain of 2.2 dBi. One advantage is that the blade of the antenna can rotate 360° for optimal antenna orientation.

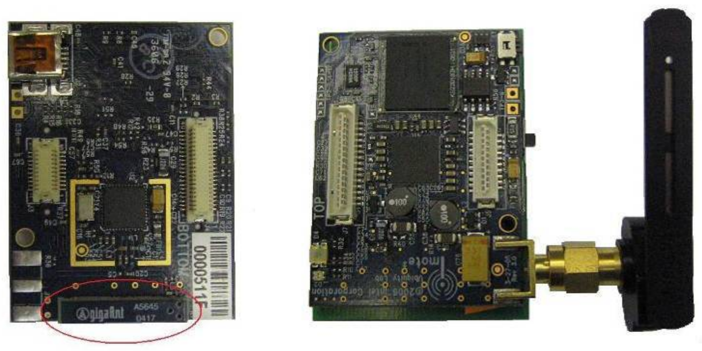


Figure 6.49 Imote2 with onboard antenna (left) and external antenna (right).

6.1.2 Communication hardware-software interface

Within the embedded software, the data to be communicated and its routing information are placed in packets. Communication on the Imote2 implements a fixed payload scheme where the number of bytes in a packet does not change. The packet size is 28 bytes, consisting of a 4-byte header 24 bytes for data. Numerous packets are required to send long data records.

Prior to transmission, the radio chip adds a preamble and cyclic redundancy check (CRC) to each packet as shown in Figure 6.50. CRCs characterize the packet and are used for error detection after transmission. The radio packet is then separated and transmitted byte-by-byte.

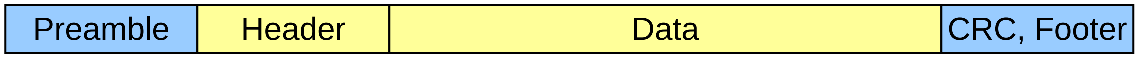


Figure 6.50 Radio packet.

Similarly, in receiving mode, the radio chip receives the transmitted data byte-by-byte, which it places into the radio packet based on the preamble. Upon successful packet reception, TinyOS processes the packet to determine whether it should be retained or dropped, based on whether any error exists. If the CRC does not correspond with the received packet then error has occurred and the packet is dropped.

6.1.3 Radiation pattern characterization

While the hardware data sheets provide ideal performance characteristics, the as-built system requires careful evaluation to assess its actual performance. Anechoic chamber tests were conducted in the wireless wind tunnel on the University of Illinois campus to determine the radiation pattern of the as-built sensor and get a better understanding of the antenna performance. Three configurations were considered: (1) onboard antenna, (2) external antenna, and (3) external antenna, in conjunction with the environmentally hardened enclosure as shown below.

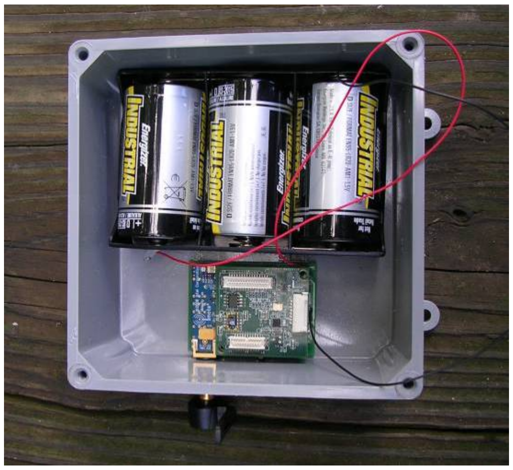


Figure 6.51 PVC Imote2 enclosure.

Power radiated by an antenna varies as a function of the directional coordinates, and decreases inversely with increasing radial distance, *d*, from the antenna. Radiation patterns are used to plot the variation in transmission power with angular position. While a complete description of the antenna pattern requires a three-dimensional plot, it is common to use a few two-dimensional polar plots to convey the most important information, especially if the antenna radiation has a symmetric pattern. The anechoic chamber uses a linearly polarized standard gain horn to measure the electric field transmitted by the antenna.

The polarization of an antenna is the direction of the electric field vector of the radiated wave. The antennas in this study are supposed to be linearly polarized, which ideally means that the electric field is only in one direction. However, antennas never radiate pure linearly polarized fields; hence, fields are generally measured for two perpendicular orientations of the horn to record the complete field (Chang 2002). The electric field component in the desired orientation is referred to as the *co-polarized* field and any field in the perpendicular (undesired) direction is referred to as the *cross-polarized* field. Generally antennas with high polarization purity are desired, *i.e.*, the co-polarized fields are considerably stronger than the cross-polarized fields.

For the first two test configurations, the antennas were run and powered by the Imote2 and its corresponding battery board to assess the as-built performance. The node, powered by three AAA batteries, was mounted in a plastic support as shown in Figure 6.52. A *ContinuousSend* program, which continuously transmits dummy packets, was used for transmission, and power measurements were taken at 10 degree intervals with an Aglient E4440A spectrum analyzer. Co-polarized and cross-polarized field measurements were taken in the azimuthal plane and used to calculate power levels.

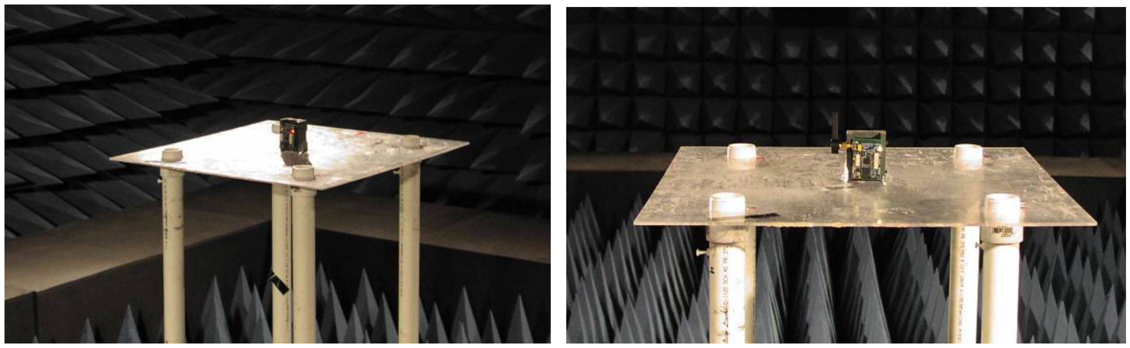


Figure 6.52 Onboard and external antenna set-up in anechoic chamber.

In the third test configuration, the Imote2 was mounted in the environmentally hardened enclosure as shown in Figure 6.51 to determine the impact of the enclosure on antenna performance. The *ContinuousSend* program was used for transmission and power measurements were taken at 10 degree intervals with the spectrum analyzer for comparison with the external antenna measurements.

The power patterns for the co-polarized and cross-polarized electric fields for the antenna tests in the anechoic chamber are given in Figure 6.53 and Figure 6.54.

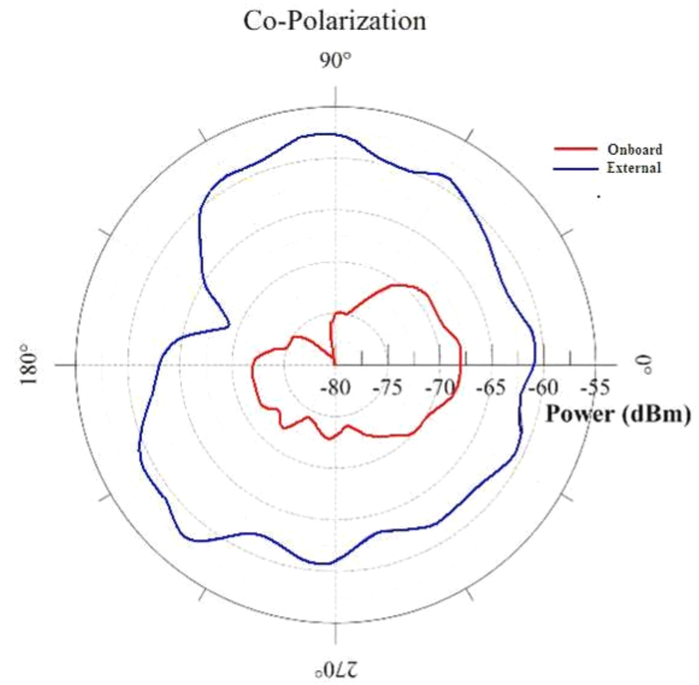


Figure 6.53 Co-polarization plots of onboard and external antennas.

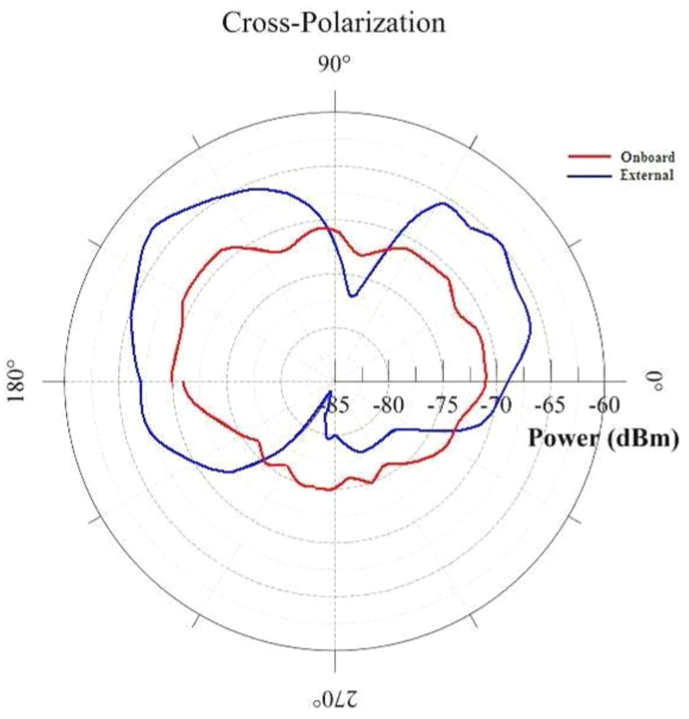


Figure 6.54 Cross-polarization plots of onboard and external antennas.

The onboard antenna does not exhibit typical dipole antenna characteristics (azimuthal symmetry). The co-polarized fields exhibit a slight preference for 0°; however, both the co-polarization and cross-polarization measurements show a relatively uniform power level of -70 dBm. Additionally, elevation plane pattern measurements were taken that also showed a fairly uniform gain. Thus, this antenna exhibits primarily isotropic behavior and has no ideal orientation for communication. Nonetheless, aligning two sensors along the 0° direction would be the best orientation for communication if the onboard antenna was used as it shows a higher directivity.

The external antenna exhibits more typical dipole antenna characteristics as illustrated in Figure 6.54. The co-polarized fields show a relatively uniform power, which is about 10 dBm above (or approximately ten times) the cross-polarization power. In the elevation plane patterns (not included here), the expected dipole antenna butterfly pattern was seen.

The external antenna and the enclosure resulted in the same pattern as the external antenna on its own; however, the gain was about 5 dBm better. Because the tests were conducted on different days, these gain measurements may not be directly comparable due environmental changes potentially impacting the communication performance. Thus, the similarity in pattern characteristics is more significant and demonstrates that the enclosure does not negatively affect the external antenna's performance.

In their ideal orientations, the external antenna power is about 7 dBm better than the onboard antenna, which is approximately five times more powerful in linear units. Since power decays as a function of $1/d$ for large distances, where d is distance, use of the external antenna should result in at least twice the communication range.

6.1.4 Ideal communication range

\The ideal communication range of the as-built wireless sensor unit indicates the optimal transmission range of the Imote2 in the absence of environmental factors that reduce the quality of the transmission signal. While the ideal communication ranges are not expected to be achieved in an implemented wireless sensor network, they do provide a baseline for comparison of various power, frequency, and antenna configurations.

To assess the packet loss associated with single-hop communication, the testing protocol used loopback tests along with variations in (1) power levels, (2) communication channel, (3) sensor and antenna orientation, and (4) environmental factors. Loopback tests, illustrated in Figure 6.55, are used to verify the communication effectiveness under varying circumstances. A loopback test consists of sending a set number of packets from the sender node to a remote node. The remote node records the number of packets it receives and sends this information along with all of its received packets back to the sender. Finally, the sender records the number of packets it receives from the remote node. The results of the loopback test are: (1) the number and percentage of packets that made it to the remote node, and (2) the number and percentage of packets that made the complete round-trip back to the fixed sender. For each set of test parameters, the loopback test was repeated at least five times to obtain an average packet reception rate. This testing method may also be used to verify failing nodes within a network.

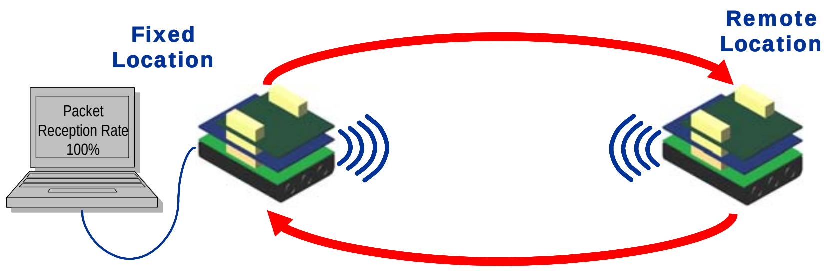


Figure 6.55 Loopback test set-up.

The loopback tests were run on the Imote2 using the *TestRadio* test application. Figure 6.56 shows a flow chart of how *TestRadio* is used to perform the loopback tests. At the start of the test, the user specifies the transmission channel, transmission power, the ID of the remote node(s) (up to 10 remote nodes may be tested at one time) and the number of packets to send. All of the command packets used to perform the tests are sent using a reliable communication protocol, while the packets sent to test the communication performance are sent only once. This approach ensures that the command packets are received, and the test application can run to completion even when there is poor communication performance indicated by the results achieved in the loopback test. When the communication distance becomes too great, even the command packets being sent reliably will not reach their intended target, resulting in a test failure.

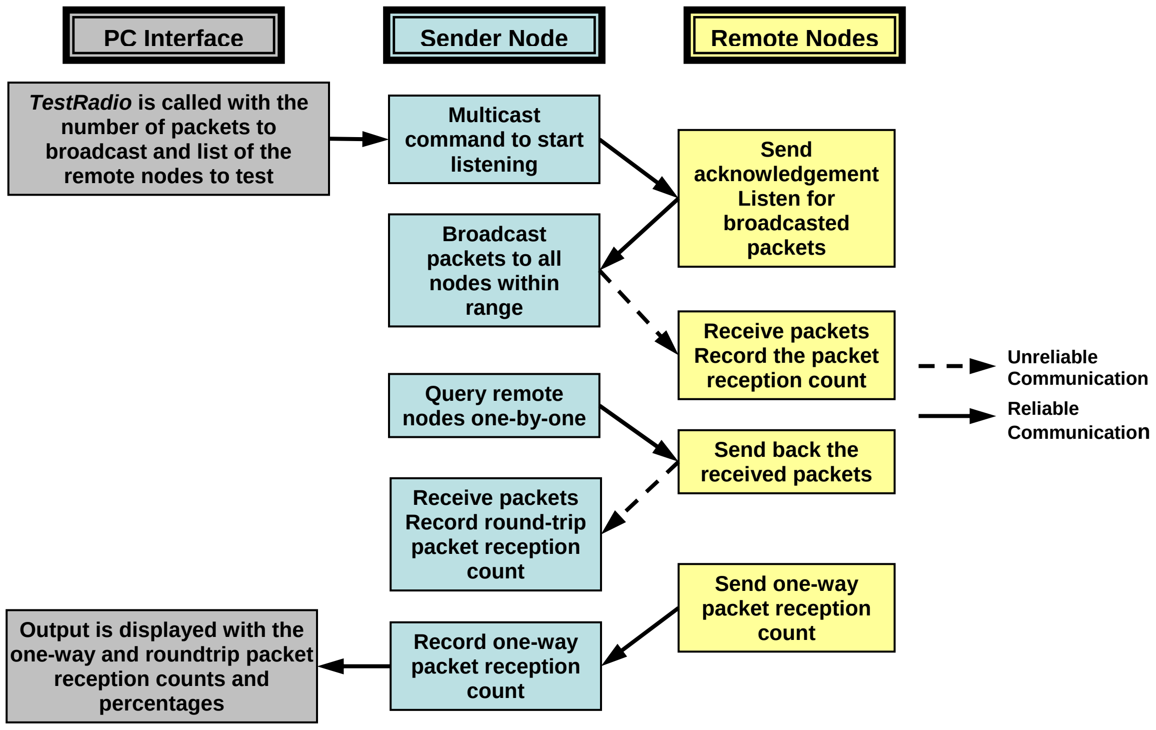

PC Interface
Sender Node
Remote Nodes
TestRadio is called with the number of packets to broadcast and list of the remote nodes to test
Multicast command to start listening
Send acknowledgement Listen for broadcasted packets
Broadcast packets to all nodes within range
Receive packets Record the packet reception count
Query remote nodes one-by-one
Send back the received packets
Receive packets Record round-trip packet reception count
Send one-way packet reception count
Record one-way packet reception count
Output is displayed with the one-way and roundtrip packet reception counts and percentages
Unreliable Communication
Reliable Communication

Figure 6.56 Loopback test implementation with the *TestRadio* application.

Beyond the testing program, the antenna orientation and environmental setting can be adjusted as desired. Two variables were kept constant throughout all tests: 1) the sensors were kept at a constant height above the ground of 4 feet, and 2) 1000 packets were sent between nodes.

The quantitative measurement of packet delivery performance used in these tests was the packet reception rate. This value refers to number of packets that were received out of the number of packets that were sent. The complementary measurement of packet reception rate is packet loss, which is the number of packets lost out of the number of packets that were sent.

For the ideal communication range tests, the influence of environmental factors was kept to a minimum. The tests were conducted outdoors with unobstructed line-of-sight (LOS) between sensor nodes with no other 2.4 GHz wireless networks present. Figure 6.57 illustrates the testing conditions. Using the radiation pattern characteristics determined from the anechoic chamber tests, the optimal antenna orientation was used for both the onboard and external antenna communication range tests. In this set of experiments, two remote nodes were used with one base node in order to test for variation in performance among nodes. The two remote nodes were placed 6 feet apart to reduce the possibility of mutual interference.

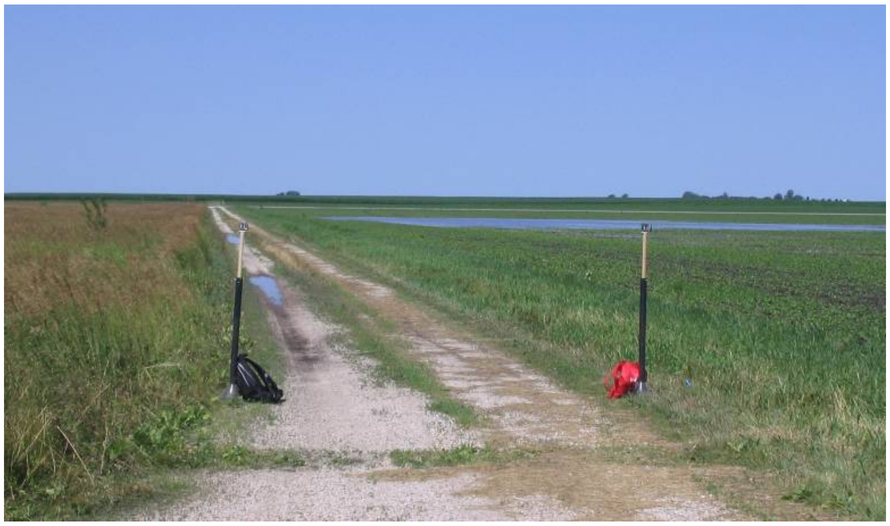


Figure 6.57 Outdoor testing environment.

The results for the outdoor loopback tests are shown in Figure 6.58 through Figure 6.61. For each plot, the largest distance given is the distance just before the tests could no longer be completed and corresponds to the maximum communication range of that configuration.

The external antenna performance is significantly better than the onboard antenna, as was expected. The external antenna outperformed the onboard antenna in both distance and consistency. At the same power level, the external antenna reached 1200 feet with a 92-percent packet reception rate, which is three times the distance of the onboard antenna, as expected from the anechoic chamber tests. Furthermore, the onboard antenna exhibited somewhat inconsistent communication ranges as the reception rate fluctuates significantly over short distances. In contrast, the external antenna exhibited more consistent behavior, with the reception rate at almost 100 percent until final drop-off.

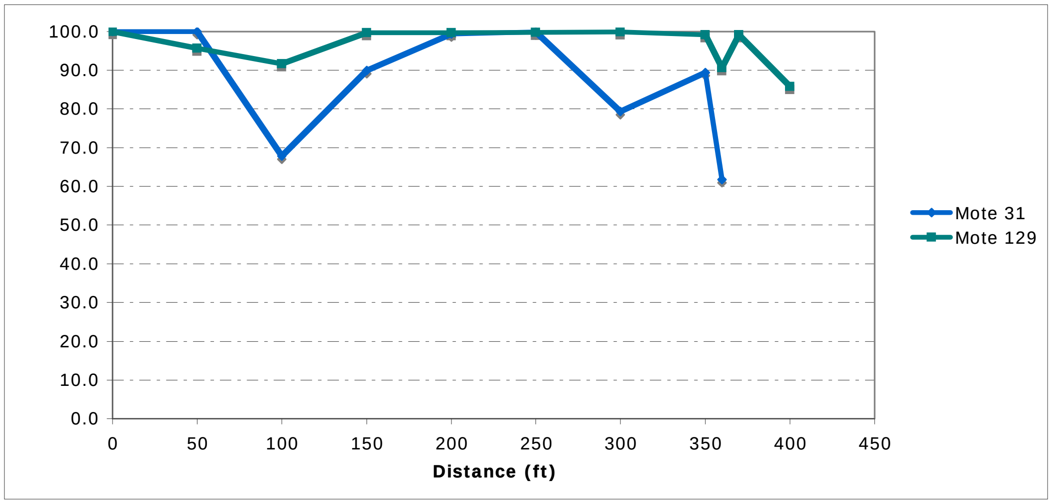


Figure 6.58 Onboard antenna performance at Power Level 31.

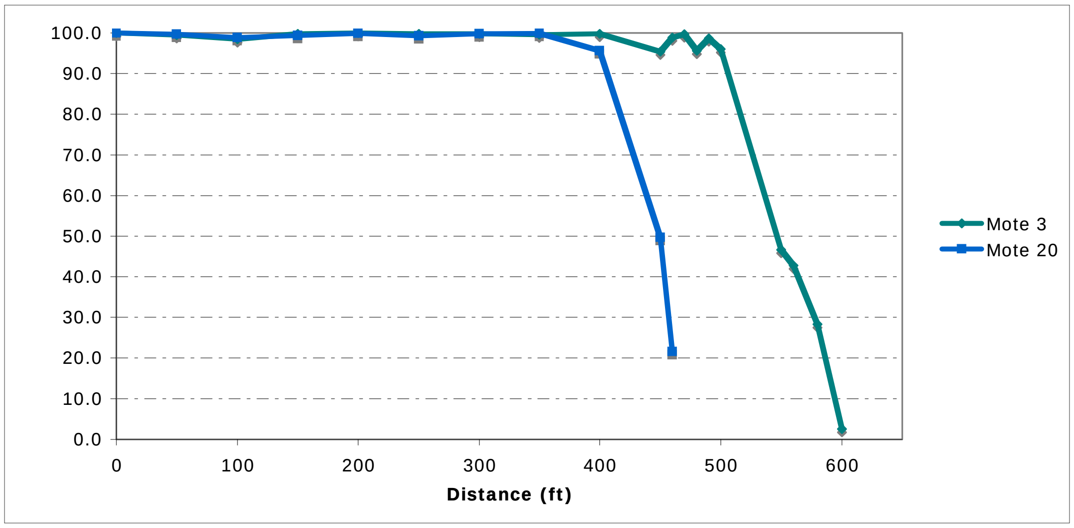


Figure 6.59 External antenna performance at Power Level 5.

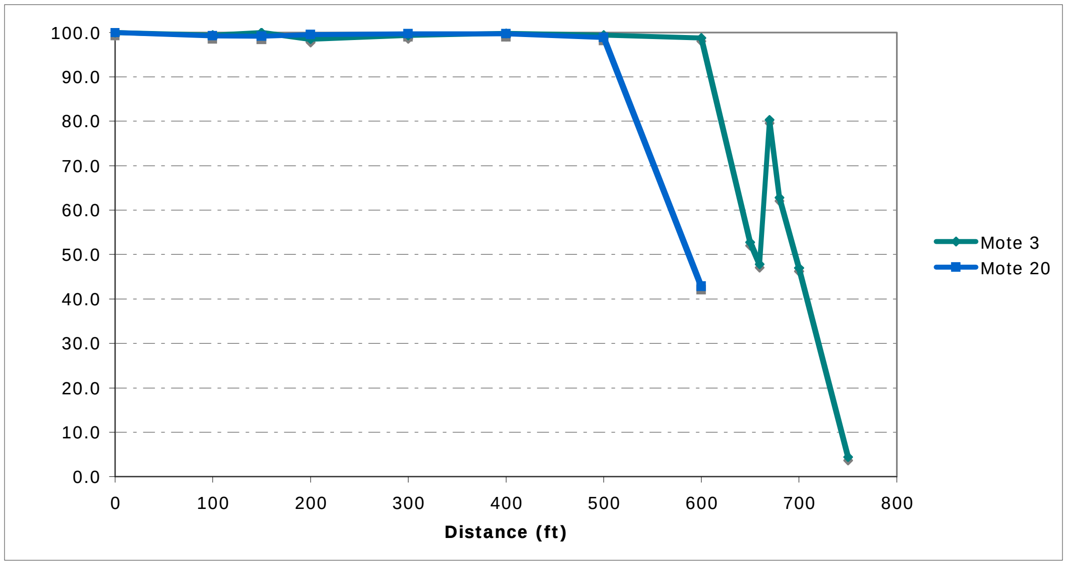


Figure 6.60 External antenna performance at Power Level 10.

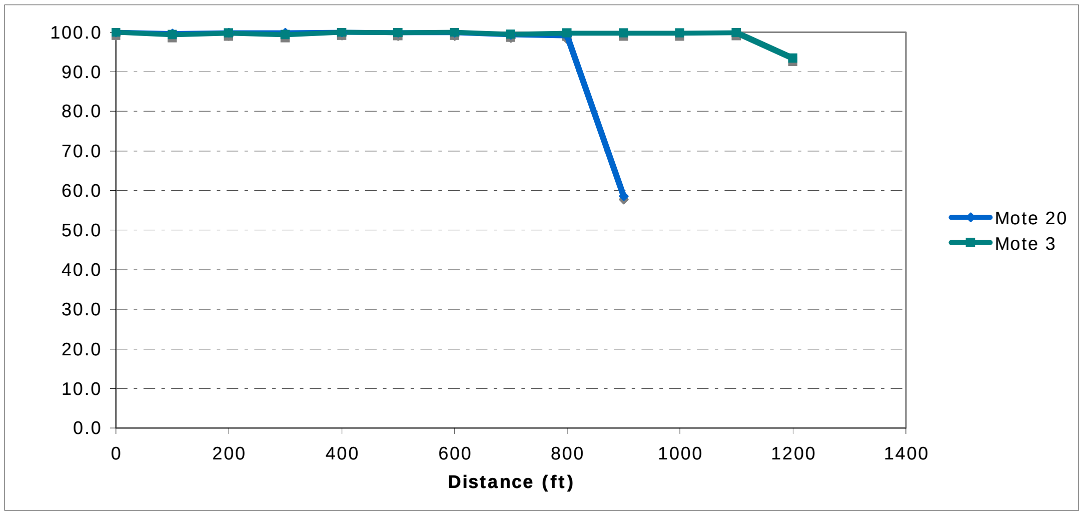


Figure 6.61 External antenna performance at Power Level 31.

For both antenna configurations and all power levels, visible variation is observed in the performance for different remote nodes. In these tests, one node consistently underperforms the other in communication range; however, they exhibit similar reception rate behavior. This variation in performance is expected due to variability in off-the-shelf components. Based on the shorter of the two communication ranges, an estimated usable communication range for optimal antenna orientation is established and given in Table 6.18.

Table 6.18 Ideal communication range for optimal antenna orientation.

Antenna/Power Configuration	Estimated Ideal Communication Range (ft)
Onboard *Power 31*	350
External *Power 5*	400
External *Power 10*	500
External *Power 31*	800

6.1.5 Influence of environmental factors

While the ideal communication range tests indicate the optimal performance of the Imote2 communication hardware, these conditions will seldom exist with sensor deployment on a civil engineering structure. Thus, the influence of common building environments and materials will be explored in this section. The impact of wireless internet networks, steel and concrete structures, and CMU infill will be specifically addressed. Since the ultimate influence of the built environment is of primary interest with respect to network implementation, the quantitative measure of the impact will be the reduction in the baseline reception rate that was observed in the same testing environment. Table 6.19 at the end of the section combines the results from all tests.

Wi-Fi network

Wireless sensors using the IEEE 802.15.4 protocol operate over the same 2.4 GHz band as wireless LAN (WLAN) devices using IEEE 802.11b,g protocols (Wi-Fi). Figure 6.62 illustrates this wireless channel overlap. Given this situation, each device can experience interference from the transmissions on the other network (Shin et al. 2007). Ultimately, this mutual interference can degrade performance and lead to packet loss. IEEE 802.15.4 standards recommend using the clear channels where the energy of Wi-Fi network is typically lower (Hubler 2005); however, if many IEEE 802.15.4 devices are operating in the area, these clear channels may not be used by other devices leaving insufficient bandwidth to ensure that interference is avoided.

Given the possible interference of wireless networks on sensor communication, the impact of WLAN on communication reliability was evaluated with loopback tests in Newmark Civil Engineering Laboratory on the University of Illinois Urbana-Champaign campus. The tests were conducted in an open portion of the crane bay. The wireless network in the building operates on channel 11, which corresponds to channel 21 through 24 in the 802.15.4 spectrum.

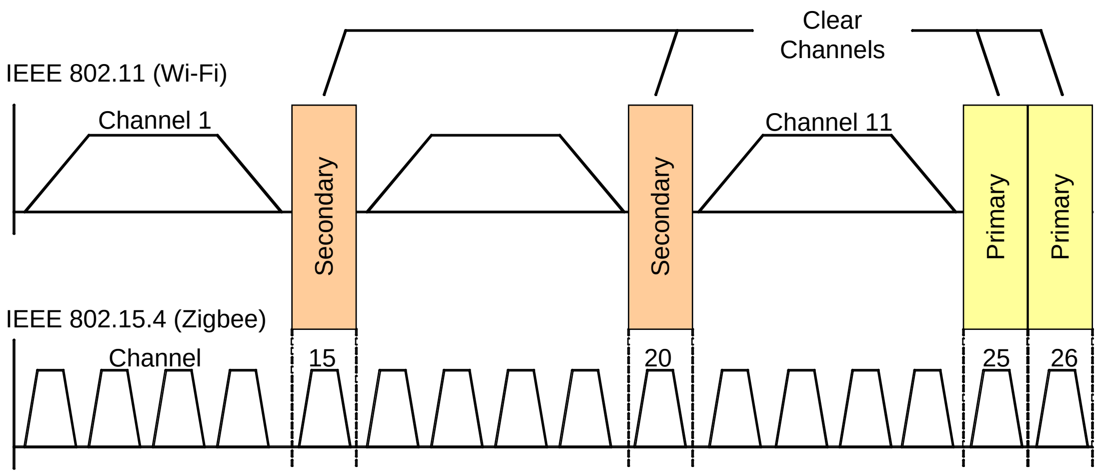


Figure 6.62 Wireless channel overlap for 802.11 network and 802.15.4 spectrum (Hubler 2005).

Two sets of tests were conducted with nodes equipped with external antennas in the optimal orientation. They were placed 25 feet apart and the radios were operated at full power (Power Level 31). In the first, the loopback tests were conducted on 802.15.4 channel 11, which falls within 802.11b, channel 1 and therefore outside of the operating frequency of the Wi-Fi network in the building. In the second, the loopback tests were conducted within the wireless network channel (802.11b channel 11) operating in the building. The screenshots (Figure 6.63 and Figure 6.64) produced using WiSpy 2.4i by MetaGeek (2008) illustrate the two test configurations. WiSpy is a 2.4GHz spectrum analyzer that measures the amplitude of signals over the 802.11 frequency bandwidth. The plots show transmission power vs. frequency. The numbers along the *x*-axis correspond to 802.11b channels.

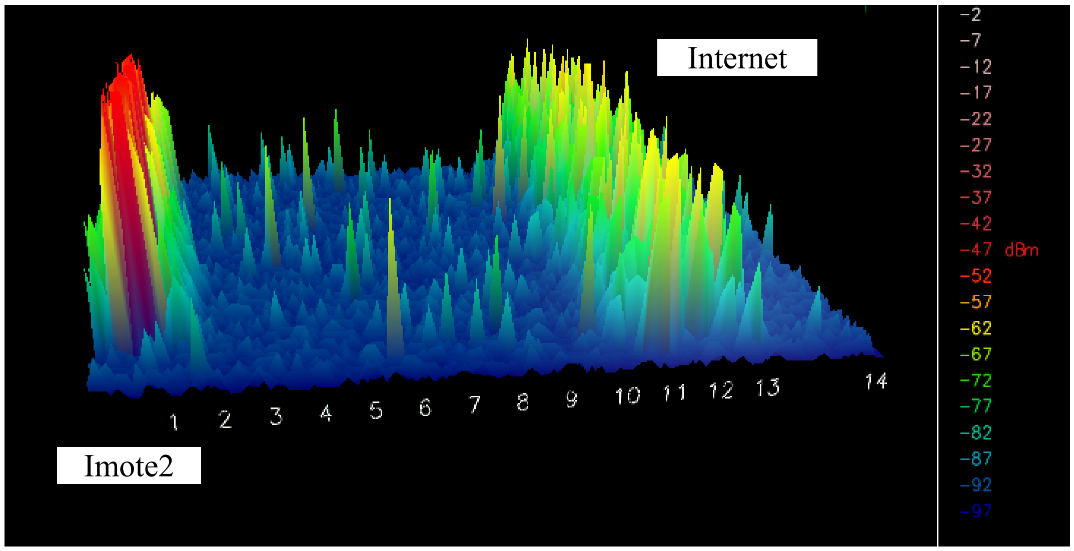


Figure 6.63 Communication outside of wireless internet channel.

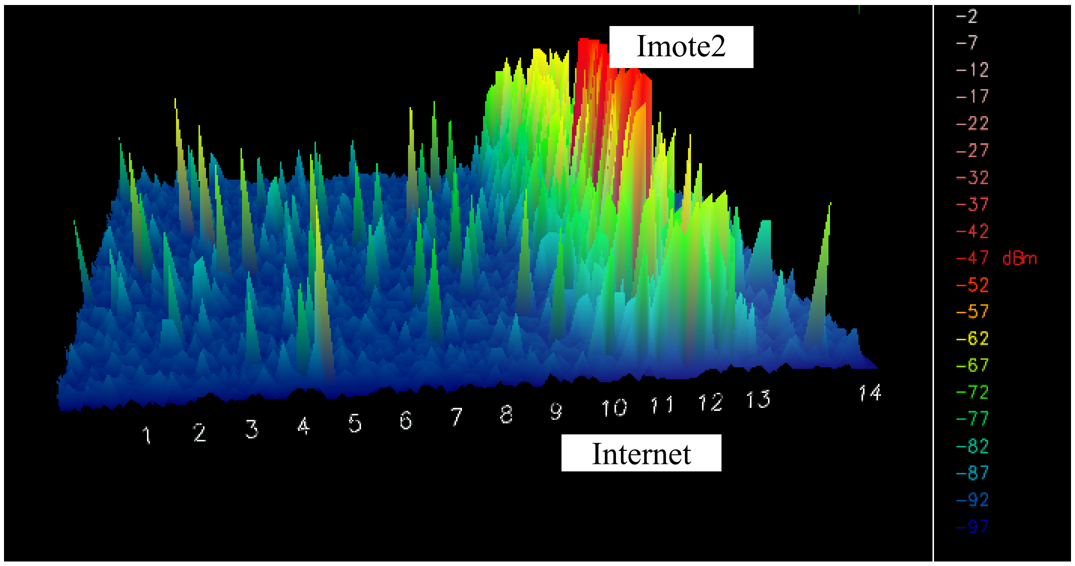


Figure 6.64 Communication within the wireless network channel.

Communication within the wireless channel resulted in a 15% reduction in reception rate as compared to communication outside the wireless channel. While the decrease is not drastic, it has the potential to affect the overall sensor network performance and power consumption. With some knowledge of other networks in proximity that are operating in the 2.4 GHz bandwidth, this interference can be avoided by careful operating channel selection.

Building materials

Unlike wireless interference, signal attenuation caused by building materials is unavoidable and can be significant. Three main building types and materials were considered: steel structures, reinforced concrete structures, and concrete masonry infill.

A steel building with poured concrete floors and typical finishing was used as a representative test of the expected signal attenuation to occur in a steel structure. A set of loopback tests were conducted in the Siebel Center on the University of Illinois at Urbana-Champaign campus using two nodes with external antennas. The testing environment is pictured in Figure 6.65.

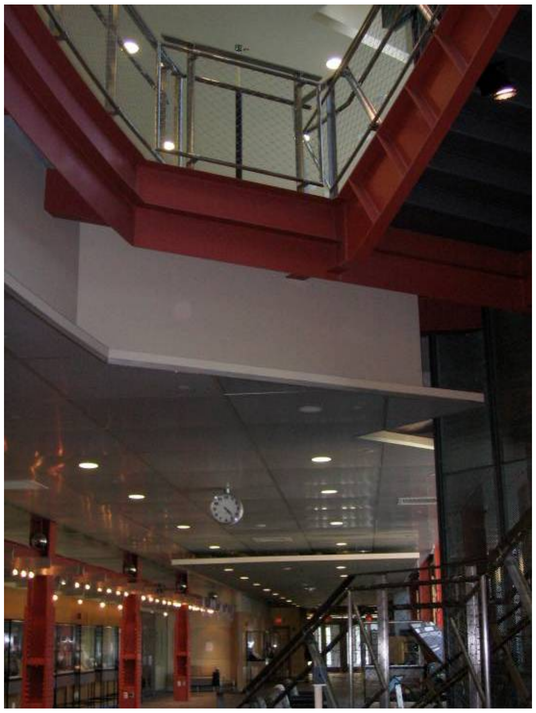


Figure 6.65 Representative steel structure and detail of floor system.

The testing consisted of two parts. In the first, the nodes were located in a multi-story atrium in the center of the structure. One node was placed on the upper level while the second node was located a line-of-sight distance of 19 feet. In the second test, the nodes were moved into the structure so they were no longer line-of-sight but still 19 feet apart as before. Two separate power levels were tested: power level 20 and power level 31. Power level 20 corresponded to the lowest transmission power required for a 100% reception rate in the atrium. For both tests, the sensors were mounted vertically 4 feet above the ground to limit multi-path effects. Based on the radiation patterns determined from the anechoic chamber tests the sensor nodes were oriented parallel to one another and the external antennas were vertical and parallel to one another and the board. In both cases, the steel structure resulted in approximately a 10% reduction in reception rate.

Reinforced concrete structures typically consist of both reinforced concrete framing and floor systems. A parking garage on the University of Illinois campus was chosen as an extreme test situation for the attenuation due to reinforced concrete, since parking garages are usually heavily reinforced. A set of loopback tests was conducted using two nodes and external antennas. The testing environment is pictured in Figure 6.66.

Two types of tests were conducted. In the first, line-of-sight tests were conducted on the upper floor of the parking garage. One node was placed on the upper level of the garage while the second node was placed a LOS distance of 19 feet away on the ramp to the lower story. In the second test, the garage structure was located between the two nodes so they were no longer line-of-sight. The nodes were moved east the same horizontal distance until the second node was located well under the upper story of the garage. Thus, the nodes were still 19 feet apart but no longer line of sight. Three power levels were tested: power levels 4, 5, and 10. Power level 4 corresponds to the highest power level where there was a decrease in reception rate due to the garage. For both tests, the sensors were mounted vertically 4 feet above the ground to limit multi-path effects. The sensor nodes were oriented parallel to one another and the external antennas were vertical and parallel to one another and the board.

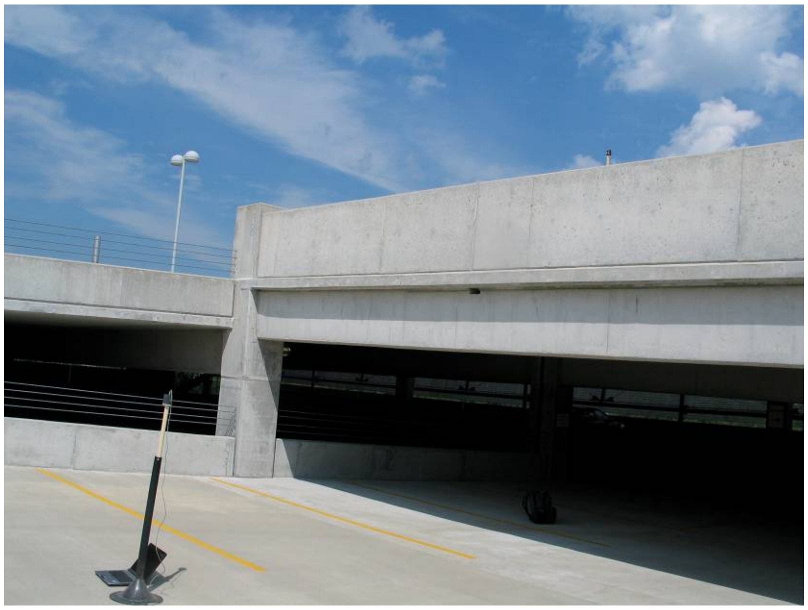


Figure 6.66 Reinforced concrete testing environment.

Overall, a negligible decrease in the reception rate due to the concrete structure was observed. Even at power level 4 the decrease in reception rate was only about 1%, while at higher power levels there was no observed change in reception rate. Given that the parking garage would be an extreme case of reinforcement, minimal signal attenuation in a building due to the concrete structure alone is expected. In any structure, however, there are many more influences on the transmission than the building structure alone; the communication environment is the result of the structural materials, cladding and finish materials, contents of the building, etc.

Concrete masonry unit (CMU) infill walls are common in many civil engineering structures. A set of loopback tests were conducted in the Newmark Civil Engineering Building on the University of Illinois campus to evaluate the degree of associated attenuation. The testing environment is pictured in Figure 6.67.

Again, the testing consisted of two parts. In the first test, one node was located inside the classroom 20 feet line-of sight through the doorway from the second node in the hallway. In the second test, the nodes were moved north until the CMU wall was in-between and they were no longer line-of-sight but still 20 feet apart as before. Both nodes were set at a height of 4-feet, configured with external antennas in the optimal orientation, and conducted at a transmission power of 10. Power level 10 was the lowest power level at which a 100% reception rate was achieved in the line-of-sight testing configuration.

Figure 6.67 CMU test environment.

The CMU infill resulted in a negligible reduction in the reception rate. This result is promising for other types of partitions that might be used in civil engineering structures such as gypsum board.

Table 6.19 Overall attenuation results of built environment elements with respect to line-of-sight tests in the same environment.

Built Environment Element	Reduction in Reception Rate (%)
Wireless Network	15
Steel Structure	10
Reinforced Concrete	~ 0
CMU Infill	~ 0

Mahomet Bridge implementation

Wireless communication tests were conducted on a bridge in Mahomet, Illinois, which will be used for the full-scale implementation of a smart sensor network (Jang et al. 2009). When the *ReliableComm* communication protocol is used in an application, it ensures that no command or data packets are dropped during transmission because initially dropped packets are resent until 100 percent of the data is transmitted. As a result, applications using *ReliableComm* do not require communication environments to support 100 percent packet reception rates at all time. For this reason, the communication tests conducted at the Mahomet Bridge determined the antenna power required for at least 95 percent reception rate between girders.

Given a proposed sensor layout for monitoring the bridge, the bridge geometry, and the radiation pattern tests, the following test protocol was developed. The fixed node was placed underneath the bridge on the girder closest to the north support (on the right in Figure 6.68). The remote node was moved from girder to girder underneath the bridge. The nodes were located in environmentally hardened enclosures. Two antenna configurations were considered: the external antenna aligned perpendicular to the plane of the bridge, which is optimal for communication along the face of the bridge, and parallel to the face of the bridge, which is considered poor for communication along the face of the bridge, but optimal for communication across the deck of the bridge.

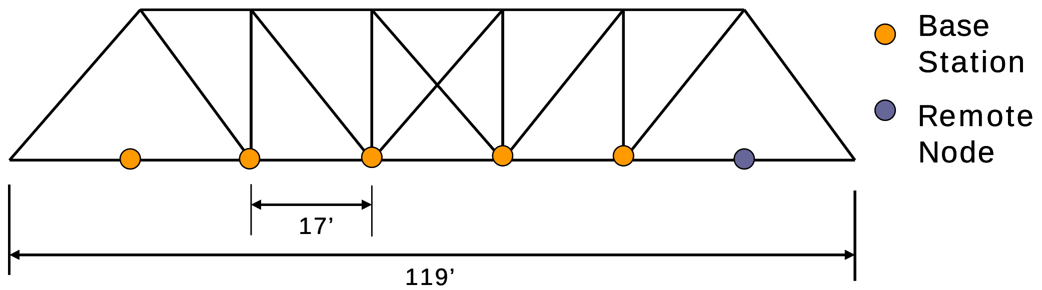


Figure 6.68 Mahomet Bridge testing layout and dimensions.

Table 6.20 below gives the power required to achieve at least 95% reception rate for both orientations. Based on previous test results, the external antenna at power level 5 should have been sufficient to reach to the furthest girder with at least 95% reception rate; however, significantly more power was required. Therefore, the steel structure had a considerable impact on the antenna performance. The fact that the steel of the Mahomet bridge proved to introduce more interference that the steel Siebel Center building is likely due to the higher density of the steel members at the bridge, although other unexplained factors effecting interference may be present. These results highlight the need to consider the built environment and conduct on-sight testing prior to network implementation. Furthermore, in-situ tests could be used to optimize network communication protocols by selecting the optimal power level for a specific type of communication, *i.e.*, single-hop or multi-hop.

Table 6.20 Power level required for 95% reception rate on the Mahomet Bridge.

Distance to Girder (feet)	Power Level Required	
	Good Orientation	*Poor Orientation*
17	5	5
34	5	5
51	15	10
68	10	15
85	10	15

The use of an external antenna with the Imote2 significantly improves both ideal transmission range and reliability for all power levels. However, building materials can significantly reduce the reception rate and thus increase packet loss. Although IEEE 802.11b network interference was found to cause packet loss, the decrease in reception rate can be avoided by accounting for the wireless network in the testing environment.

Extensive Imote2 radio and antenna communication evaluation has been conducted. The results of this work highlight the importance of understanding antenna characteristics as well as the communication environment to achieve optimal communication between the sensor nodes. Given that the antenna behaves as a dipole, placing the antennas parallel to one-another and the board, which radiates as well, would be the optimal antenna orientation for communication. The ideal antenna configuration will ensure the minimization of interference while limiting power consumption. The information this study provides will be useful to other researchers and engineers deploying Imote2s for a variety of applications.

6.2 Power management

One of the most critical issues in achieving a long-term smart sensor SHM implementation is careful power management. Power management must be addressed from both sides of the equation: power supply and power consumption. The primary focus of this section will be on the relative effects of parameter selection on the network power consumption. Although batteries represent only one of many options for powering the sensor nodes, a predictive model for battery life based on the estimated average power consumption of the smart sensors is also presented to illustrate the implications of resource management decisions.

6.2.1 Relative power consumption calculation

The amount of power that a smart sensor consumes depends on the power consumption of each of its components and how they are used at any given time. As previously described, the Imote2's processor, the PXA271, can run at varying speeds based on application requirements, resulting in varying current consumption. In addition, the sensor board draws a certain amount of current when it is running and when it is idle. For the applications presented in Chapter 3 and 4, there are five primary power consumption states of the Imote2 with the sensor board attached:

1) Deep sleep mode
2) Startup – initial state when Imote2 is turned on or wakes from deep sleep mode
3) Imote2 processor @ 13MHz (lowest operating speed)
4) Imote2 processor @104MHz (intermediate operating speed)
5) Sensing with the Imote2 processor @ 104MHz

The current draw in each of the four states depends on the hardware used, in particular the battery board and the sensor board. The battery board used will determine the current draw in deep sleep mode; the current draw is the sum of the deep sleep mode of the Imote2 plus any idle current draw of the battery board. The two types of battery boards used throughout this study are the Intel battery board, a first generation battery board, and the newer Crossbow battery board (IBB2400CA). The fundamental difference between the battery boards is that the Intel board has a buck-boost regulator on board to regulate the power from the batteries to the Imote2 at ~3.2V, while the Crossbow battery board does not. The implications of this difference are that the Intel battery board can operate over a wider range of battery voltages without affecting the performance of the Imote2 but it consumes more power when the Imote2 is asleep; the Crossbow battery board is limited to a smaller range of battery voltages, but it consumes no power when it is idle. Both battery boards are presented in this study to illustrate the effects of these hardware differences on overall power consumption.

When first switched on or after waking from the deep sleep mode, the Imote2 experiences an initial high current draw that lasts less than a second. As shown in Figure 6.69, if the Imote2 is put into deep sleep mode for four seconds, the current will spike during the last 0.67 seconds before it settles to the idle/listening mode. While the spike is only a short increase in the current level, its effects will compound if the Imote2 is in a sleep-wake cycle during most of its life.

The sensor board used determines the power consumption in the non-sleep states. The greatest impact of the power consumption of the sensor board is during sensing, because power consumption in that state is the sum of the power consumed by the active sensor board and the power consumed by the Imote2 operating at 104MHz. In addition, if all portions of the sensor board are powered when the Imote2 is powered, the power consumption is increased, even in the non-sensing power states. The SHM-A sensor board draws approximately double the amount of current as the ITS400CA (basic sensor board) during sensing; however, in the non-sensing states, the power to most portions of the SHM-A board is cut-off, while the ITS400CA is powered all the time the Imote2 is not in the deep sleep mode.

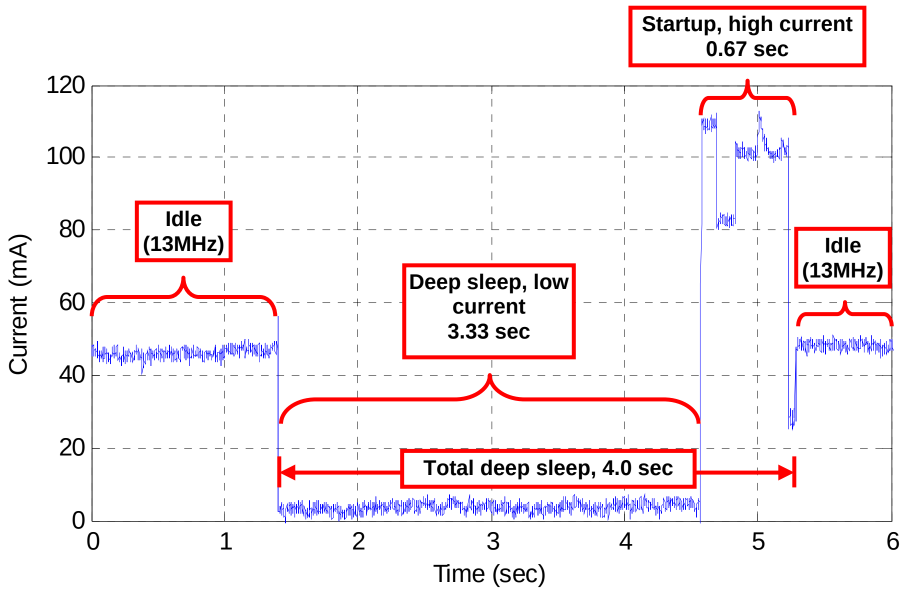


Figure 6.69 Deep sleep power states.

During each phase of the *ThresholdSentry/AutoMonitor* application, the remote nodes are in one of the five power states. The current draw from three D-cell batteries during each state has been determined experimentally for both battery boards and both sensor boards. Figure 6.70 shows the approximate values measured for each state with different hardware configurations.

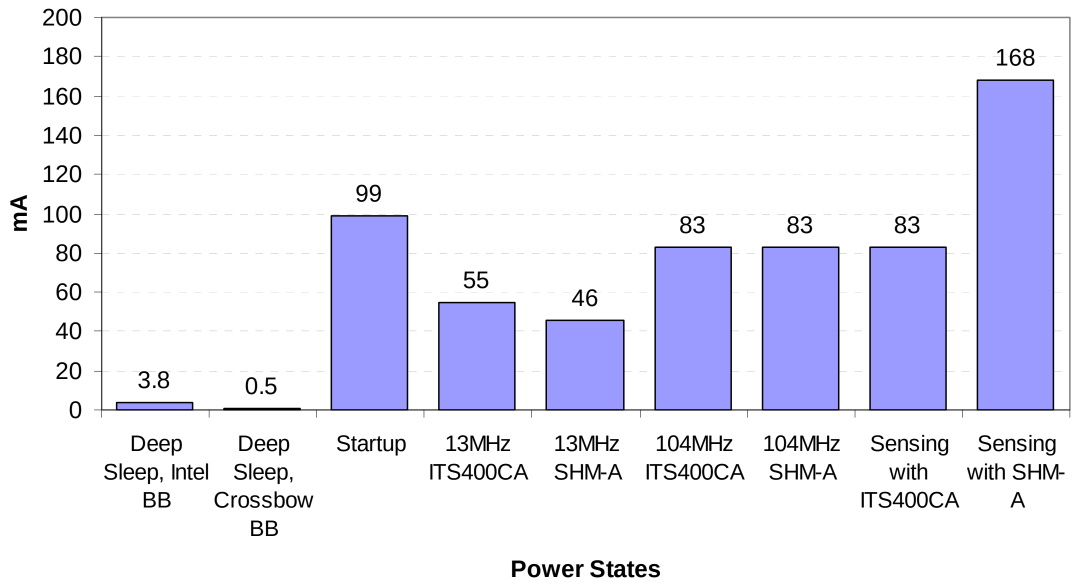


Figure 6.70 Approximate current consumption of the power states for the Imote2 with different battery and sensor boards.

These current values may be used to assess the relative impact of the various application parameters on the overall power consumption at each node. In addition, a large number of software/application parameters that can be considered when optimizing power consumption and network performance. Some parameters are application constants that are not expected to change significantly by most users, while other parameters are expected to change significantly based on the needs of the applications. The parameters are summarized in Table 6.21.

The quantity used in this study to represent the power consumption on the nodes in the network is the average daily current draw, I_{avg}. This value is determined based on the amount of time the sensor node is in each power state in a given day with a given set of parameters. The assessment of these time periods must be investigated for each of the applications running on the network: *RemoteSensing*, *SnoozeAlarm* and *ThresholdSentry*.

Table 6.21 Power management optimization parameters

Category	Parameter	Description
Network parameters	Network size	Number of nodes in network
	Sentry network size	Number of nodes involved in
Sensing parameters	Sampling rate	*RemoteSensing* sampling rate
	Number of points	Number of data points in *RemoteSensing*
	Number of channels	Number of channels measured in *RemoteSensing*
	Number of *RemoteSensing* events	Number of *RemoteSensing* events per day
RemoteSensing wait times	Synchronization wait time	Time before sensing starts to synchronize the network
	Extra wait time	Extra time added to the total time the base station node waits between sending the sensing command and requesting data
	Extra sensing delay per node	Additional extra wait time per node to account for longer communication times in larger network
SnoozeAlarm times	Sleep interval	Sleep interval in *SnoozeAlarm* mode
	Listen interval	Short wake/listen time in *SnoozeAlarm* mode
ThresholdSentry times	Check interval	Time between sentry node checks
	Check sensing time	Time sentry node senses when checking data

RemoteSensing

The power consumption associated with the *RemoteSensing* application is not only the result of sensing, but also the time each node spends waiting and communicating data. When *RemoteSensing* runs on a network, the longest portion of the total application is often the time dedicated to data communication. Prior to sensing on the remote nodes, a period of network time synchronization occurs where the nodes are at the lowest processor speed (13MHz). Following time synchronization, time for communication of commands and some additional wait time take place to ensure all nodes have received their sensing parameters. The sensor board is powered down after sensing but the processor speed remains at 104MHz until resampling is complete. Subsequently, the node returns to the lowest processor speed and waits for a request from the base station node to send its data. Once the data has successfully been sent, the remote node puts itself back into the *SnoozeAlarm* cycle. For a large network significant differences will be experienced in the time each node is in an idle state waiting to send its data and the amount of time it is in the *SnoozeAlarm* mode depending on its place in the node ID order; the result is large differences in power consumption. Figure 6.71 illustrates this difference for a network of 30 nodes acquiring 10,000 data points on three channels at

100Hz. Clearly the first node to report its data (Node 1) will use much less power than the last node to report its data (Node 30). To counter this effect, the node order is reversed in the *AutoMonitor* application each time *RemoteSensing* is called.

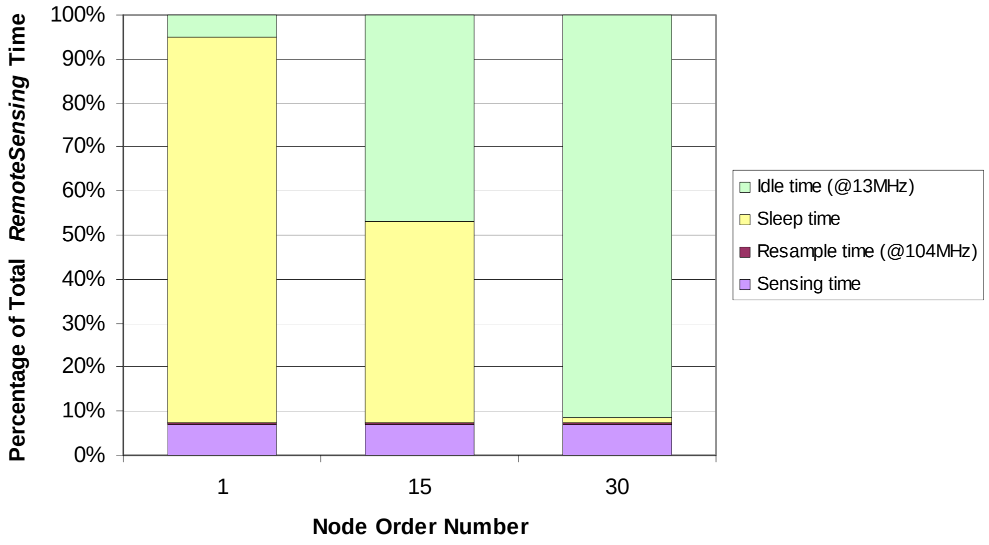


Figure 6.71 Effect of node order on *RemoteSensing* power state times.

The average current draw during *RemoteSensing* for the *i*th node in an *n*-node network is given by:

$$I_{\mathrm{RS\_avg}}(n,i,\ fs,ch,d)=\frac{I_{13}t_{13}(n,i,\ fs,ch,d)+I_{\mathrm{slp}}t_{\mathrm{slp}}(n,i,\ fs,ch,d)+I_{\mathrm{sense}}t_{\mathrm{sense}}(\ fs,d)+I_{104}t_{104}(\ fs,ch,d)}{t_{\mathrm{RS}}(n,\ fs,ch,d)} \quad \textbf{(6.8)}$$

where:

fs = Sampling rate
ch = Number of sensing channels
d = Number of data points
I_{13} = Imote2 @ 13MHz (idle) current draw
t_{13} = Time @ 13MHz (idle)
I_{slp} = Imote2 in deep sleep mode
t_{slp} = Time in deep sleep mode
I_{sense} = Imote2 with sensor board running/acquiring data
t_{sense} = Sensing time
I_{104} = Imote2 current @ 104MHz (not sensing)
t_{104} = Resampling time
t_{RS} = Total *RemoteSensing* application time

Although accounting for the highest current draw during the remote sensing application, sensing does not necessarily account for the majority of the power consumption. For the same sensing parameters (*fs* = 100Hz, *ch* = 3, *d* = 10,000), using the SHM-A sensor board and the Intel battery board, the contribution to the average current draw for the middle node from each power state is shown in Figure 6.72. This figure illustrates that the waiting time (i.e. the time a node waits to be called on for data) dominates the power consumption as the network size grows. This waiting is primarily a function of the number of sensed data points and the network size.

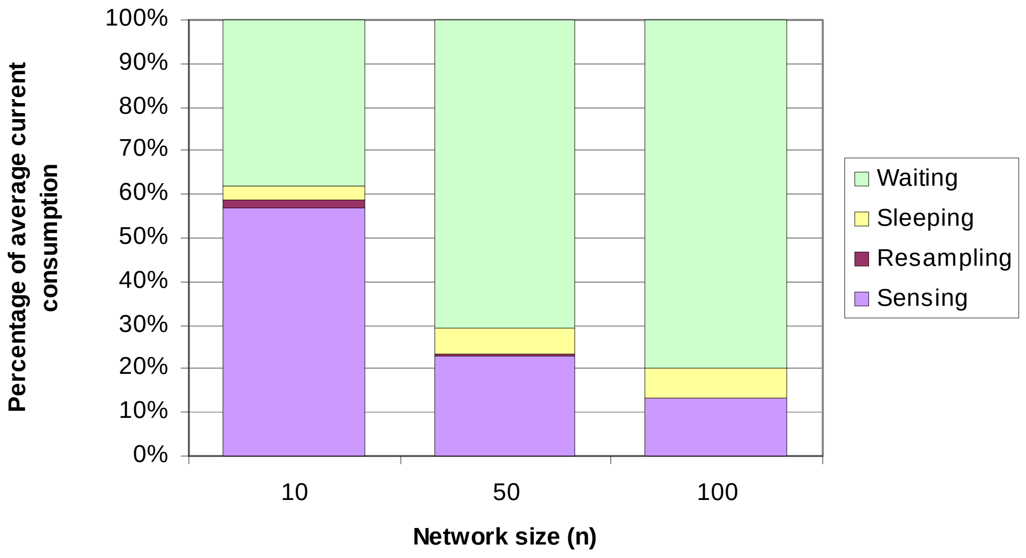


Figure 6.72 Current consumption contribution from various portions of *RemoteSensing* as a function of network size.

The average current draw over the total application time decreases with the size of the network because the time that the nodes are either sleeping or idle increases relative to the amount of time sensing, which remains the same. The total application time, however, increases linearly with the network size as the time to communicate and write the data to the PC increases. Figure 6.73 shows these trends for the middle node in the *RemoteSensing* application acquiring 10,000 data points from three channels sampling at 100 Hz.

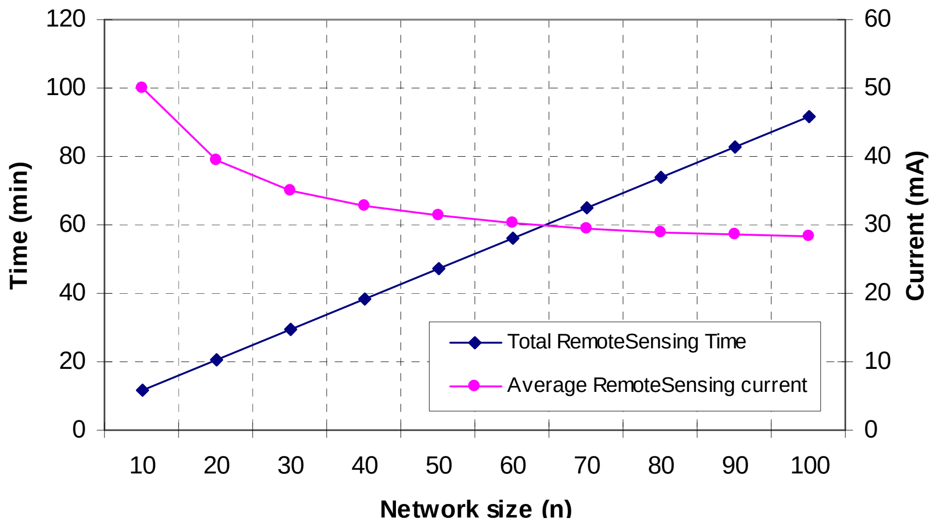


Figure 6.73 *RemoteSensing* application time and current consumption.

The duty cycle of the remote sensing application, δ_{RS}, is defined in terms of the number of remote sensing events in a given day, n_{RS}:

$$\delta_{\mathrm{RS}}(n_{\mathrm{RS}}, n, fs, ch, d) = \frac{n_{\mathrm{RS}} t_{\mathrm{RS}}(n, fs, ch, d)}{24\mathrm{hr}} \tag{6.9}$$

ThresholdSentry

The power consumption associated with *ThresholdSentry* will only affect the sentry nodes. *ThresholdSentry* is not intended to involve all of the nodes in the network. It operates on those nodes that are expected to see reasonable levels of vibration such as nodes towards the middle of a bridge span or nodes measuring critical members. Since these nodes have additional duties beyond their involvement in network-wide sensing, they are expected to consume more power than non-sentry nodes. Their additional power consumption depends on how many sentry nodes share the sentry duty, how often they are woken, and how long they are awake during their sentry check.

One complete *ThresholdSentry* cycle occurs when each of the nodes in the sentry network have taken one turn acting as the sentry node. The total time, t_{TS}, for one cycle (assuming *RemoteSensing* is not triggered) is:

$$t_{\mathrm{TS}}(n_{\mathrm{s}}) = n_{\mathrm{s}}(t_{\mathrm{TS\_int}} + t_{\mathrm{TS\_check}}) \tag{6.10}$$

where n_{s} = the number of sentry nodes

$t_{\mathrm{TS\_int}}$ = the interval between sentry checks

$t_{\mathrm{TS\_check}}$ = the time the sentry node is awake during the threshold check

The *ThresholdSentry* duty cycle, δ_{TS}, the ratio of the time each sentry node is awake to the total *ThresholdSentry* cycle time:

$$\delta_{\mathrm{TS}}(n_{\mathrm{s}}) = \frac{t_{\mathrm{TS\_check}}}{t_{\mathrm{TS}}(n_{\mathrm{s}})} \tag{6.11}$$

The current consumption during the sentry check is that of the sensing power state (I_{sense}).

SnoozeAlarm

The primary purpose of the *SnoozeAlarm* mode is to save power by allowing the nodes to be in the deep sleep state the majority of the time they are not involved in another application. In this state, the nodes sleep for a period of time followed by a very short listening period; this process repeats until the node is called upon to participate in another function. In the *AutoMonitor* application, when the network nodes are not involved in *RemoteSensing* or *ThresholdSentry*, they are in the *SnoozeAlarm* mode. The *SnoozeAlarm* duty cycle, δ_{TS}, is the ratio between the time the node is awake, t_{wake} and the time for one *SnoozeAlarm* cycle, $t_{\mathrm{wake}} + t_{\mathrm{sleep}}$:

$$\delta_{\mathrm{SA}} = \frac{t_{\mathrm{wake}}}{t_{\mathrm{wake}} + t_{\mathrm{sleep}}} \tag{6.12}$$

The current draw during wake time is that of the idle power state (I_{13}).

Each time the Imote2 is put into deep sleep mode, 0.67 seconds of the time dedicated to sleeping is actually spent in a high current draw startup state immediately before it wakes up. The duty cycle for this startup current spike is:

$$\delta_{\text{start}} = \frac{0.67\,\text{sec}}{t_{\text{sleep}}} \tag{6.13}$$

The average current draw during this state is I_{start} = 99 mA.

Average current draw

Once the duty cycles for each portion of the daily operation of the nodes in the network have been established, the time associated with each portion in a given day can be calculated for the sentry and non-sentry nodes. The times for the sentry nodes are:

$$T_{\text{RS}} = 24\text{hr} \cdot \delta_{\text{RS}} \tag{6.14}$$

$$T_{\text{TS\_sensing}} = 24\text{hr} \cdot (1-\delta_{\text{RS}})\delta_{\text{TS}} \tag{6.15}$$

$$T_{\text{SA\_listen\_sentry}} = 24\text{hr} \cdot (1-\delta_{\text{RS}})(1-\delta_{\text{TS}})\delta_{\text{SA}} \tag{6.16}$$

$$T_{\text{SA\_sleep\_sentry}} = 24\text{hr} \cdot (1-\delta_{\text{RS}})(1-\delta_{\text{TS}})(1-\delta_{\text{SA}})(1-\delta_{\text{start}}) \tag{6.17}$$

$$T_{\text{SA\_start\_sentry}} = 24\text{hr} \cdot (1-\delta_{\text{RS}})(1-\delta_{\text{TS}})(1-\delta_{\text{SA}})\delta_{\text{start}} \tag{6.18}$$

where:

T_{RS} = the time per day spent in the *RemoteSensing* application
$T_{\text{TS\_sensing}}$ = the timer per day spent sensing for *ThresholdSentry*
$T_{\text{SA\_listen\_sentry}}$ = the time per day the node is awake for *SnoozeAlarm*
$T_{\text{SA\_sleep\_sentry}}$ = the total time per day that a node is asleep (outside of the *RemoteSensing* application)
TSA_start_sentry = the total time per day that a node spends in the startup/wakeup mode

The non-sentry nodes do not have any time spent sensing apart from during *RemoteSensing,* so their sleep and listen times ($T_{\text{SA\_listen\_nonsentry}}$ and $T_{\text{SA\_sleep\_non-sentry}}$, respectively) are slightly higher:

$$T_{\text{SA\_listen\_non-sentry}} = 24\text{hr} \cdot (1-\delta_{\text{RS}})\delta_{\text{SA}} \tag{6.19}$$

$$T_{\text{SA\_sleep\_non-sentry}} = 24\text{hr} \cdot (1-\delta_{\text{RS}})(1-\delta_{\text{SA}})(1-\delta_{\text{start}}) \tag{6.20}$$

$$T_{\text{SA\_start\_non-sentry}} = 24\text{hr} \cdot (1-\delta_{\text{RS}})(1-\delta_{\text{SA}})\delta_{\text{start}} \tag{6.21}$$

The average current draw for the sentry and non-sentry nodes can be determined by weighting the amount of time the nodes are in a particular power state by the current consumed in that power state as follows:

$$I_{\mathrm{avg\_sentry}} = \delta_{\mathrm{RS}} I_{\mathrm{RS\_avg}} + (1-\delta_{\mathrm{RS}})\delta_{\mathrm{TS}} I_{\mathrm{sense}} + \cdot(1-\delta_{\mathrm{RS}})(1-\delta_{\mathrm{TS}})\delta_{\mathrm{SA}} I_{13} + \dots$$
$$\dots + (1-\delta_{\mathrm{RS}})(1-\delta_{\mathrm{TS}})(1-\delta_{\mathrm{SA}})(1-\delta_{\mathrm{start}}) I_{\mathrm{sleep}} + \dots \qquad (6.22)$$
$$\dots + (1-\delta_{\mathrm{RS}})(1-\delta_{\mathrm{TS}})(1-\delta_{\mathrm{SA}})\delta_{\mathrm{start}} I_{\mathrm{start}}$$

$$I_{\mathrm{avg\_non\text{-}sentry}} = \delta_{\mathrm{RS}} I_{\mathrm{RS\_avg}} + (1-\delta_{\mathrm{RS}})\delta_{\mathrm{SA}} I_{13} + (1-\delta_{\mathrm{RS}})(1-\delta_{\mathrm{SA}})(1-\delta_{\mathrm{start}}) I_{\mathrm{sleep}} + \dots \qquad (6.23)$$
$$\dots + (1-\delta_{\mathrm{RS}})(1-\delta_{\mathrm{SA}})\delta_{\mathrm{start}} I_{\mathrm{start}}$$

Eqs. 6.15 and 6.16 illustrate the many parameters that impact the current consumption calculation. The relative effect of the parameters on the overall power consumption of the network is investigated in the following section.

6.2.2 Power consumption parameter sensitivity

With the average current draw, I_{avg}, as the relative measure of the node's power consumption, the parameters listed in Table 6.21 can be varied to determine their effect on this value. The parameters investigated in this study are:

- Network size, n
- Number of *RemoteSensing* events, n_{RS}
- Sentry network size, n_s
- *SnoozeAlarm* duty cycle, δ_{SA}

The effects of the network size on the average current draw during the *RemoteSensing* application has already been investigated in the previous section. In this section, the effect of the network size on the overall current draw will be evaluated. Unless otherwise noted, the sensing and network parameters used in the following examples are given in Table 6.22. The sensor board investigated is the SHM-A sensor board and both the Intel and the Crossbow batteries will be evaluated.

Table 6.22 Network and sensing parameters for power consumption parameter evaluation.

Parameter	Value
Network size, n	30
Number of *RemoteSensing* events per day, n_{RS}	2
Sampling rate, fs	100 Hz
Channels sampled, ch	3
Number of data points, d	10,000
Node order number, i	$\lceil 0.5n \rceil$
Number of sentry nodes, n_s	$\lceil 0.25n \rceil$
ThresholdSentry check interval, t_{TS_int}	15 min
ThresholdSentry check time, t_{TS_check}	20 sec
SnoozeAlarm duty cycle, δ_{SA}	4.76%
Startup duty cycle, δ_{start}	6.70%

The following plot shows the average *AutoMonitor* current draw for sentry and non-sentry nodes as a function of the network size (using the Intel battery board). For a very small network, the discrepancy between the sentry and non-sentry currents is larger, because when the sentry network is very small, only a few nodes fulfill the sentry requirements. This effect diminishes as the network size increases. Also, as the network size increases, the discrepancy between the two node types decreases.

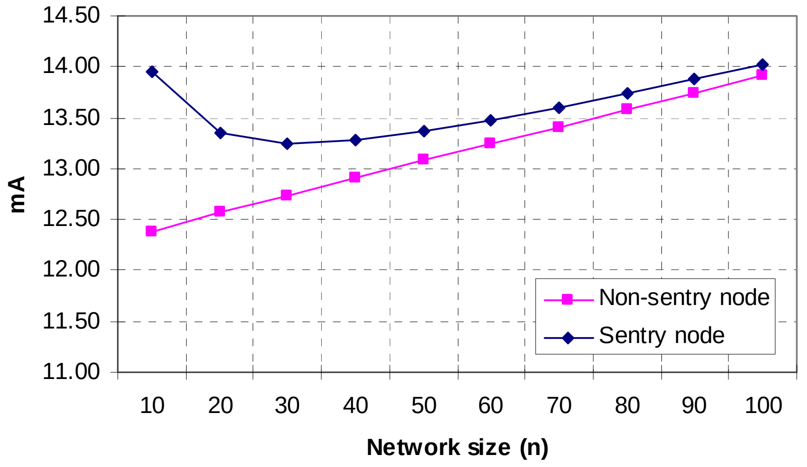


Figure 6.74 Average current draw as a function of network size.

One parameter expected to have a significant effect on the network power consumption is the number of times per day that *RemoteSensing* is run on the network. This value is controlled by the user via the input file for *AutoMonitor*. Figure 6.75 shows the linear increase in average current as the number of *RemoteSensing* events increases. Increasing the number of *RemoteSensing* events from one to two times per day increases the average current by approximately four to five percent, depending on the hardware in use.

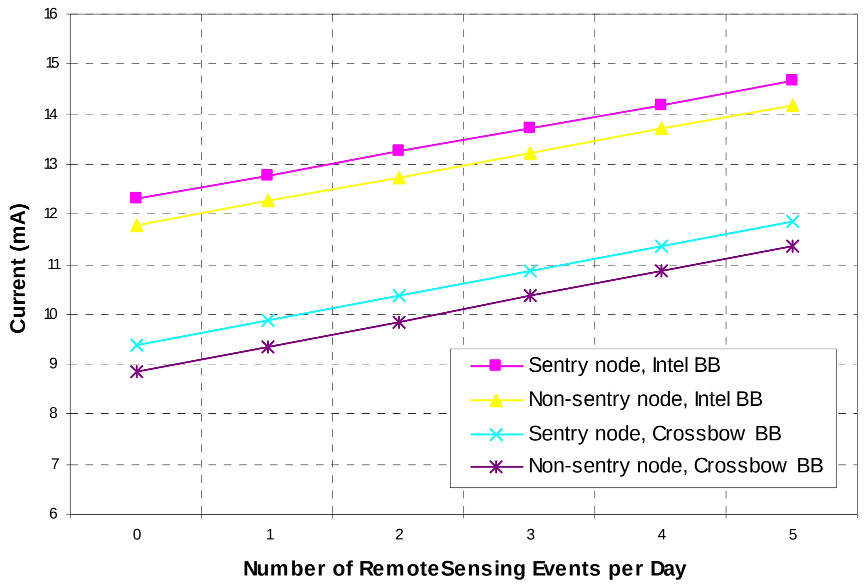
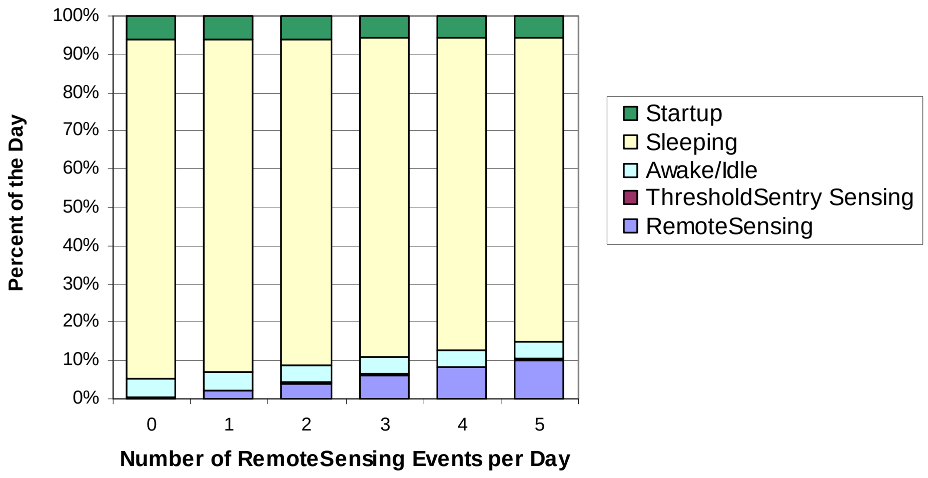
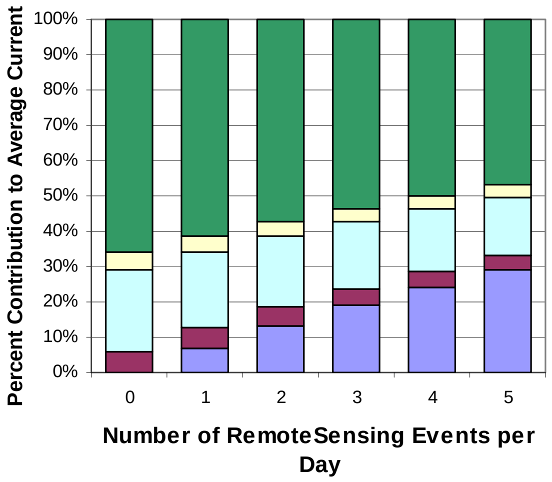


Figure 6.75 Average current as a function of number of *RemoteSensing* events per day.

While *RemoteSensing* is the most power hungry application running in the course of daily network operation (as opposed to *SnoozeAlarm* or *ThresholdSentry*), doubling the number of times it occurs does not have a significant effect on overall network power consumption. The relative time spent in *RemoteSensing* is small compared to the amount of time the nodes are in the *SnoozeAlarm* mode. Figure 6.76 illustrates the relative time contribution of each portion of the network operations for the sentry nodes and how this translates to the relative current consumption contribution of each portion. Although the deep sleep mode is by far the least power hungry mode in terms of current draw, the length of time spent sleeping can result a large portion of the power consumption in a given day. The most significant source of power drain, and perhaps the most unexpected, is that which results from the startup current spike each time the node is awoken.





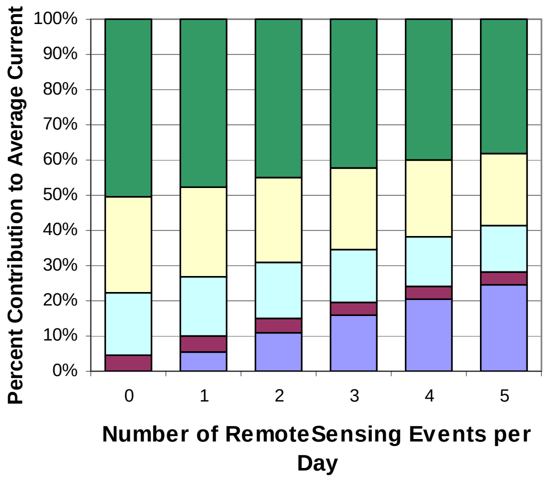


Figure 6.76 Relative contribution of each portion of monitoring to the total time (top) and the average current consumption for the sentry nodes (Crossbow battery board, left, and Intel battery board, right).

Given that the nodes may spend the majority of their day in the *SnoozeAlarm* mode, the effect of the *SnoozeAlarm* duty cycle on power consumption must be investigated. The wake/listen time should be at least 500 ms to ensure that the node has time to wake up and to receive/process any commands it may be sent during that period. In some cases, this value may need to be increased to accommodate a poor communication environment. For the purpose of this investigation, the wake time is held constant at 500 ms, while the sleep time is varied. The longer the sleep time, the more power savings can be expected, both from minimizing the startup current spikes and the reduced current draw associated with deep sleep. However, long sleep times will make waking the network a more time-consuming and thus power-consuming task. Because the wake up commands are sent in a series of unicast messages, it is possible for nodes emerging from the deep sleep mode to miss their wake up message the first time around. The longer the sleep time, the longer the elapsed time before the node has the chance to receive another wake up message. The duty cycles represented in Figure 6.77 represent sleep times from 8 to 24 seconds. Tripling the sleep time from 8 to 24 seconds, which is the same as reducing the reducing the duty cycle from 5.9 to 2 percent, results in over a 6 mA average current draw decrease on both the sentry and non-sentry nodes.

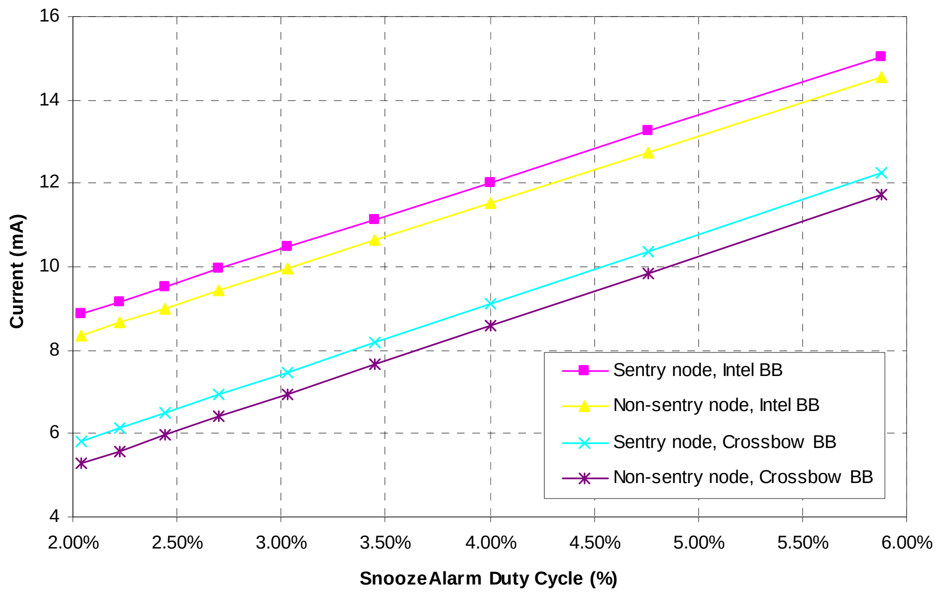


Figure 6.77 Average current as a function of *SnoozeAlarm* duty cycle for sentry and non-sentry nodes

6.2.3 Battery life prediction

To date, WSSNs have typically been battery powered, as batteries provide an inexpensive and convenient means to power the nodes. However, for long-term applications, batteries may not represent the best solution. The cost of accessing sensor nodes for battery replacement may mitigate any savings associated with their initial deployment versus a traditional wired network if replacement has to be done too often or access is especially challenging. Larger batteries, or more batteries connected in parallel, can provide a longer life; however, size and mounting restrictions may limit these measures. Careful battery selection and application design may allow batteries to be a viable option for powering WSSNs in settings where occasional access to the nodes is reasonable.

The purpose of this section is to provide a basis for estimating battery life when the average current consumption is established, either through published values or testing. In particular, the battery life of 3 D-cell batteries will be investigated, because the three full-scale validation studies presented in Chapter 7 utilize 3 D-cell batteries to power the nodes.

To convert the average current draw into an estimated battery life, the capacity of the batteries being used must be considered. Typical battery capacities are given in milliamp-hours (mAh). This value gives an indication of how long a battery can supply a particular constant current and thus is usually defined for a specific discharge rate (in mA). Due to the way in which the chemical reactions within a battery take place, the current that can be supplied is limited. If the current draw on the battery is higher, it will drain the battery more quickly than for a lower current draw. Additionally, the voltage drop as a function of time, even for a constant current draw, is not linear as shown in Figure 6.78. For these reasons, a general relationship between average current consumption (as calculated in the previous section) and estimated battery life is difficult to establish. Figure 6.79 shows the estimated capacity (down to 0.8V) for Energizer Industrial D-cell batteries (EN95) at four discharge rates. The resulting estimated battery life for each discharge rate is given in Table 6.23.

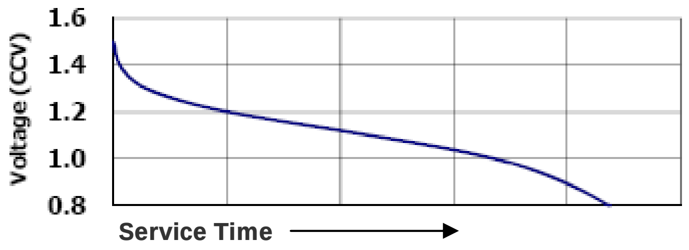


Figure 6.78 Typical voltage discharge curve for D-cell alkaline batteries (Energizer, 2009).

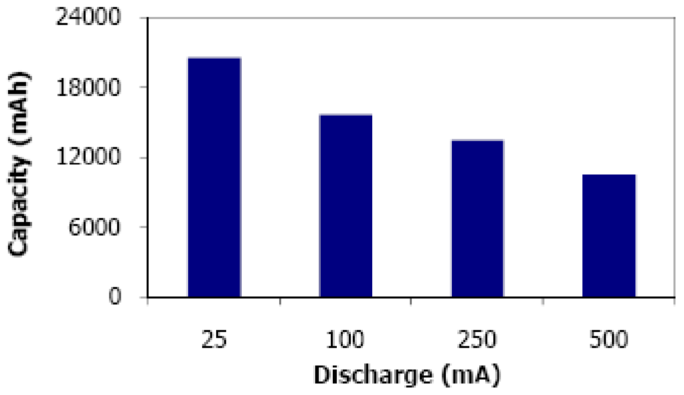


Figure 6.79 Alkaline D-cell battery capacity to 0.8V in terms of discharge rate (Energizer, 2009).

Table 6.23 Estimated battery life to 0.8V for D-cell batteries with varying discharge rates.

Discharge rate (mA)	Capacity (mAh)	Estimated life (hrs)
25	20500	820
100	16000	160
250	13500	54
500	10500	21

According to this these values, the relationship between the estimated life (to drain to 0.8V) and the discharge is determined to be:

$$T_{\text{service,est}} = 41{,}960\text{hrs} \cdot (I_{\text{discharge}})^{-1.25} \qquad (6.24)$$

This relationship must be adjusted to account for the fact that the minimum required battery voltage (per battery) is greater than 0.8V for the Imote2. The minimum battery voltage required by the Imote2 to operate depends on the battery board being used and the power state of the node. As mentioned in the previous section, the range of battery voltage that the Imote2 can operate with varies for each battery board. During sensing, the battery voltage is drawn down quite significantly. It recovers almost completely after the completion of sensing, especially if it is put back into *SnoozeAlarm* mode. Though the voltage will recover after sensing, the lowest voltage that is experienced during sensing must stay above the minimum voltage required by the battery board being used with the Imote2. Figure 6.80 illustrates this phenomenon.

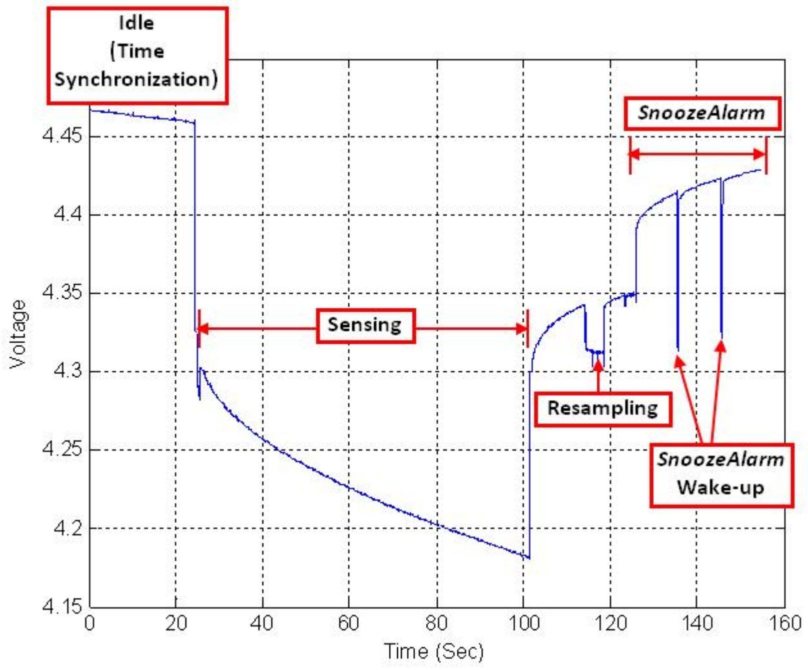


Figure 6.80 *RemoteSensing* voltage drop and recovery.

The size of the voltage drop during *RemoteSensing*, V_{drop}, is not only a function of the sensor board but also the initial voltage, V_{i}. The values shown in Table 6.24 have been determined experimentally for all combinations of battery board and sensor board. Although the nominal voltage of most alkaline batteries is 1.5V, they typically start a ~1.6V. The maximum voltage for the Intel battery board is limited by the voltage provided by three batteries: 1.6Vx3 batteries = 4.8V, although it can handle up to 5.5V. The Crossbow battery board does not allow the Imote2 to turn on if the battery voltage is above 4.7V.

Table 6.24 Voltage ranges for each battery board/sensor board combination

Battery Board	V_{max}	$V_{min\_SHM\text{-}A}$	V_{min_ITS}	$V_{range\_SHM\text{-}A}$	V_{range_ITS}
Intel	4.8	2.9	2.9	1.9	1.9
Crossbow	4.7	3.8	3.6	0.9	1.3

To get the adjusted estimated service life, $T_{service,est}$, for the Imote2 running *RemoteSensing*, the estimated service life must be scaled by the ratio of the actual usable voltage range, V_{range}, and the specified range for the battery:

$$T_{service,est} = \frac{V_{range}}{3 \cdot (1.5\text{V} - 0.8\text{V})} \cdot 41{,}960\text{hrs} \cdot (I_{discharge})^{-1.25} \tag{6.25}$$

V_{range} is a function of the sensor board and battery board combination as defined in Table 6.24. This relationship is plotted in log-log scale in Figure 6.81.

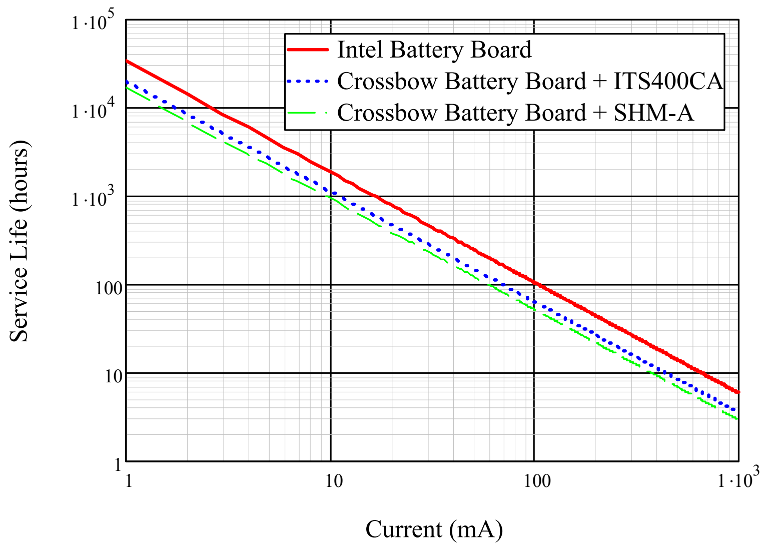


Figure 6.81 Estimated constant current service life for various hardware configurations for 3 D-cell batteries.

Using the average current calculated in the previous section for a network with the parameters defined in Table 6.22 the estimated life of 3 D-cell batteries are shown in Table 6.25.

Table 6.25 Estimated service life (days) for 3 D-cell batteries with various hardware combinations.

Battery Board	Sensor Board			
	SHM-A		ITS400CA	
	Sentry	Non-Sentry	Sentry	Non-Sentry
Intel	63	66	62	63
Crossbow	49	52	57	59

An Imote2 using the SHM-A board with the Intel battery board is expected to last longer than with the Crossbow battery board because the Intel battery board has a larger voltage range over which the Imote2 and sensor board can operate. In the case of the ITS400CA sensor board, the difference between the two battery boards is not as significant, because the low sleep current of the Crossbow battery board has as much of an effect as the available voltage range, as the ITS400CA experiences less of a voltage drop during sensing. While these estimate values help illustrate some of the relative differences associated with various hardware combinations, the actual battery life may vary somewhat. The reason for this variation is that these estimates are based on an average current draw and a constant current service life relationship. In reality, the current draw is non-constant. Even with the battery life estimates presented in this section, batteries alone may not be the best solution for all application. Alternative power sources, such as solar or wind power with rechargeable batteries, should be considered.

6.3 Summary

Many of the essential considerations for the implementation of a long-term, autonomous network of smart sensors have been presented in this chapter. Achieving optimal communication through careful selection of radio and antenna configuration is critical to the successful functionality of the network and guidelines for their selection have been summarized. The effect of network hardware and operating parameters on power consumption has also been investigated. To conserve limited power resources, the appropriate selection of the parameters by the network designer is important. In the case of both communication and power optimization, the network designer/operator must be aware of the network environment and balance the desired network output with long-term performance goals.

Chapter 7

FULL-SCALE VALIDATION

This chapter presents three full-scale validation tests that have proven the hardware and software developed by this research and driven further development and improvement of both. The tests conducted at the Stawamus Chief Pedestrian Bridge demonstrate the functionality of the basic synchronized sensing software and fault tolerance measures and shows the performance of the SHM-A sensor board. The seven-week Siebel staircase implementation illustrates the power savings and robust behavior of the sleep/wake cycle used in conjunction with synchronized network sensing and ultimately informs larger-scale network deployments. As the goal of this research is to prove the suitability of the developed framework for full-scale, autonomous operation, the last validation test, at the Jindo Bridge, demonstrates the autonomous network operation software and is currently ongoing.

7.1 Stawamus Chief Pedestrian Bridge

The newly constructed Stawamus Chief Pedestrian Bridge (Figure 7.82) employs an arched suspension design that crosses over the Sea-to-Sky Highway in Vancouver, British Columbia. A series of tests were conducted to characterize the dynamic behavior of the arches with specific focus on their susceptibility to wind-induced vibration. These tests utilized the Imote2 smart sensor platform and the SHM-A sensor board as well as application software developed under the Illinois Structural Health Monitoring Project (ISHMP). These tests highlighted the applicability of the Imote2s deployed for short-term structural testing. The results of the tests also validated several aspects of the sensor hardware and application software designs.

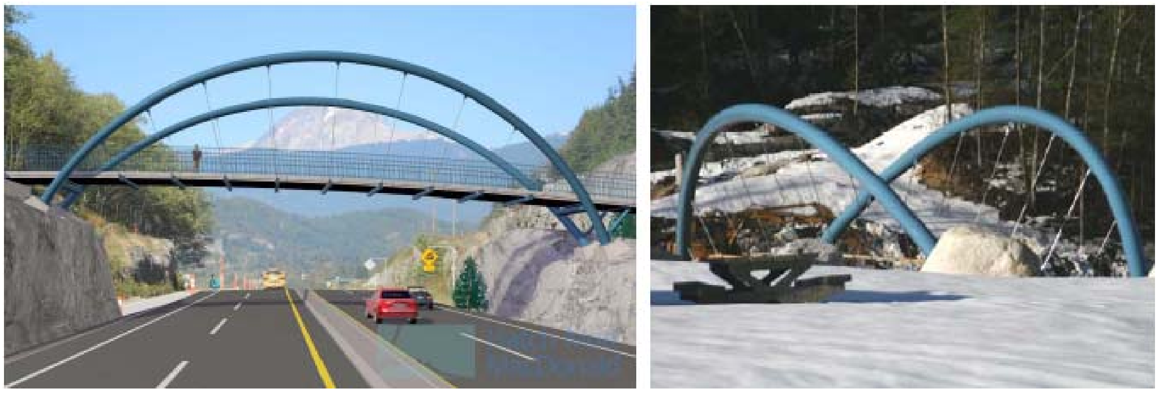


Figure 7.82 Stawamus Chief Bridge: Rendering of the completed bridge (left) and the as-built arches (right).

The splayed-arch design of the bridge coupled with potentially strong coastal winds raised concern for their susceptibility to excessive wind-induced vibration. The bridge designer added Stockbridge dampers internally to the arches to improve the safety of the bridge design, but recognized that testing and evaluation of the as-built structure was the only way to ensure its safe performance.

Outdoor tests, such as those conducted on the Stawamus Chief Pedestrian Bridge and the other deployments presented in this chapter, require that electrical components such as the Imote2 and any accompanying sensor boards have adequate environmental hardening measures to allow deployment in a variety of outdoor exposure conditions. The following factors must be considered in the selection of the appropriate environmental hardening measures:

- Weather conditions at the monitoring site
- Temperature at the monitoring site
- Humidity at the site and within the enclosure
- Precipitation
- Projected life of monitoring systems
- Total size of elements to be contained in enclosure (batteries, sensors, connectors, Imote2, etc.).
- Access (for maintenance or battery replacement)
- Sensor alignment
- Sensor exposure (i.e. pressure sensor to wind flow or light sensor to light source)
- Mounting considerations
- Safety
- Vandalism

Many of the factors listed above are specific to the application and the structure to be monitored. For this research, the enclosure was selected to accommodate The Imote2 and SHM-A sensor board, three D-cell batteries, and an external antenna and provide resistance to most environmental elements.

The selected enclosure is a PVC molded non-metallic junction box that carries a NEMA 6P rating. These enclosures protect against rain and water submersion (see: http://www.ruggedpcreview.com/3_definitions_nema.html).

Figure 7.83 shows the components of the bridge testing. The arches were excited manually by pull-down and horizontal tug tests. Four Imote2s with SHM-A sensor boards, Intel battery boards, 3 D-cell batteries and external antennas were installed at the top of the arches to measure their vibration along three axes in response to the excitation. On each arch, the two nodes were installed next to one another so that the measurements could be compared for verification of the acceleration and the time synchronization/ resampling software. The base station node was connected to a laptop that operated on the bridge deck.

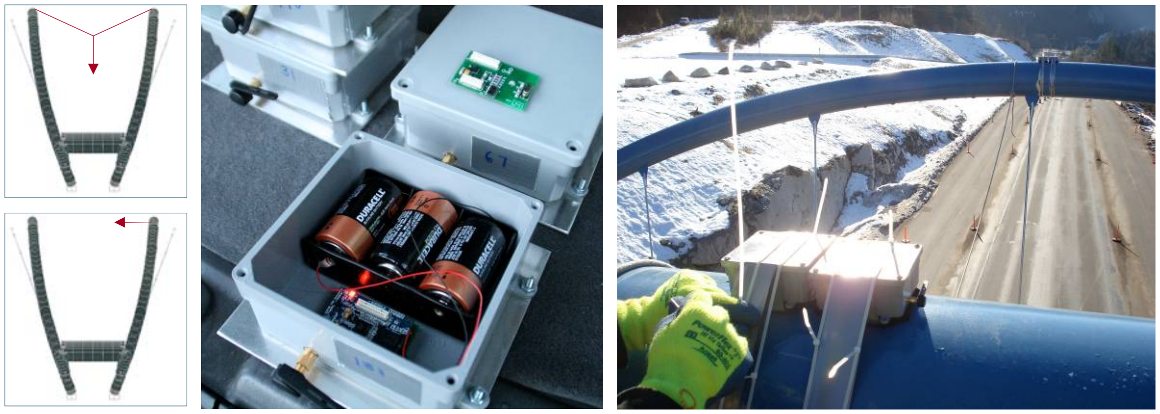


Figure 7.83 Stawamus Chief Pedestrian Bridge testing: Manual excitation (left), hardware (center), sensors installed on bridge arches.

The Imote2 hardware and software validation goals for the bridge tests were to:

- Assess the Imote2 performance in cold temperatures (< 0°C)
- Validate the performance of *RemoteSensing*, including the *WDT*, and synchronized sensing
- Validate the performance of the SHM-A sensor board

Pre-tests conducted at the University of Illinois in a freezer as well as outdoors in cold weather (-10°C) showed that the Imote2 had unreliable performance in colder conditions. One method to counteract the temperature effects is to increase the core voltage of the Imote2 processor (see Chapter 5 for details). Increasing the core voltage from the default value of 0.85V to 1.1V showed promise in the pretests and was implemented for the Stawamus Chief Pedestrian Bridge Tests.

During testing, the temperature at the bridge ranged from -5 to 0°C. In total, 22 60-second tests were conducted with a 50 Hz sample rate. In two cases, the tests did not complete due to incorrect user commands resulting in freezing of one or more of the remote nodes so they were not responsive to radio commands. In these cases, the *WDT* expired after 10 minutes, the nodes reset themselves and the tests could resume without any physical interaction with the nodes. In the other 20 cases, *RemoteSensing* with resampling was successful with excellent data quality and no data loss. Figure 7.84 shows some representative time histories of adjacent sensor nodes. The data shows very good time synchronization and good agreement both lower (< 3mg) and higher amplitude ranges. The small discrepancy between the two signals that is observed in the lower amplitude range is the result of using generic calibration constants (scale and offset) for these tests rather than individually calibrating each sensor.

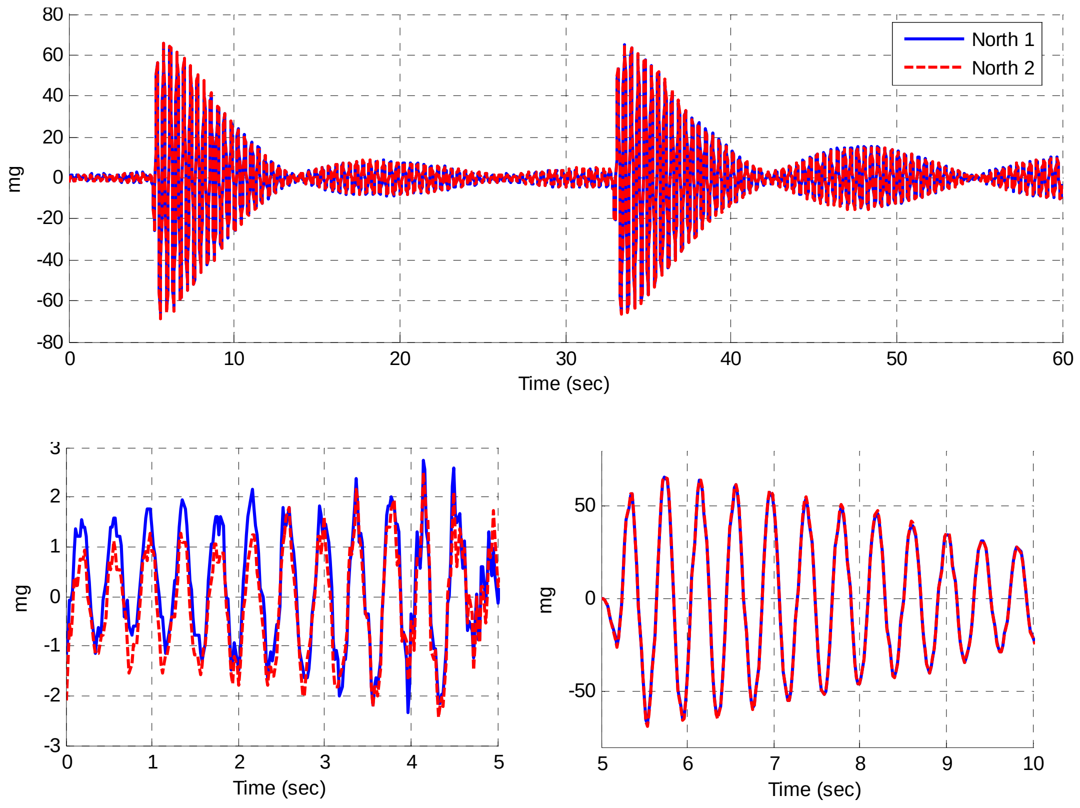


Figure 7.84 Measured acceleration data adjacent sensors during one Stawamus Chief Pedestrian Bridge test.

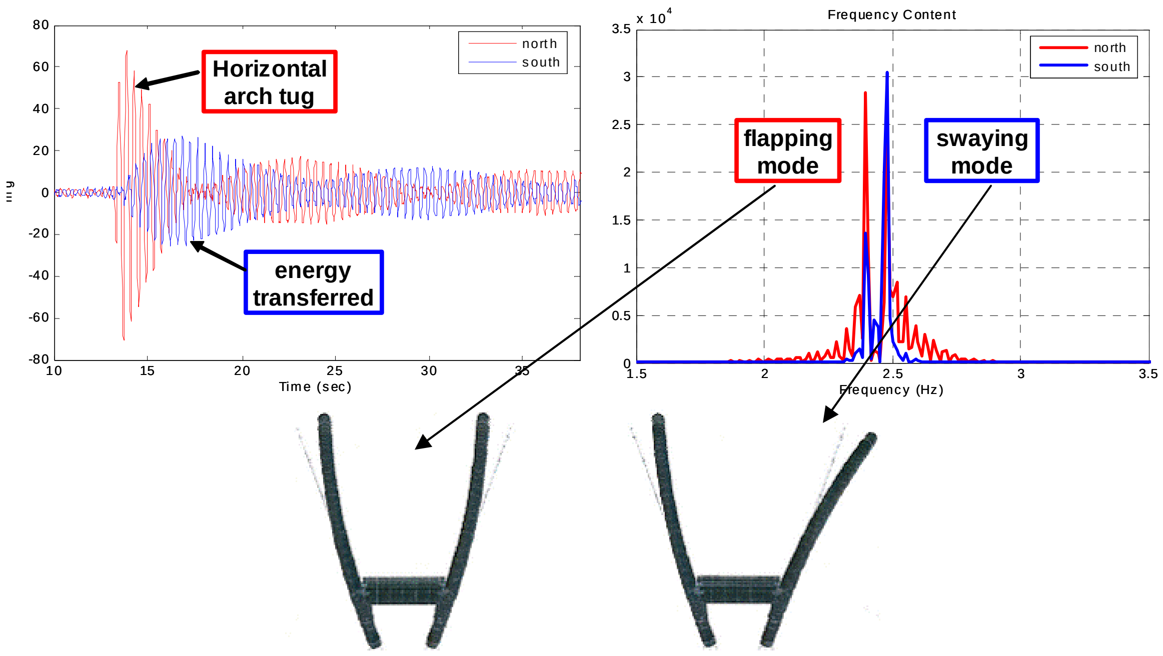


Figure 7.85 Measured response of the Stawamus Chief Pedestrian bridge arches in the time (top left) and frequency domain (top right) and the corresponding mode shapes (below).

The measured response at the top of the arches revealed that the two predominant modes of vibration are closely coupled, resulting in a "beat-phenomenon" response in the time domain as illustrated in Figure 7.85. The phenomenon was particularly evident in tests where one of the arches was subjected to a horizontal impulse load (a single horizontal tug imparted from with a rope attached to the highest point of the arch) and the energy transferred from arch to arch as the vibration during free vibration of the structures. The two modes, shown in Figure 7.85, were identified as a "flapping" mode, where the arches move out of phase with one another, and a "swaying" mode, where the arches move in phase with one another. The transfer of energy between the arches via the tie-beams connecting them reduces the potential for frequency-locking required for excessive resonant vibration.

In all, including setup, sensor installation, damper engagement/disengagement and teardown, the testing took approximately 6 hours. This test demonstrates that, in addition to long-term autonomous monitoring, the Imote2 provides a quick and convenient method for conducting short-term structural testing.

7.2 Siebel Center Staircase

The Thomas M. Siebel Center for Computer Science (Siebel Center) at the University of Illinois provided a test-bed environment for a small deployment of Imote2s to test the long-term stability of the *RemoteSensing* application employing the *SnoozeAlarm* mode. This effort served to inform more extensive, larger-scale deployments by first assessing the network performance on a small scale in a more controllable environment. Six Imote2s were installed on the staircase from the second to third floor of the Siebel Center in the main lobby/atrium area. A base station positioned in an office near the staircase allowed the network to be operated remotely.

The Imote2s were programmed with *RemoteSensing* with *SnoozeAlarm* enabled. The ITS400CA sensor board (Crossbow) was used (the SHM-A sensor board was undergoing revision updates and prototypes were not available at the time of these tests) with the Imote2 in the PVC enclosures discussed in the previous section as shown in Figure 7.86. They sensor nodes were outfitted with external antennas (Titanis 2.4 GHz Antenna) and powered by three D-cell batteries each. Three of the nodes utilized Crossbow battery boards while the other three utilized Intel battery boards. The initial battery voltage for each of the nodes was approximately 4.5V. The *RemoteSensing* and *SnoozeAlarm* parameters used for the tests are shown in Table 7.26.

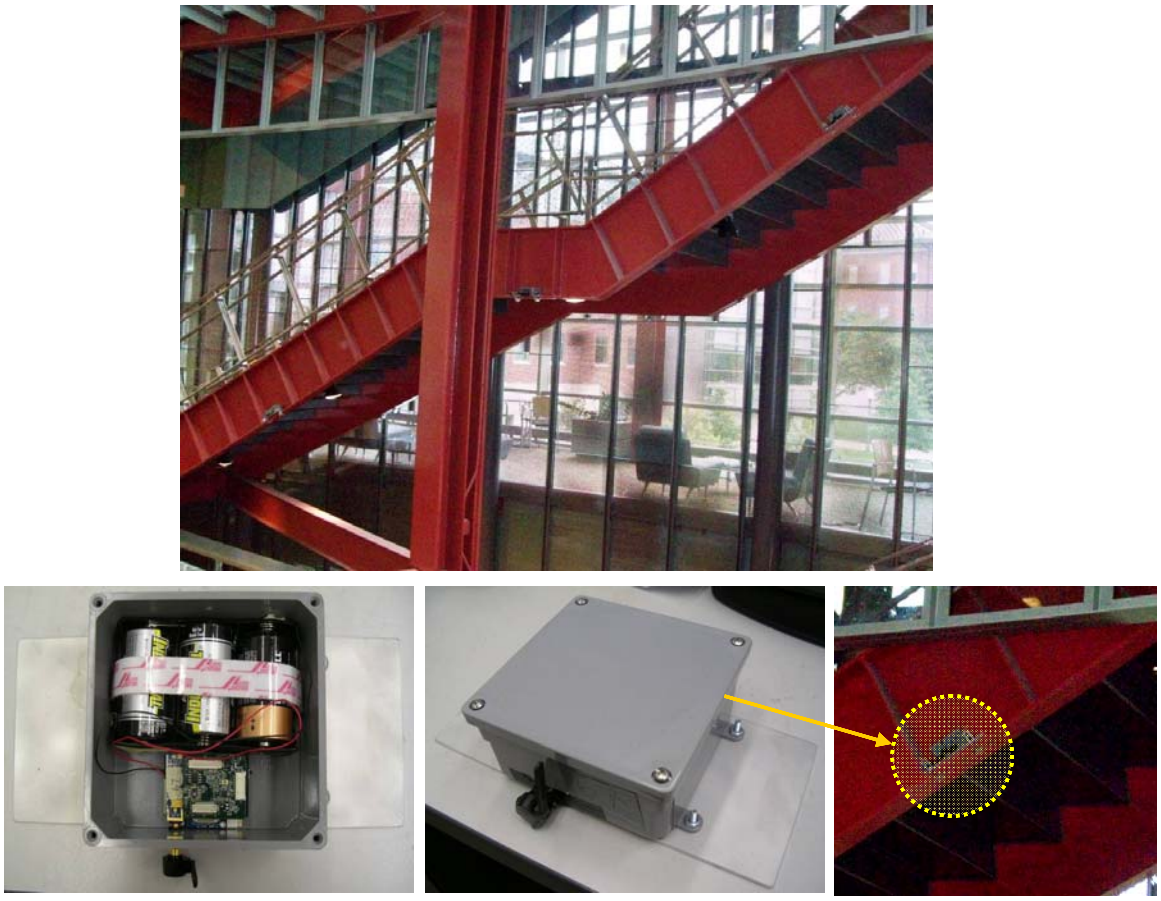


Figure 7.86 Siebel Center staircase (top) with sensor nodes installed (bottom).

Table 7.26 Network and sensing parameters for Siebel Center Staircase tests.

Parameter	Value
Network size, n	6
Number of *RemoteSensing* events per day, n_{RS}	1
Sampling rate, fs	280 Hz
Channels sampled, ch	3
Number of data points, d	10,000
SnoozeAlarm wake/listen time, t_{wake}	500 ms
SnoozeAlarm sleep time, t_{sleep}	10 sec
SnoozeAlarm Duty cycle, δ_{SA}	4.76%

According to the initial battery voltages and the power consumption calculations presented in Chapter 6, the estimated average current and service life for each battery board is given in Table 7.27.

Table 7.27 Siebel Staircase projected power consumption.

Battery Board	Average Current (mA)	Estimated Service Life (days)
Intel	12.59	56
Crossbow	9.67	54

Battery readings were taken remotely via the *RemoteVbat* utility command. This command returns the voltage supplied to the Imote2 processor, which is 0.4V less than the actual battery voltage due to a drop over a diode on the battery board. After one week of testing, the actual battery voltage was reported at ~4.2V on the Crossbow battery board. The readings from the Intel battery boards were constant values (~3.2V) due to the voltage regulator and thus it was not possible to determine the change in voltage over time. After the first week, the battery voltage was recorded periodically on the Crossbow battery boards. The average drop in the battery voltage (after the first week of testing) was approximately 12.3 mV/day. However, the voltage drop in the first week averaged to 0.3V/7days = 42.9 mV/day. These observations are consistent with the typical battery voltage profile versus service life illustrated in Figure 6.31 where there is a steep initial drop-off in the voltage followed by a somewhat linear drain.

The Crossbow battery boards with the ITS400CA sensor boards can run until the batteries reach 3.6V. After the initial drop in the first week, this value is expected to be reached in (4.2V – 3.6V)/0.0123V/day = 50 days. The resulting total expected life based on these initial observations and calculations is 57 days. This projection is fairly consistent with the value estimated using the constant current and battery life predicted using the relationships presented in Chapter 6.

After 37 days, one of the nodes with an Intel battery board was no longer responsive. Because the remainder of the nodes lasted between 52 and 55 days, the relatively short life of the first node to die indicates that there was likely unexpected behavior on this node. The node may have encountered a hang-up where it was in a high power consumption state for an extended period of time, thus more quickly draining its battery. The observed battery life (52 to 57 days) is very consistent with the value predicted by the calculations presented in Chapter 6 as shown in Table 7.27 and the projected life based on early battery readings from the network. This agreement validates the methods presented in Chapter 6 as reasonable means to predict battery life on a network of smart sensors with a variety of hardware and software parameters.

Applying the same battery life projection methodology presented in Chapter 6, the projected battery life of the Siebel Center staircase network utilizing SHM-A sensor boards with the Intel and Crossbow battery boards is 59 and 47 days, respectively. These battery life estimates are similar to those estimated and realized for the ITS400CA sensor boards. Although the SHM-A sensor board draws more current during sensing, its lower current draw in the idle state actually leads to a longer projected life when used in conjunction with the Crossbow battery board.

In terms of the long-term performance of the network deployed on the Siebel Center staircase, no problems were observed with the network or any need to physically access it throughout the duration of its life; all software functioned as intended. On a single occasion in the network's 52-day operation, *RemoteSensing* had to be run twice in one day due to failure of the first attempt. These results verify the robust performance of the *RemoteSensing* application with *SnoozeAlarm* enabled. The only limiting factor on the life of the network was the power supply.

7.3 Jindo Bridge

The software and hardware developed in this research were validated on the Jindo Bridge in South Korea. This deployment is part of a tri-lateral collaboration between South Korea (Korean Advanced Institute of Science and Technology, KAIST), Japan (University of Tokyo) and the USA (University of Illinois at Urbana-Champaign). The purpose of the Jindo Bridge deployment is to demonstrate the suitability of the Imote2 smart sensor platform, the SHM-A sensor board, and the ISHMP software for full-scale, structural health monitoring. The Jindo Bridge (Figure 7.88) is made up of twin cable-stayed bridges that connect Jindo Island to the far southwestern tip of the Korean Peninsula near the town of Haenam (Figure 7.87). The older span finished construction in 1984 and the newer span was completed in 2005. The subject of this study is the newer span (on the left in Figure 7.88).

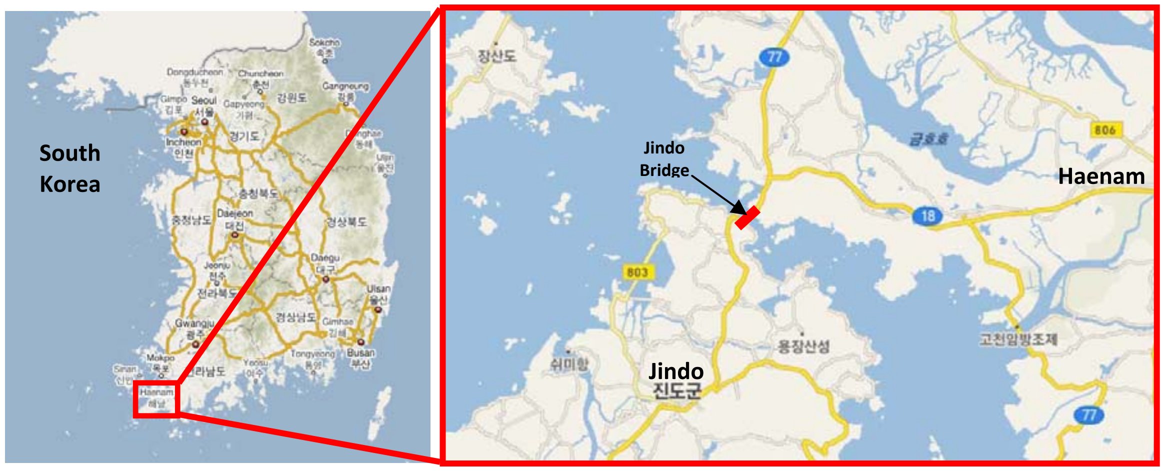


Figure 7.87 Location of the Jindo Bridge on the Korean Peninsula (left), between Jindo and Haenam (right).

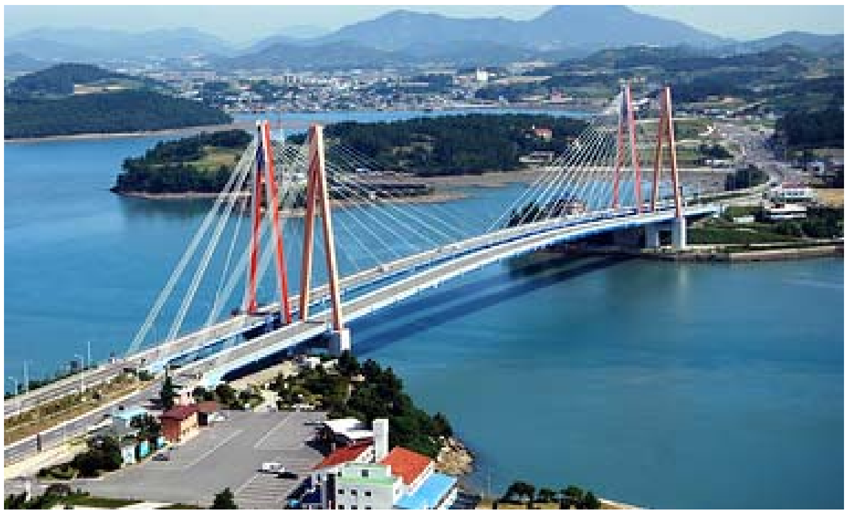


Figure 7.88 Twin spans of Jindo Bridge connecting Jindo Island with the Korean Peninsula with the newer span on the left.

7.3.1 Deployment goals

The primary goal of the Jindo Bridge deployment is to realize the first large-scale, autonomous network of smart sensors for structural health monitoring. This deployment is expected to highlight the challenges and opportunities associated with such a large scale test-bed and thus provide rich information for researchers and engineers interested in achieving a similar SHM system. In terms of the research developed in this report, the primary goals are as follows:

- Validate the performance of the SHM-A sensor board for full-scale testing
- Validate the autonomous network operation software
- Identify the software parameters that are most critical to the functionality of the network in a demanding communication environment
- Assess the time associated with network communication
- Investigate power consumption

7.3.2 Deployment setup

In total, 70 Imote2 sensor nodes with SHM-A sensor boards and Crossbow battery boards have been installed on Jindo Bridge. To achieve a final deployment setup, there will be several phases aimed at addressing difficulties encountered during each phase of the deployment. After sensor installation and communication range tests, the initial data acquisition phase (Phase I) began on June 13, 2009 and after some adjustments a second preliminary phase (Phase II) began on June 23, 2009. In these initial phases, the peer-to-peer network communication is employed meaning each node in the network must be within communication range of the base station. To maintain this communication range and reduce the time to transmit the sensed data back to the base station, the network was divided into two sub-networks (see Figure 7.91). The Jindo side sub-network consists of 37 nodes with 26 nodes on the underside of the box-girder deck (13 on each side, Figure 7.89), three nodes on the pylon, and 8 nodes on the cables. The Haenam side sub-network consists of 33 nodes with 22 nodes on the underside of the box-girder deck (11 on each side), three nodes on the pylon, and 8 nodes on the cables. The base stations are located at the pylons, near the deck on the older Jindo Bridge span in an attempt to achieve more consistent line-of-site communication with nodes on the instrumented span. The main span of the bridge is approximately 344m in length. The distance between each base station and the furthest node in the sub-network (the nodes located at the mid-span of the deck) is 172m. The line-of-sight from the base stations to the nodes on the opposite side of the deck (furthest from the location of the base station nodes on the older span) is obstructed by the deck.

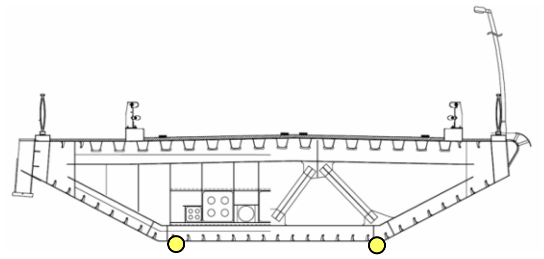


Figure 7.89 Sensor placement on either side of the box-girder deck.

The sensor nodes, pictured in Figure 7.90, are housed in PVC enclosures with 3 D-cell batteries and Antenova Titanis external antennas. The initial battery voltage for each node was 4.7V. The remote nodes were loaded with *RemoteSensing* software with *SnoozeAlarm* enabled and each of the two base station nodes was programmed with the *AutoMonitor* application. The two networks were set up to operate on two different radio channels, 20 and 25, to eliminate potential interference between them to allow for simultaneous operation.

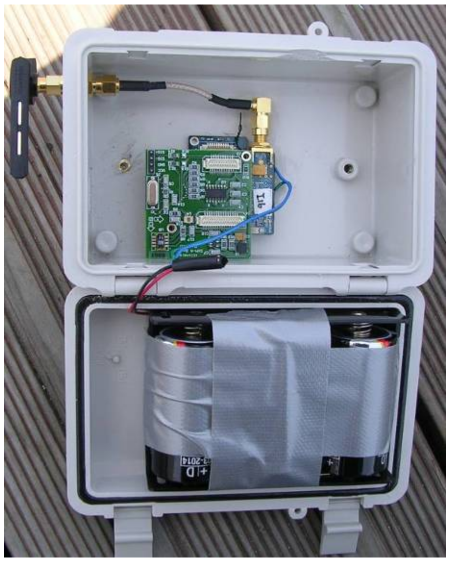


Figure 7.90 Sensor in enclosure with batteries and antenna for Jindo Bridge deployment.

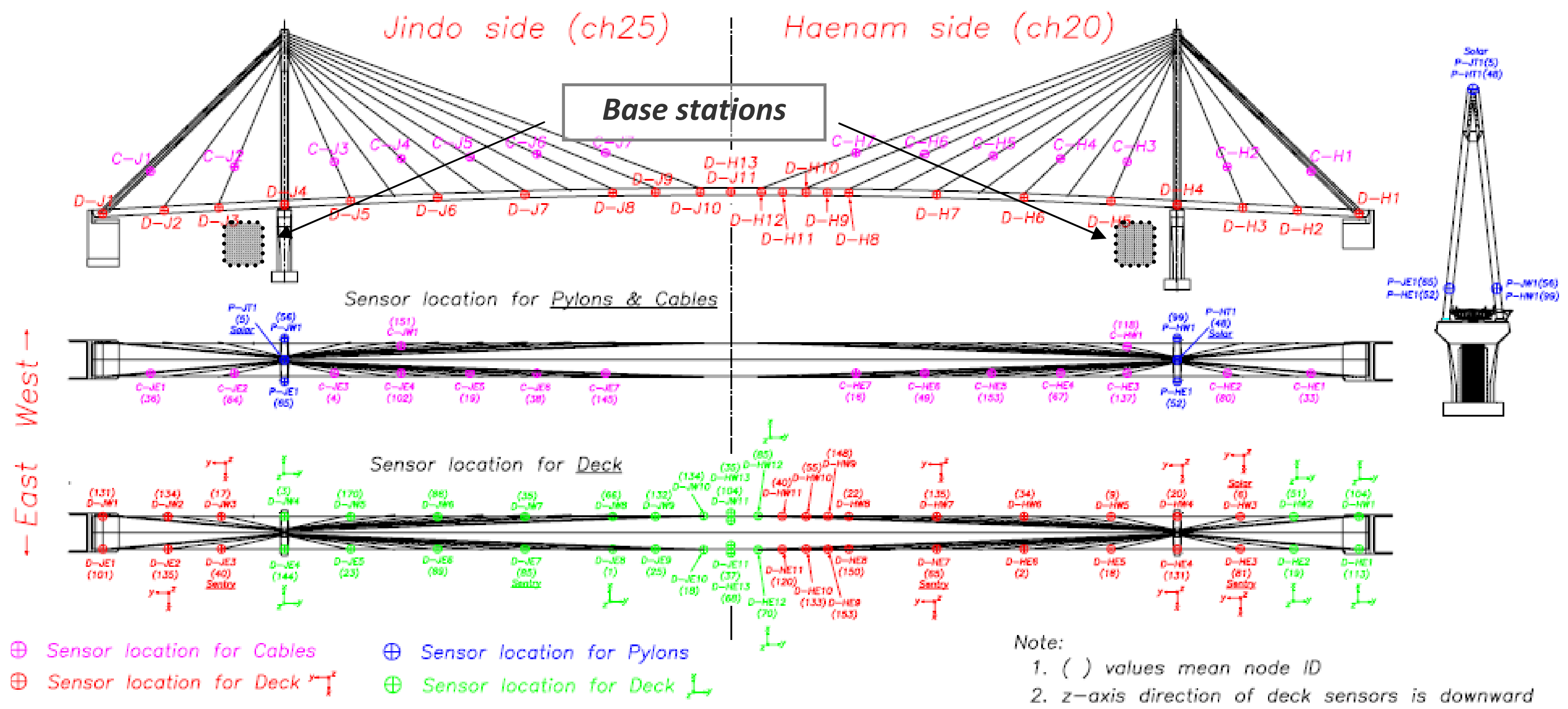

Jindo side (ch25)
Haenam side (ch20)
Base stations
Sensor location for Pylons & Cables
Sensor location for Deck
West
East
Sensor location for Cables
Sensor location for Deck
Sensor location for Pylons
Sensor location for Deck
Note:
1. ( ) values mean node ID
2. z-axis direction of deck sensors is downward

Figure 7.91 Sensor node and base station locations on the Jindo Bridge on the Jindo side (left) and the Haenam side (right).

Phase I

The network parameters (for each sub-network) for the Phase I efforts are given in Table 7.28. Prior to the bridge deployment, these parameters were tested on 70 nodes on the KAIST campus with the nodes placed on tables in close proximity to the base station nodes. The networks performed as expected with no problems. The expectation was that some of the parameters, especially the network time intervals, would require adjustment at the bridge to account for a more demanding communication environment.

Phase I of the testing was focused on deploying the system and establishing the general performance of the network with the parameters given in Table 7.28 and determining those that required adjustment. Although preliminary testing done in advance of the deployment indicated that communication quality between the base station node and each of the nodes in its sub-network would be reasonable at the bridge site, the majority of the challenges encountered in the Phase I tests were associated with the unreliable communication present some of the longer transmission distances, especially with the nodes located on the opposite side of the deck, cables and pylons from the base stations. Also, some variability in the communication ranges of the nodes was evident; those nodes with poorer communication ranges were placed closer to the base station.

Initially, 1000 data points were taken from each sensor channel due to the limitation imposed by the 10-minute Watchdog timer (*WDT*) interval. If the sensor nodes are idle (i.e., do not receive any commands and do not perform any tasks) long enough for the *WDT* to expire, the nodes will reset themselves. When the nodes are reset, they lose any data that has not yet been transferred back to the base station after *RemoteSensing*. For data records longer than 1000 points per channel, the waiting time for nodes to send back data was too long, especially those later in the node order, and the nodes reset prior to sending data. The *WDT* timer interval is an easily adjustable parameter and was been modified in the next phase.

AutoMonitor was initiated on the base station with the two sentry nodes checking data every 20 minutes. For the Phase I tests, the 35-nodes sub-networks were not fully utilized in the application as the initial parameters did not support the lengthy communication times required by the network topology; on the Haenam side 28 nodes were used and on the Jindo side 30 nodes were used. If the 10 mg threshold (given in Table 7.28) was exceeded, the sentry node successfully sent the flag back to the base station which, in turn, would wake the network for *RemoteSensing*. Due communication challenges, not all nodes in the network were able to be woken up before the request timed out. In all attempts at least 25 of the nodes on each side were successfully woken. Once the wakeup time expired, the base station would move on with *RemoteSensing* by broadcasting the sensing parameters to the nodes that were successfully awoken. The broadcast message timed out occasionally (according to the broadcast wait time as shown in Table 7.28) without successfully reaching all of the nodes. If this broadcast message timed-out more than three times, the network was put back into the *ThresholdSentry* mode. If the broadcast message was successful, *RemoteSensing* would run on the network with the data being sent back from each node to the base station. Upon reception of the last node's data, the base station would return to the *ThresholdSentry* mode.

Table 7.28 Jindo Bridge Phase I network parameters

Category	Parameter	Value
Network parameters	Network size	33 (Jindo) 37 (Haenam)
RemoteSensing parameters	Number of *RemoteSensing* events per day	1
	Sampling rate	50 Hz
	Channels sampled	3
	Number of data points	1,000
Network time intervals	Time synchronization wait time	30 sec
	Watchdog timer interval	10 min
	Broadcast message wait time	90 sec
Snooze Alarm parameters	*SnoozeAlarm* wake/listen time	750 ms
	SnoozeAlarm sleep time	15 sec
	SnoozeAlarm duty cycle	4.76%
ThresholdSentry parameters	Sentry network size	2
	ThresholdSentry check interval	20 min
	ThresholdSentry sensing time	10 sec
	Threshold value	10 mg

The time to wake the network and run *RemoteSensing* (initiated by *AutoMonitor* on the base station) was approximately eight to nine minutes for 28 nodes with the sensing parameters listed in Table 7.28). Of this time, approximately 100 seconds were taken to wake the network, up to 90 seconds to send the channel parameters, 30 seconds were taken for network time synchronization, 20 seconds for sensing, with an additional 10 seconds for temperature correction. This leaves over half of the total time devoted to communication of data from the network back to the base station PC for saving to a file. The time it takes each node to transfer its data to the PC consists of two parts: 1) wirelessly communicating the data to the base station node (communication) and 2) transferring the data from the base station node to the PC via the USB serial port (writing). For each node that sends data to the base station approximately 60 percent of the time is devoted to communication while 40 percent of the time is associated with writing.

Throughout the setup process, *AutoMonitor* was periodically stopped to manually run *RemoteSensing*. *AutoMonitor* was allowed to run without interruption was 4 days when manual *RemoteSensing* runs were not required. During this time, the network operated continuously, even with communication difficulties.

The Phase I implementation highlighted the need for parameter optimization but also validated that the *AutoMonitor/ThresholdSentry* software could operate continuously despite sometimes poor network performance. Representative time history plots are given in the following section.

Phase II

After just over a week of operation, researchers returned to the bridge to adjust some of the parameters and install larger 9 dBi antennas on the base station nodes (compared to the 2 dBi on the network nodes) in an attempt to improve the communication performance of the network. The new parameters are shown in Table 7.29. The two changes made to the parameters were the increase in the *WDT* interval to allow for longer data records and the increase in the broadcast message wait time to allow longer time to send the channel parameters to the network before timing out.

Table 7.29 Jindo Bridge Phase II network parameters.

Category	Parameter	Value
Network parameters	Network size	23 to 31
RemoteSensing parameters	Number of *RemoteSensing* events per day	1
	Sampling rate	50 Hz
	Channels sampled	3
	Number of data points	10,000
Network time intervals	Time synchronization wait time	30 sec
	Watchdog timer interval	60 min
	Broadcast message wait time	180 sec
Snooze Alarm parameters	*SnoozeAlarm* wake/listen time	750 ms
	SnoozeAlarm sleep time	15 sec
	SnoozeAlarm duty cycle	4.76%
ThresholdSentry parameters	Sentry network size	2
	ThresholdSentry check interval	20 min
	ThresholdSentry sensing time	10 sec
	Threshold value	10 mg

During the second phase of testing, the base station PC controlling the Jindo side of the bridge was found to be non-responsive, potentially due to overheating. The results presented for the second phase are therefore limited to those received from the Haenam side of the bridge. After the new base station antenna installation and the parameter updates, the network was initially tested to determine which nodes were consistently responsive. While 31 of the 37 nodes consistently woke up successfully, only 23 of the 31 would consistently receive sensing parameters before timing out. The unresponsive nodes are not in one particular location. Of the six nodes that are not responsive to wakeup commands, three are located on the deck, one on the pylon and two on the cables. The unresponsive nodes are at varying distances from the base station but one consistency is that they are on the opposite side of the bridge deck from the base station. Future physical investigation of the nodes may provide more insight on their lack of response. One reason for the discrepancy in the reception of messages during wakeup and parameter broadcast could be in the way different messages are transmitted. As described in Chapter 4, the wake up commands are sent to the nodes in a successive series of unicast *ReliableComm* commands until the network wake up. However, the sending of sensing parameters is done with a *ReliableComm* broadcast message. Future updates to the *ReliableComm* broadcast protocol are expected to improve the reception rate of sensing parameters. Additionally, sending the sensing parameters at the same time as the wakeup command would streamline the initialization of *RemoteSensing* while maximizing the number of responsive nodes while minimizing the time overhead of the application. The communication distances used in this deployment challenge the limits of peer-to-peer network communication. A multi-hop communication protocol is expected to increase the size of the consistently responsive network and improve communication times.

With the increase in the *WDT* interval from ten minutes to one hour, the number of data points acquired could be increased to 10,000 data points sampled at 50 Hz (200 sec.) from 3 acceleration channels for a total of 78 kB of data, including 16-bit timestamps, from each sensor node.

The ambient acceleration levels observed at the Jindo Bridge during wind and traffic excitation ranged from over 30 mg in the vertical direction on the mid-span of the deck to less than 5 mg in the transverse and longitudinal direction on the mid-span of the deck and in all directions on the tower; the vibration levels were easily captured by the SHM-A sensor boards The data was analyzed using Frequency Domain Decomposition (Brincker et al. 2001) to capture the first six modes of the bridge, ranging in frequency from 0.330 Hz to 1.355 Hz. Some sample time histories are shown in Figure 7.92 and Figure 7.93.

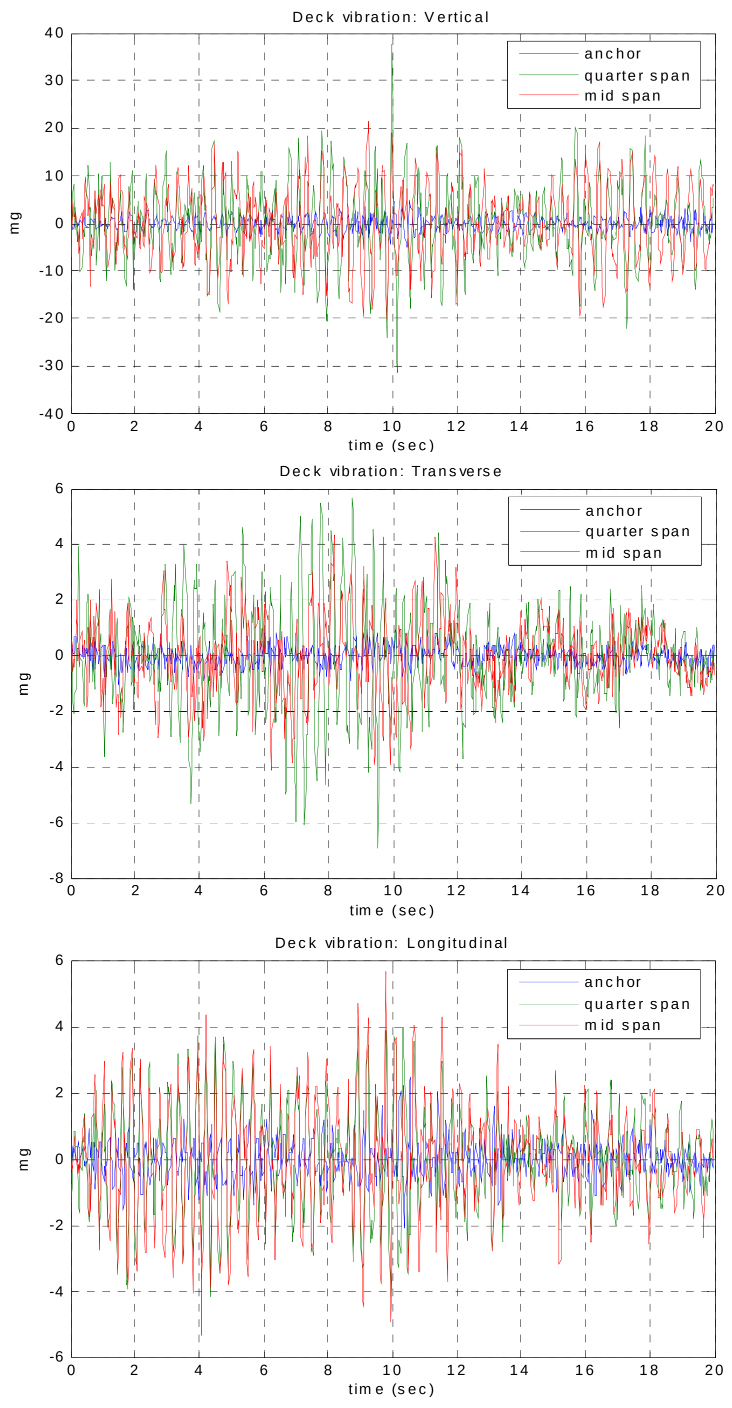
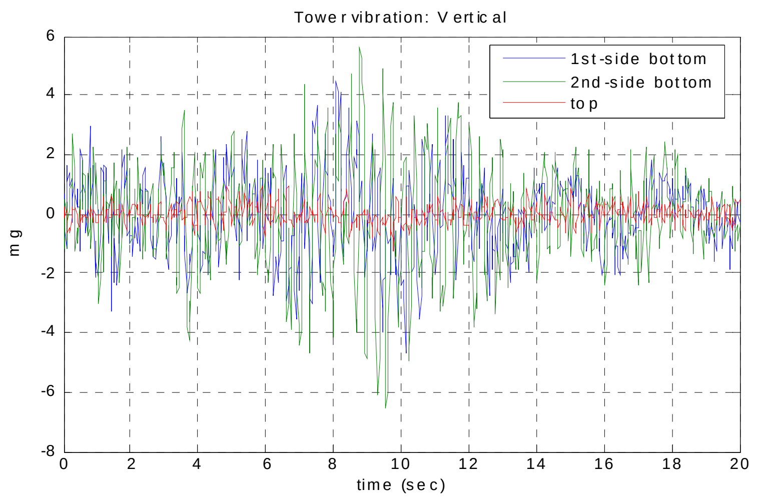
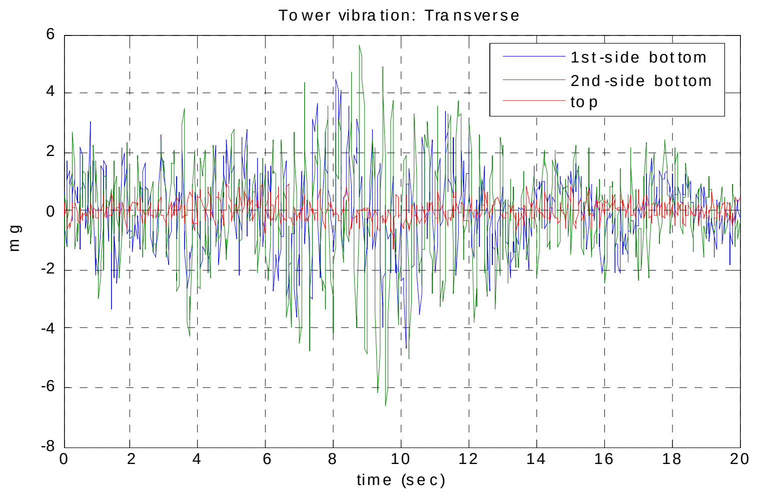


Figure 7.92 Deck acceleration time histories vertical (top), transverse (middle), and longitudinal (bottom).





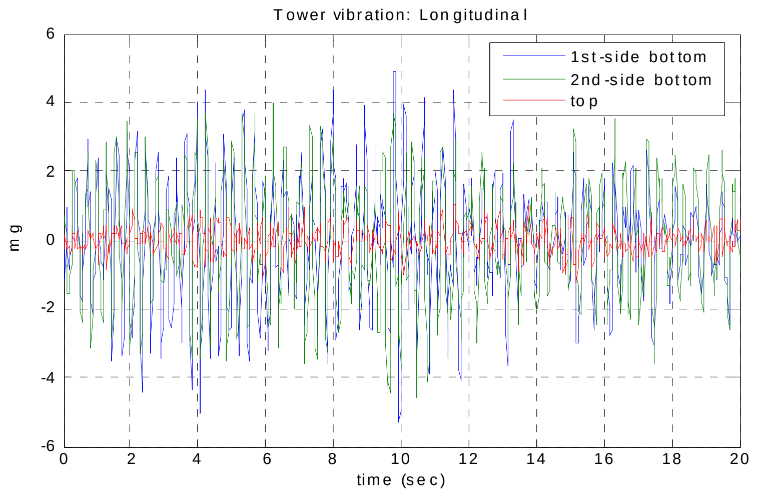
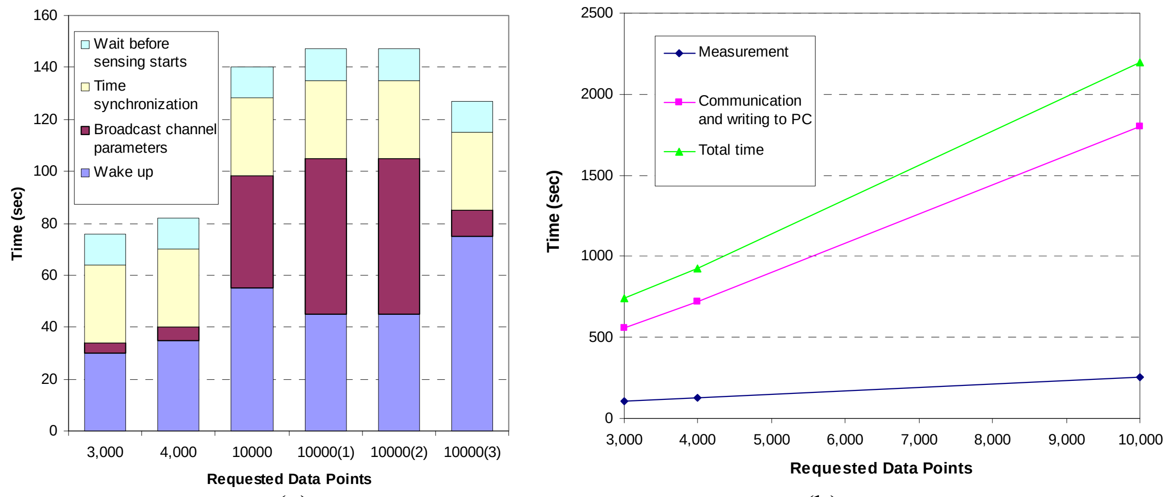


Figure 7.93 Tower acceleration time histories vertical (top), transverse (middle), and longitudinal (bottom).

Table 7.30 shows the recorded times associated with the various phases of *RemoteSensing* for 23 sensor nodes measuring three axes of acceleration at 50 Hz for different requested numbers of samples. The times to wakeup the network and broadcast the channel sensing parameters and transmit the data back to the base station are all dependent on the communication environment, which can vary over the course of the day due to environmental factors and other interference. The other times (the extra wait time before starting sensing, the measurement time and writing the data to the PC) are set fixed in software or by the sensing parameters. The timeout time for waking 23 nodes is 99 seconds; however, for these nodes, the wakeup time ranges from 30 to 75 seconds. The time to broadcast the sensing parameters takes anywhere from just a few seconds up to a minute. Figure 7.94a illustrates the variation in the *RemoteSensing* times prior to the start of data acquisition for five runs, which should be independent of the number of requested samples. Figure 7.94b compares the sensing time, the communication/writing time, and the total application time for different numbers of requested data points, which increase fairly linearly with the number of data point requested. Figure 7.95 shows the *RemoteSensing* times for the total application when 10,000 data points from three channels are sampled at 50 Hz, which takes approximately 30 minutes to complete. With two operational base stations the network would be twice the size however the total application time would not increase because the sub-networks can operate simultaneously; acquiring 10,000 data points for 46 nodes would still only take 30 minutes. As long as the base station nodes operate on different channels (up to 16 are available in the IEEE 802.15.4 2.4 GHz band), multiple base stations nodes can run their sub-networks simultaneously. Therefore, additional base stations could be added to either increase the total network size with no additional cost in time or maintain the total network size while decreasing the time to complete sensing tasks. The latter option will be explored in subsequent phases of the deployment.

Table 7.30 Times associated with *RemoteSensing* on the Jindo Bridge

Network operation time (sec.)	Number of data points (per channel)				
	3,000	4,000	10000	10000	10000
Wake up time	30	35	45	45	75
Broadcast channel parameters	4	5	60	60	10
Time synchronization	30	30	30	30	30
Wait before sensing starts	12	12	12	12	12
Measurement	103	125	257	257	257
Communication and writing to PC	560	720	1827	1827	1745
Total time (min.)	12.32	15.45	37.18	37.18	35.48



(a) (b)

Figure 7.94 (a) *RemoteSensing* times prior to the start of data acquisition (b) and during sensing and communication.

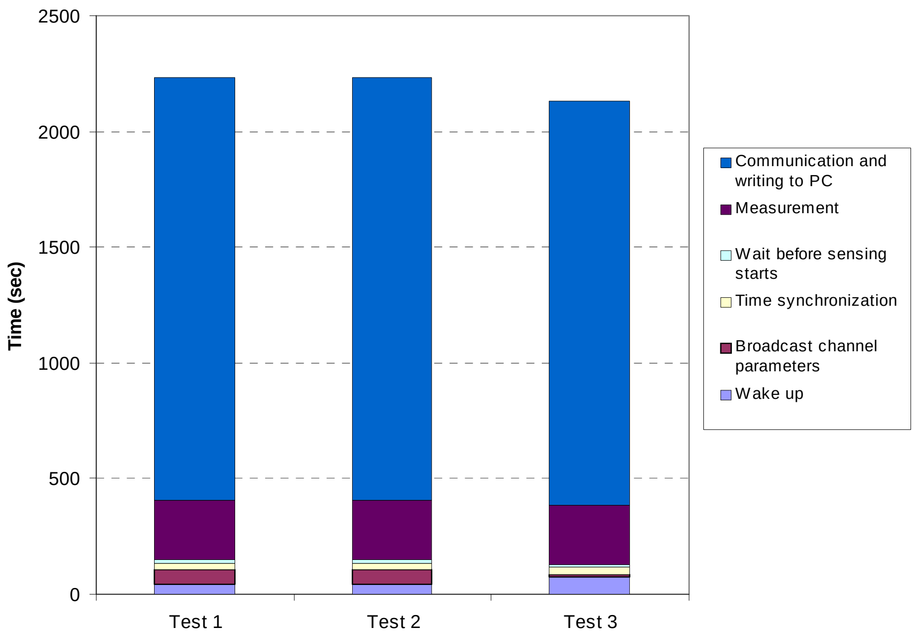


Figure 7.95 Total application times for three runs of 10,000 data points on three channels at 50 Hz.

The battery voltage on the nodes after 18 days of operation ranges from 4.15 to 4.24V. Extrapolation of these voltages to an expected battery life based on this initial drop is not informative as the operation of the network has not been consistent during the network testing and parameters establishment phases of operation. On some days during the initial phases, multiple *RemoteSensing* tests with varying sensing parameters were run to establish the performance of the application. Additionally, some of the nodes that were ultimately left out of the network tests due to communication difficulties caused some of the initial tests to have lengthy completion times. Based on consistent use, from the beginning of deployment, of the on the parameters given in Table 7.29 and initial battery voltages of 4.7V, the estimated battery life of 3 D-cell batteries on the Jindo Bridge would be approximately 74 days. The primary reason that the projected battery life for this deployment is longer than for the Siebel Staircase is the *SnoozeAlarm* sleep interval is increased from 10 to 15 seconds (a 50 percent increase). The result is fewer wake-up events throughout the day thus allowing the node to sleep more and experience fewer of the current spikes associated with emerging from the deep sleep mode. Future deployment will be able to validate this estimate. Without the implementation of the *SnoozeAlarm* functionality, the estimated battery life is just over 7 days for this deployment. *SnoozeAlarm* allows extended WSSN implementations, even with the limited power provided by batteries. Using rechargeable batteries in combination with small solar panels is expected to extend the life of the network considerably and will be investigated through the continued course of the Jindo Bridge deployment.

The tests at the Jindo Bridge are ongoing. A more permanent deployment phase is scheduled for early September, 2009. At that time, the software will be updated to streamline the initiation of *RemoteSensing* events. Additionally, the use of other measures, in addition to vibration levels, will be investigated for triggering the network via *ThresholdSentry*. More base-stations will be added to increase the number of sub-networks operating on the bridge in an effort to decrease application times. The use of multi-hop communication and the addition of multiple solar-powered nodes are also anticipated for the September deployment.

While there are many improvements to be made to the network prior to a more long-term deployment, the initial phases of deployment have satisfied the objectives laid out in the beginning of the section. Specifically the following goals have been achieved:

- The performance of SHM-A sensor board for full-scale testing has been validated
- The robust functionality of the autonomous network operation software has been validated, even with challenging network communication circumstances
- The software parameters have been identified which are most critical to the functionality of the network in a demanding communication environment
- The times associated with network operations have been established
- Preliminary investigation of the power consumption has been carried out

7.4 Full-scale validation summary

The results presented in this chapter demonstrate how each of the components of the WSSN come together to create a full-scale, autonomous deployment. The hardware, software and implementation considerations addressed in this research have enabled these deployments and created a fully-integrated framework to serve as an example for other researchers and engineers. The Stawamus Chief Pedestrian Bridge tests highlight smart sensors' ability to enable quick structural assessment with minimal installation and data acquisition equipment requirements. The results from Jindo Bridge represent the first autonomous, large-scale deployment of a network of smart sensors for structural monitoring. The framework provides the basis for future implementation of distributed data processing, which will further propel smart sensor technology for SHM applications.

Chapter 8

CONCLUSION AND FUTURE STUDIES

8.1 Conclusion

The research presented in this report laid the foundation for the first autonomous implementations of wireless smart sensor networks (WSSN) for full-scale structural health monitoring (SHM). The flexible framework encompasses the necessary hardware, software, and implementation considerations to enable distributed SHM strategies in a wide range of applications. The result of this research is a road map for the broader research community for achieving autonomous, full-scale WSSN applications to improve infrastructure monitoring practices.

Extensive background on structural health monitoring and wireless sensor technology has been given, which indicates the potential of wireless smart sensors (WSS) to dramatically transform SHM practices by providing pertinent information regarding the condition of a structure at a lower cost and higher density than traditional monitoring systems. However, while much of the technology associated with smart sensors has been available for nearly a decade, there have been limited numbers of full-scale autonomous implementations due to the lack of critical hardware and software elements. These elements include modular software for simplified application development and deployment, flexible and accurate sensor hardware to interface with the smart sensor platform being utilized, appropriate communication hardware and configuration, and the integration and validation of each of the system components on a full-scale structure.

To address the complexity of application software development that has limited the widespread use of SHM systems deployed on WSSN, a service-oriented software framework has been presented. This software framework provides modular components that are the common building blocks of many SHM applications that can be linked together to build fully integrated systems. In addition, the software framework contains all of the necessary components to achieve highly synchronized data from a network of smart sensors with no data loss for use by a variety of numerical services. A host of tools and utilities provide the means to evaluate network performance and implement debugging measures to create fully integrated, robust applications. The open-source nature of the software framework, along with detailed instructions and documentation, enable users with a broad range of background and experience the ability to engage in SHM research using smart sensors.

Building upon the foundation of the service oriented architecture for building robust SHM applications, this research addresses the additional software required to achieve autonomous, full-scale, and potentially extended deployments. The challenges associated with such WSSN deployments include effective handling of resources, decision making on the pertinent data to extract from the network, and autonomous management software to control the network for extended periods of time with minimal external interaction. Three applications have been developed and integrated to achieve these goals. A sleep/wake cycle, *SnoozeAlarm*, allows the network to be in a very low power state the majority of the time it is not engaged in SHM applications, while waking periodically to listen for commands from the base station. The power savings associated with this sleep/wake cycle are critical for achieving WSSN deployments beyond just a few days when battery power is utilized. A second service, *ThresholdSentry*, is deployed on a subset of the nodes in the network to alert the base station if measured data is at a level that warrants network-wide action, such as sensing alone or sensing in combination with distributed data processing strategies. This service limits data acquisition and communication to events of interest thus minimizing unnecessary data communication and processing while conserving network resources. The *AutoMonitor* network management service developed in this research provides over-arching and continuous control of network functionality including the establishment of network parameters, the operation of *ThresholdSentry*, the wake up and initialization of network wide synchronized sensing in response to critical loading, and data file generation.

A highly versatile multimetric sensor board designed specifically to address the data quality requirements of SHM applications has been presented. Until now, the sensor hardware typically used with smart sensor platforms has not provided the necessary data quality for such applications. Even in cases where the critical issue of anti-aliasing has been addressed, the solutions have limited the bandwidth of possible measurements and potentially introduced phase and amplitude errors with the use of bulky analog circuitry. The Imote2 sensor board developed and validated in this research provides user-selectable anti-aliasing filter and wide range of accurate sampling rates to achieve high-fidelity data comparable to traditional wired instrumentation. The SHM-A sensor board has three axes of temperature corrected acceleration measurement which can resolve vibration levels to clearly capture response to ambient vibration in many structures. Additional temperature, humidity and light sensors provide more insight into the environmental conditions surrounding the structure that may affect its performance.

The radio and antenna performance of the Imote2 has been to gain an understanding of their transmission characteristics and to evaluate the effects of potential interference. The result of this work has been to identify the optimal configuration for communication in a full-scale network deployment, ultimately enabling a large peer-to-peer deployment of a network of Imote2s in the Jindo Bridge.

A thorough investigation of the power consumption of the Imote2 and accompanying hardware has been presented. The numerical study was based on measured current draws for the various operating modes and hardware combinations used with the Imote2 and revealed that the network events with the greatest impact on power consumption were not necessarily those associated with network-wide sensing and data communication. The power management study highlighted the importance of appropriate hardware and software parameter selection for effectively conserving resources and demonstrated that the effects of hardware that draws higher currents, such as the SHM-A sensor board, can be offset through the implementation of low power states over the majority of the network life. Because many WSSN rely on battery power, the average current draw estimates where translated to battery life projections. Experimental results verified the methodology as an accurate way to project the battery life of the network of smart sensors.

All of the components of the full-scale autonomous framework developed in this research were validated with a series of full-scale implementations. The first deployment, on the Stawamus Chief Pedestrian Bridge in Vancouver, BC, demonstrated the performance of many of the software components that make up the flexible services-oriented software presented in Chapter 3. The built in fault-tolerance methods ensured robust operation of the network. The tests also successfully utilized the SHM-A sensor board and highlighted the suitability of the Imote2 for short-term deployments by providing quick and easy installation, implementation, data acquisition and removal.

The Siebel Center Staircase deployment demonstrated the viability of the Imote2 platform for extended deployments through utilization of the *SnoozeAlarm* service to minimize power consumption. The majority of the nodes lasted almost two months on three D-cell batteries with daily data acquisition while maintaining excellent performance of the embedded software.

Ongoing efforts at the Jindo Bridge in South Korea have validated the autonomous network management software through the deployment of a large-scale network of Imote2 smart sensors employing the SHM-A sensor board. The *AutoMonitor* application has successfully managed the network by running *ThresholdSentry* to trigger network-wide synchronized sensing when it is necessary while limiting unnecessary or undesired data acquisition events. Thus far, the Jindo Bridge deployment has tested the limits of peer-to-peer WSSN implementations with reasonable success and has already provided a wealth of information and insight into the critical issues that are still being addressed as part of the ongoing study. The results from this bridge represent the first autonomous, large-scale deployment of a WSSN for structural monitoring.

The framework presented in this research enables a broad range of SHM applications, from short-term deployments for rapid structural instrumentation and assessment to more complex, extended deployments capable of achieving distributed data processing on a large, autonomous network of smart sensors. This framework combines modular and adaptable software components to facilitate simplified application software development, with versatile, high-fidelity sensor hardware to achieve the data quality required by SHM methods. The implications of hardware and software configurations have been addressed in the context of achieving effective communication and conserving network resources. The components of the framework have been validated in three separate full-scale deployments. Additionally, the appendices of this report provide detailed documentation on the use of both the hardware and software and software that make up the framework presented herein. This research will ensure that wireless smart sensor technology sees more widespread use in SHM applications, ultimately driving the technology forward to improve infrastructure maintenance and enhance public safety.

8.2 Future studies

8.2.1 Software enhancement

Although the modular software framework presented in Chapter 3 represents a large leap toward making smart sensor application development available to a wider audience, users still must provide the control logic to connect the services for their applications. This process could be further improved and simplified by utilizing a graphical user interface (GUI) or drag-and-drop programming environment enabled by dynamic macroprogramming (Mechitov et al. 2007).

The power management study presented in Chapter 6 highlighted the software parameters that have the greatest affect on the power consumption of the network.

Additional optimization efforts could be made in the software parameters to achieve considerable power savings, especially to improve the speed of the node startup. In addition, the parameters associated with the *ReliableComm* protocol (Nagayama and Spencer 2007) could be investigated to improve its efficiency.

The Jindo Bridge deployment brought many issues to light, both in terms of software and hardware. In the software domain, it became clear that multi-hop communication is necessary to achieve more reliable network performance in deployments of that magnitude. Efforts by researchers at the University of Illinois, among others, show promise that multi-hop functionality ensuring reliable communication will be available for the Imote2 shortly. The initial Jindo bridge deployment also demonstrated the need for a simplified sensing initiation process to improve the likelihood that all nodes in the network respond and initiate sensing.

8.2.2 Hardware improvement

Sensor boards

In applications requiring higher accelerometer resolution than that provided by the SHM-A sensor board, a lower-noise accelerometer, such as the SD1221 or the Colybris Siflex as shown in Chapter 2 (Table 2.3), may be used to interface with the same ADC, the Quickfilter QF4A512, utilized in this research. Combining measurements from more expensive, high-resolution sensors with less expensive, lower quality sensors may provide ability to elevate the overall quality of the data acquired from the network while maintaining a low cost per sensor node. Software measures to aggregate the data from the two sensor types would be required to realize this approach.

A strain sensing board may be created as a second board that interfaces with the SHM-A board via the basic connector set, utilizing the single channel analog input currently available. Attractive options for the strain sensing element are a quick-mounting strain transducers such as those available from Bridge Diagnostics Inc. (BDI 2008) or those developed by Professors Socie and Downing in the Mechanical and Industrial Engineering department at the University of Illinois at Urbana-Champaign. This strain transducer has been used for several successful railroad fatigue monitoring applications (Peltier et al. 2007**).** The analog strain voltage would be converted by the QF4A512 in the same way as the accelerometer channel however due to the flexibility of the QF4A512, a different gain, sampling rate and digital filter may be applied for the strain channel.

Temperature correction of the mean value of the accelerometer output was presented in Chapter 5. Similar correction to the actual temperature readings should be investigated so that the output temperature readings represent the ambient temperature in the vicinity of the sensor and not the temperature augmented by the self-heating of the hardware components. The challenges associated with determining a consistent relationship between the actual ambient temperature and the temperature sensor reading would have to be addressed for the correction to be implemented in the sensor board driver.

The flexible nature of the Imote2 sensor interface allows the design of a wide range of sensor boards that can interface with it. For example, the I2C digital interface can accommodate many sensors on a single digital bus, allowing a highly multimetric sensor board design. Some of the sensing options that could be explored include:

- Pressure (for wind speed measurements)
- Air/water quality
- GPS
- Electrical resistance for corrosion monitoring

With the sensor software framework in place, these signals could easily be captured in a synchronized manner, along with the existing sensors, to gain a complete picture of the environmental and structural conditions. Additionally, the output from a range of sensors could be incorporated into the threshold detection and triggering software developed in this research.

Power interface

The power consumption study conducted in Chapter 6 illustrated the benefits and drawbacks of two types of battery boards. One battery board (the Intel battery board) utilized a buck-boost voltage regulator in its design. The advantage of this design is that more of the range of the batteries may be used; the drawback is the idle power consumed by such hardware. The Crossbow battery board does not incorporate a voltage regulator/booster and is therefore limited to using less of the battery's capacity; however, it consumes less power in an idle state. A power interface board could be designed to improve the power efficiency in systems such as the one presented in this research that utilize sleep/wake cycles with intermittent periods of high current draw states. This battery board could have a voltage regulator that is only active when battery voltage is below a certain threshold. The result would be less current in the idles state when the battery voltages are reasonable, while utilizing more of the battery capacity as it nears the end of its usable life.

8.2.3 Full-scale deployments

The full-scale deployments presented in this research illustrate how they can inform the advancement of WSSN for SHM. More deployments with different structural, loading, communication, environmental, and as yet unknown conditions should be sought out to continue to drive the research in this area.

Using additional base stations to create multiple subnets has the potential to improve the scalability of the framework presented in this report by keeping communication times reasonable while still achieving synchronized sensing from a large network of sensors. Because these sub-networks can operate simultaneously on different channels within the IEEE 802.15.4 band, up to 16 sub-networks could be achieved (assuming no other interference is present in the 2.4 GHz band).

Solar power used in conjunction with rechargeable batteries provides a promising solution for achieving a more long-term WSSN deployment that is supported by the framework developed in this research. Preliminary solar powered Imote2 nodes have been deployed by researches at the Jindo Bridge with promising results. Further investigation is required for this technology and other energy harvesting techniques, such as wind power and the conversion of structural vibrations into usable energy.

Continued efforts in employing distributed SHM strategies on WSSNs is required. To truly achieve scalable systems, effective data aggregation and processing methods must be proven in large-scale deployments. Ultimately, the successful implementation of distributed damage detection in full-scale deployments has the potential to truly transform monitoring practices.

8.2.4 Rare event monitoring

Effective SHM systems should have the ability to capture extreme events, such as earthquakes or bridge overloads. The current framework does not support real-time triggering of network-wide sensing in reaction to a short-term, high structural response levels. The primary limitations are the time it takes to wake the network (when it is in the *SnoozeAlarm* state) and the time associated with network time synchronization. In a large network, such as the one deployed on Jindo Bridge, it can take over one minute from a sentry node sending an alert to the actual start of data acquisition. In the event of an earthquake the entire occurrence could be missed while the network is initiated. Solutions to achieve the ability to capture such an event will likely require an integrated hardware/software approach. For example, the Imote2 possesses an alarm pin that causes the node to power up if the pin receives a signal. To take advantage of this alarm pin to wake the node, the sensor modules could incorporate very low-power sensor in combination with a very basic, low-power microcontroller that are on a separate power supply than the rest of the module. This separate alert system would remain powered at all times and send a signal to the alarm pin when a particular threshold is exceeded. Upon waking, the primary sensor node would proceed to acquire timestamped data. Time synchronization could be implemented after data acquisition is complete, thus allowing the timestamps to be corrected to allow the data to be resampled, thereby achieving synchronized data record in response to an extreme loading event. The additional power consumption and the logic required pose the greatest challenges to such a system.

8.2.5 Multi-functional WSSN

Once networks of wireless smart sensors are in place within a structure they can be leveraged for other complementary purposes in addition SHM. For example, sensors deployed on a bridge could provide insight into traffic flow to aid in congestion mitigation efforts and provide insight to first-responders following an emergency. Sensors installed in buildings could be used to measure environmental factors such as temperature, humidity, indoor air quality. These multi-functional systems can enhance quality of life while lowering operation costs in addition of addition to measuring structural performance to improve public safety.

REFERENCES

Abdel-Ghaffar, A. M., and Scanlan, R. H. (1985). Ambient vibration studies of golden gate bridge: I. suspended structure. *J. of Engineering Mechanics, 111*(4), 463-482.

Adler, R., Flanigan, M., Huang, J., Kling, R., Kushalnagar, N., Nachman, L., et al. (2005). Intel mote 2: An advanced platform for demanding sensor network applications. *Proc. of the 3rd International Conference on Embedded Networked Sensor Systems,* 298-298.

Ali, M., Dunkels, A., Römer, K., Langendoen, K., Polastre, J., and Uzmi, Z. A. (2006). Medium access control issues in sensor networks. *ACM SIGCOMM Computer Communication Review, 36*(2), 33-36.

Analog Devices, Inc., “High Common-Mode Voltage, Gain Difference Amplifier AD628,” Norwood, MA (2007).

Arms, S. W., Galbreath, J. H., Newhard, A. T., and Townsend, C. P. (2004). Remotely reprogrammable sensors for structural health monitoring. *Smart Materials Technology: NDE/NDT for Highways and Bridges,*

Bar-Cohen, Y. (1999). In-service NDE of aerospace structures--emerging technologies and challenges at the end of the 2nd millennium. *Materials Evaluation, American Society for Nondestructive Testing*

Bernal, D. (2002). “Load vectors for damage localization.” *J. of Engineering Mechanics*, 128(1), 7-14.

Bridge Diagnostics, Inc., (2008). Boulder, CO. http://www.bridgetest.com.

Brincker, R., Zhang, L., and Anderson, P. (2001) Modal identification of output-only systems using frequency domain decomposition. *Smart Mater. Struct.*, 10:3, 441–445.

Brownjohn, J. M. W. (2007). Structural health monitoring of civil infrastructure. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365, 589-622.

Buonadonna P, Hill J, Culler D. (2002). Active Message Communication for Tiny Networked Sensors, www.tinyos.net/papers/ammote.pdf.

Caicedo, J. M., Clayton, E., Dyke, S. J., Abe, M., and Tokyo, J. (2002). Structural health monitoring for large structures using ambient vibrations. *Proc. of the ICANCEER Conference,* Hong Kong, August 15-20.

Castaneda, N., Sun, F., Dyke, S., Lu, C., Hope, A., Nagayama, T. (2008). Implementation of a Correlation-based Decentralized Damage Detection Method Using Wireless Sensors. *Proc. 2008 ASEM Conference*, Jeju, Korea.

Çelebi, M. (2002). Seismic instrumentation of buildings (with emphasis on federal buildings). Report no.0-7460-68170, United States Geological Survey (USGS), Çelebi, M., and EERI, M. (2006). Real-time seismic monitoring of the new cape girardeau bridge and preliminary analyses of recorded data: An overview. *Earthquake Spectra*, 22, 609.

Çelebi, M., Purvis, R., Hartnagel, B., Gupta, S., Clogston, P., Yen, P., et al. (2004). Seismic instrumentation of the Bill Emerson Memorial Mississippi River Bridge at Cape Girardeau (MO): A cooperative effort. *Proc. of 4th Int'l Seismic Highway Conf.,* Memphis, TN.

Chang K. (2002). "RF and Microwave Wireless Systems". Wiley and Sons, New York.

Chintalapudi, K., Paek, J., Gnawali, O., Fu, T., Dantu, K., Caffrey, J., et al. (2006). Structural damage detection and localization using NetSHM. *Proc. 5th International Conference on Information Processing in Sensor Networks: Special Track on Sensor Platform Tools and Design Methods for Networked Embedded Systems,*

Chipcon AS SmartRF CC2420 Preliminary Data Sheet (rev 1.2). June 9, 2004.

Citizen Component Sales Div., HCM49, Torrance, CA (2007).

Colibrys Inc., "Si-Flex SF1500S Accelerometer," Neuchatel, Switzerland (2007).

Crossbow Technology, Inc, "MICA2 Wireless Measurement System," San Jose, CA (2007).

Crossbow Technology, Inc., ITS400, Imote2 Basic Sensor Board, San Jose, CA, (2007).

Djordjevic, B. B. (1990). NDE in space. *Proc. Ultrasonics Symposium, IEEE* , 997-1002.

Doebling, S. W., Farrar, C. R., Prime, M. B., and Shevitz, D. W. (1996). Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristics: A literature review, LA--13070-MS, Los Alamos National Lab., NM.

Doebling, S.W. and Farrar, C.R. (1999). The state of the art in structural identification of constructed facilities, A report by American Society of Civil Engineers Committee on Structural Identification of Constructed Facilities.

Doebling, S.W., Farrar, C.R., and Prime, M.B. (1998). A summary review of vibration-based damage identification methods. *Shock Vib. Dig.*, 30(2), 91-105.

Elson, J., Girod, L., and Estrin, D. (2002). Fine-grained network time synchronization using reference broadcasts. *Proc. 5th Symposium on Operating Systems Design and Implementation*, Boston, MA.

Ember (2008). Boston, MA

Energizer Battery Manufacturing, Inc. (2008) Alkaline Manganese Dioxide: Handbook and Application Manual, St. Louis, MO.

Energizer Battery Manufacturing, Inc. (2009) Energizer EN95 Product Datasheet, St. Louis, MO.

Farrar, C. R., and Worden, K. (2007). An introduction to structural health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1851), 303-315.

Farrar, C. R., Doebling, S. W., Cornwell, P. J., and Straser, E. G. (1997). Variability of modal parameters measured on the Alamosa Canyon Bridge. *SPIE Proceedings,* 257-263.

Farrar, C.R. and Doebling, S.W. (1997). Vibration-based health monitoring and model refinement of civil engineering structures, *Proc. 1st International Architectural Surety Conference*.

FHWA National Bridge Inventory 2007, http://www.fhwa.dot.gov/BRIDGE/nbi.htm

Ganeriwal, S., Kumar, R., and Srivastava, M. B. (2003). Timing-sync protocol for sensor networks. *Proc. 1st International Conference on Embedded Networked Sensor Systems*, Los Angeles, CA, 138-149.

Gao, Y. (2005). Structural health monitoring strategies for smart sensor networks, Doctoral dissertation, University of Illinois at Urbana-Champaign, 2005.

Gao, Y. and Spencer Jr., B.F. (2008). Structural health monitoring strategies for smart sensor networks, *NSEL Report, Series 011*, University of Illinois at Urbana-Champaign, http://hdl.handle.net/2142/8802.

Gu, T., Pung, H.K. and Zhang, D.Q. (2005) A service-oriented middleware for building context-aware services. *J. Network and Computer Applications* **28:1**, 1-18.

Hogenauer, E. B. (1981). An economical class of digital filters for decimation and interpolation. *IEEE Transactions on Acoustics, Speech and Signal Processing* 29(2), 155-162.

Hoult, N. A., Fidler, P. R. A., Middleton, C. R., and Hill, P. G. (2008). Wireless structural health monitoring at the Humber Bridge UK, *Bridge Engineering* 161(4), 189 – 195.

Hubler T. (2005), Worry-free Wireless Networks, *Networked Controls*, Penton Media.

Intel Corporation Research (2005), Intel Mote2 Overview, Version 3.0, Santa Clara, CA.

James, G. H., Carne, T. G., and Lauffer, J. P. (1993). The natural excitation technique for modal parameter extraction from operating wind turbine, Report No. SAND92-1666, UC-261, Sandia National Laboratories.

Jang, S., Rice, J.A., Spencer Jr., B.F. (2009) Structural monitoring of a historic truss bridge using smart sensor network, *Proc. 5th Int. Workshop on Advanced Smart Materials and Smart Structure Technology, ANCRiSST 2009*, Boston, MA.

Juang, J.-N. and Pappa, R. S. (1985). An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics* **8:5**, 620-627.

Kijewski-Correa, T., Haenggi, M., and Antsaklis, P. (2006). Wireless sensor networks for structural health monitoring: A multi-scale approach. *17th Analysis and Computation Specialty Conference, 2006 ASCE Structures Congress,* St. Louis, MO.

Kim S., Pakzad S., Culler D., Demmel J., Fenves G., Glaser S. and Turon, M. (2007). Health monitoring of civil infrastructures using wireless sensor networks. *Proc. 6th International Conference on Information Processing in Sensor Networks*, 254-263.

Kling, R. M. (2003). Intel mote: An enhanced sensor network node. *Proc. of the International Workshop on Advanced Sensors, Structural Health Monitoring, and Smart Structures*,

Kling, R., Adler, R., Huang, J., Hummel, V., and Nachman, L. (2005). Intel mote-based senor networks. *Structural Control and Health Monitoring, 12*, 469-479.

Kottapalli, V. A., Kiremidjian, A. S., Lynch, J. P., Carryer, E., Kenny, T. W., Law, K. H., et al. (2003). Two-tiered wireless sensor network architecture for structural health monitoring. *Proc. of SPIE, , 5057* 8-19.

Kurata, N., Saruwatari, S. and Morikawa, H. (2006). Ubiquitous Structural Monitoring using Wireless Sensor Networks, *Proc. ISPACS*.

Lee K, Chanson S. (2002). Packet Loss Probability for Real-Time Wireless Communications, *IEEE Transactions on Vehicular Technology*, 51(6), 1569-1575.

Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., and Culler, D. (2005), TinyOS: An Operating System for Sensor Networks. Ambient Intelligence, Weber, W., Rabaey, J.M., Aarts, E., Eds. 115-148, Springer, Berlin, Heidelberg.

Linderman, L.E., Rice, J.A., Barot, S., Spencer Jr., B.F., and Bernhard, J.T. (2009). Characterization of Wireless Smart Sensor Performance, *ASCE J. of Eng. Mech*, (in submission)

Liu, J. and Zhao, F. (2005). Towards semantic services for sensor-rich information systems. *Proc. International Workshop on Broadband Advanced Sensor Networks*.

Lynch, J. P., and Loh, K. J. (2006). A summary review of wireless sensors and sensor networks for structural health monitoring. *The Shock and Vibration Digest, 38*(2), 91-28.

Lynch, J. P., Sundararajan, A., Law, K. H., Kiremidjian, A. S., Carryer, E., Sohn, H., Farrar, C. R., (2003). Field validation of a wireless structural monitoring system on the Alamosa Canyon Bridge, *Proc. SPIE Smart Structures and Materials*, San Diego, CA.

Lynch, J. P., Wang, Y., Law, K. H., Yi, J., Lee, C. G., and Yun, C. B. (2005). Validation of a large-scale wireless structural monitoring system on the Geumdang Bridge. *Proc. 2nd International Conference on Embedded Networked Sensor Systems,* Rome, Italy.

Lynch, J. P., Wang, Y., Loh, K.J., Yi, J., and Yun, C.B. (2006) Wireless Structural Monitoring of the Geumdang Bridge using Resolution Enhancing Signal Conditioning, *Proc. of the 24th International Modal Analysis Conference (IMAC XXIV),* St. Louis, MO, January 30 - February 2, 2006.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. (2002). Wireless sensor networks for habitat monitoring. *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications,* 88-97.

Maroti, M. Kusy, B., Simon, G., and Ledeczi, A. (2004). The flooding time synchronization protocol. *Proc. 2nd International Conference on Embedded Networked Sensor Systems*, Baltimore, MD, 39-49.

Mascarenas, D., Flynn, E.; Todd, M., Park, G. and Farrar, C. (2008) Wireless Sensor Technologies for Monitoring Civil Structures. *Sound and Vibration* 42(4), 16-20.

Mechitov, K., Kim, W., Agha, G., and Nagayama, T. (2004). High-Frequency Distributed Sensing for Structure Monitoring, *Proc. First Intl. Workshop on Networked Sensing Systems, Tokyo*, Japan, 101–105.

Mechitov, K., Ravazi, R., and Agha, G. (2007). Architecture Design Principles to Support Adaptive Service Orchestration in WSN Applications, *ACM SIGBED Review*, 4(3).

MetaGeek, LLC. (2008) Boise, ID., http://www.metageek.net.

MicroStrain (2008). Williston, VT, http://www.microstrain.com.

Millennial Net, Inc. (2008). Chelmsford, MA, http://millennial.net.

Murphy, N. (2000) Watchdog Timers, *Embedded Systems Programming*, November 2000, pp 112.

Nagayama, T. and Spencer Jr., B.F., (2007). Structural health monitoring using smart sensors, *NSEL Report, Series 001*, University of Illinois at Urbana-Champaign, http://hdl.handle.net/2142/3521.

Nagayama, T., Rice, J. A., and Spencer Jr., B. F. (2006) "Efficacy of Intel's Imote2 wireless sensor platform for structural health monitoring applications," *Proc. Asia-Pacific Workshop on Structural health Monitoring*, Yokohama, Japan.

Nagayama, T., Ruiz-Sandoval, M., Spencer Jr., B.F., Mechitov, K. and Agha, G. (2004) Wireless Strain Sensor Development for Civil Infrastructure, *First International Workshop on Networked Sensing Systems (INSS)*, pp 97-100.

Nagayama, T., Sim, S-H., Miyamori, Y., and Spencer Jr., B.F. (2007) Issues in structural health monitoring employing smart sensors, *Smart Structures and Systems*, **3:3**, 299-320.

Nagayama, T., Spencer Jr., B.F., Mechitov, K., Agha, G. (2008). Middleware services for structural health monitoring using smart sensors, *Smart Structures and Systems*, in press.

Nagayama, T., Spencer Jr., B.F., Rice, J.A. and Agha, G. (2007) Smart Sensing Technology: A New Paradigm for Structural Health Monitoring, *Proc., 39th Joint Meeting of the US-Japan Joint Panel on Wind and Seismic Effects, UJNR.*

Pakzad, S and Fenves, G. (2004). Structural health monitoring applications using MEMS sensor networks, *Proc. 4th International Workshop on Structural Control, Columbia University*, New York, 47-56.

Pakzad, S. (2008). Statistical Approach to Structural Monitoring Using Scalable Wireless Sensor Networks, Ph.D. Dissertation, U.C. Berkeley, Berkeley, California.

Pakzad, S.N., Fenves, G.L., Kim, S., Culler, D.E. (2008), "Design and Implementation of Scalable Wireless Sensor Network for Structural Monitoring", *ASCE Journal of Infrastructure Engineering*, March 2008.

Paradiso, J.A. and Starner, T. (2005) “Energy Scavenging for Mobile and Wireless Electronics” *IEEE Pervasive Computing* 4(1), 18-27.

PCB Piezotronics Inc. “Model 3701G3FA3G Capacative Accelerometer Installation and Operating Manual,” Depew, NY (2007).

Pei J.S., Kapoor, C, Graves-Abe, T.L., Sugeng, Y.P., and Lynch, J.P. (2007). An Experimental Investigation of the Data Delivery Performance of a Wireless Sensing Unit Designed for Structural Health Monitoring” *Structural Control and Health Monitoring*.

Peltier, D., Barkan, C.P.L, Downing, S., and Socie, D. (2007) Measuring degradation of bonded insulated rail joints, *Proc. AREMA Annual Conference*, Chicago IL.

Polastre, J., Szewczyk, R., and Culler, D. (2005). Telos: Enabling Ultra-Low Power Wireless Research. *Proc. Fourth Int. Symposium on Information Processing in Sensor Networks*. Los Angeles, California.

Quanser Inc., “Quanser Shake Table II, Product Information Sheet,” Ontario, Canada (2006).

Quickfilter Technologies, Inc., (2005). “QFAN006 Application Note: Power Optimization for the QF4A512 Programmable Signal Conditioner,” Allen, TX.

Quickfilter Technologies, Inc., (2007a) “QF4A512 4-Channel Programmable Signal Conditioner,” Allen, TX.

Quickfilter Technologies, Inc., (2007b) Quickfilter Pro Software, Allen, TX.

Razavi, R., Mechitov, K., Agha, G. and Perrot, J.-F. (2007). Ambiance: A Mobile Agent Platform for End-User Programmable Ambient Systems. *Advances in Ambient Intelligence, Frontiers in Artificial Intelligence and Applications* **164**, 81-106.

Rice, J.A., Spencer Jr., B.F., Mechitov, K. and Agha, G. (2008) Flexible Smart Sensing Framework for Structural Health Monitoring, *Proc. US-Korea Workshop on Smart Structures Technology.*

Roundy, S., Wright, P.K. and Rabaey, J.M., (2004) “Energy Scavenging for Wireless Sensor Networks with Special Focus on Vibrations,” Springer.

Ruiz-Sandoval, M., Spencer, B.F. and Kurata, N. (2003). Development of a High Sensitivity Accelerometer for the Mica Platform. *Proc. of the 4th International Workshop on Structural Health Monitoring*, Stanford, CA.

Salawu, O. S. (1997). Detection of structural damage through changes in frequency: A review. *Engineering Structures, 19*(9), 718-723.

Salawu, O. S., and Williams, C. (1995). Review of full-scale dynamic testing of bridge structures. *Engineering Structures, 17*(2), 113-121.

Seidel S, Rappaport T. (1992). 914 MHz Path Loss Prediction Models for Indoor Wireless Communications in Multifloored Buildings, *IEEE Transactions on Antennas and Propagation*, 40(2), 207-217.

Sensicast Systems, Inc. (2008). Needham, MA. http://www.sensicast.com.

Shankar P.M. (2002). *Introduction to Wireless Systems.* Wiley and Sons, New York.

Shin S. Y., Park S. P., Kwon W. H. (2007). Mutual interference analysis of IEEE 802.15.4 and IEEE 802.11b, *Computer Networks*, 51, 3338-3353.

Silicon Designs, Inc., “Model 1221 Low Noise Analog Accelerometer,” Issaquah, WA (2007).

Sim, S. H. and Spencer Jr., B.F. (2007). "Multi-scale sensing for structural health monitoring", *Proc. World Forum on Smart Materials and Smart Structures Technology*, Chongqing, China.

Sim, S. H. Spencer, Jr., B.F., Zhang, M. and Xie, H. (2009). Automated decentralized modal analysis using smart sensors, *Journal of Structural Control and Health Monitoring,* accepted.

Sim, S.H. (2009) Decentralized Identification and Multimetric Monitoring of Civil Infrastructure using Smart Sensors, Ph.D. Research Proposal, Dept. of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, June 2009.

Singh, M.P. and Huhns, M.N. (2005). Service-Oriented Computing: Semantics, Processes, Agents, John Wiley and Sons, New Jersey.

Sohn, H. (2003). *A review of structural health monitoring literature: 1996-2001* No. LA-13976-MS)Los Alamos National Laboratory.

Sohn, H. (2007). Effects of environmental and operational variability on structural health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 365*(1851), 539-560.

Sohn, H., Worden, K., and Farrar, C. R. (2002). Statistical damage classification under changing environmental and operational conditions. *Journal of Intelligent Material Systems and Structures, 13*, 561-574.

Spectral Dynamics, Inc., San Jose, CA.

Spencer Jr., B.F., Ruiz-Sandoval, M and Kurata, N. (2004), Smart Sensing Technology: Opportunities and Challenges, *Structural Control and Health Monitoring* 11, 349–368.

ST Microelectronics (2005a). “LIS3L02DQ MEMS Inertial Sensor,” Geneva, Switzerland.

ST Microelectronics (2005b). “LIS3L02AS4 MEMS Inertial Sensor,” Geneva, Switzerland.

ST Microelectronics (2005c). “AN2041 Application Note,” Geneva, Switzerland.

ST Microelectronics (2008). “LIS3L02AL MEMS Inertial Sensor: 3-axis - +/-2g Ultracompact Linear Accelerometer,” Geneva, Switzerland.

STMicroelectronics (2009). “LIS344ALH Ultracompact MEMS Inertial Sensor,” Geneva, Switzerland.

Straser, E. G and Kiremidjian, A. S. (1998), A modular, wireless damage monitoring system for structures. Report No. 128, John A. Blume Earthquake Engineering Center, Department of Civil and Environmental Engineering, Stanford University, Stanford, CA,

Texas Instruments, Inc. (2008) Low-Power, Single-Supply, Rail-to-Rail Operational Amplifiers, MicroAmplifier ™ Series, Dallas, TX.

TinyOS, http://www.tinyos.net, (2006).

Tsai, W.T. (2005). Service-Oriented System Engineering: A New Paradigm. *Proc. IEEE International Workshop on Service-Oriented Systems Engineering*, 3-8.

Walter, P. L. (2007). The history of the accelerometer, 1920s - 1996 - prologue and epilogue, 2006. *Sound and Vibration, 4*(11), 84-90.

Wang, Y.; Lynch, J.P.; Law, K.H. (2004) Wireless structural sensors using reliable communication protocols for data acquisition and interrogation. *Proc. of the 23rd International Modal Analysis Conference*.

Whelan, M. J., Fuchs, M. P., Gangone, M. V., and Janoyan, K. D. (2007). Development of a wireless bridge monitoring system for condition assessment using hybrid techniques. *Proc. SPIE Smart Structures,* San Diego, CA.

Whelan, M.J and Janoyan, K.D. (2009) Design of a Robust, High-rate Wireless Sensor Network for Static and Dynamic Structural Monitoring, *J. Intelligent Material Systems and Structures* 20 (7): 849-863

Wong, K. Y. (2004). Instrumentation and health monitoring of cable-supported bridges. *Structural Control and Health Monitoring*, 11(2), 91-124.

Woo, A. Tong, T. and Culler, D. (2003) Taming the underlying challenges of reliable multihop routing in sensor networks, *Proc. 1st international conference on Embedded networked sensor systems,* Los Angeles, California.

Zhao J, Govindan R. (2003). Understanding Packet Delivery Performance In Dense Wireless Networks. *Proc., of the 1st Int. Conference on Embedded Network Systems*, ACM, Los Angeles, CA, 1-13.

Zimmerman A.T., Shiraishi M., Swartz R.A., Lynch J.P. (2008) Automated Modal Parameter Estimation by Parallel Processing within Wireless Monitoring Systems. *Journal of Infrastructure Systems*, 14(1): 102-113.

APPENDIX A: SHM-A SENSOR BOARD DATASHEET & USER'S GUIDE

SHM-A Board

Multimetric Imote2 Sensor Board

Datasheet and User's Guide

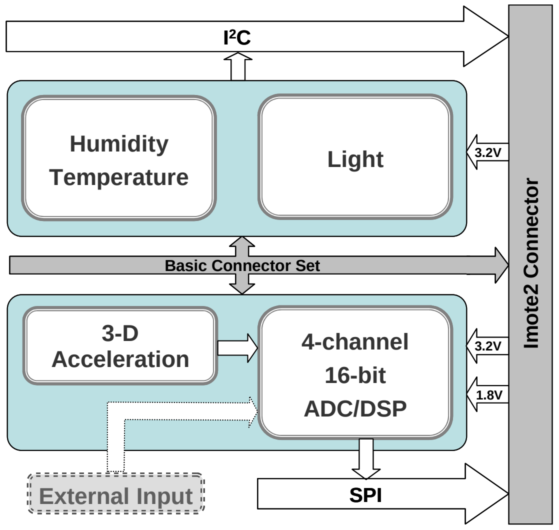
ISHMP
UIUC
OSL & SSTL

Overview

Developed as part of the Illinois Structural Health Monitoring Project, the SHM-A sensor board is designed to interface with the Imote2 smart sensor platform. This versatile sensor board is tailored to structural health monitoring (SHM) applications and is capable of providing the information required for comprehensive infrastructure monitoring. The sensor board provides three axes of acceleration as well as light, temperature and humidity measurements. The 4-channel analog to digital converter (ADC) can accommodate the addition of one external analog input signal, e.g. strain measurement.

Features

- Three axes of acceleration measurement
- Temperature and relative humidity measurement
- Single-channel external analog input to 16-bit ADC
- User-selectable sampling rates and cut-off frequencies
- Customizable digital filters
- Open-source software available for operation with the Imote2



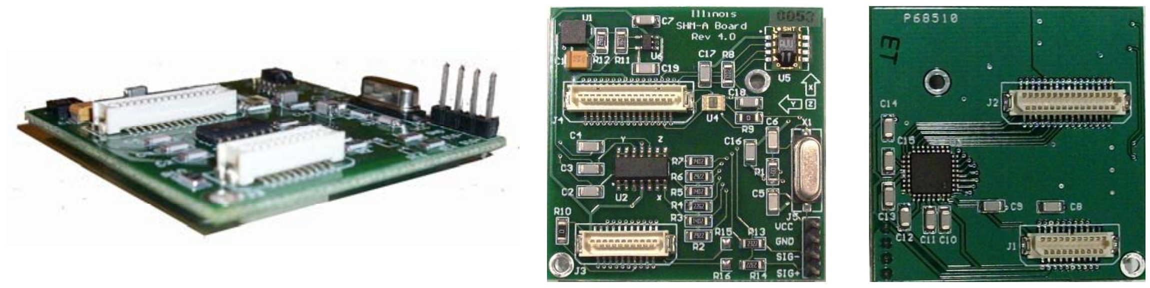


Figure 1. SHM-A sensor board: perspective (left), top (middle), and bottom (right).

Contents

1 Block Diagram and Pin Descriptions

1.1 Block Diagram

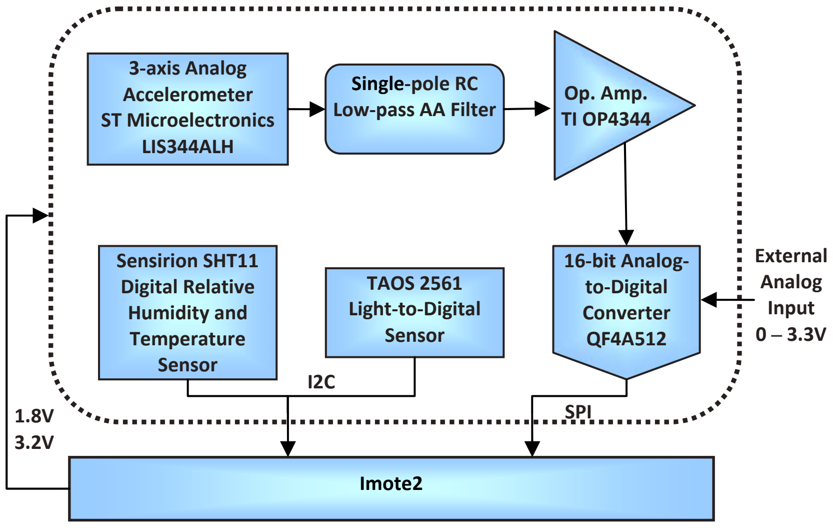

3-axis Analog Accelerometer ST Microelectronics LIS344ALH
Single-pole RC Low-pass AA Filter
Op. Amp. TI OP4344
Sensirion SHT11 Digital Relative Humidity and Temperature Sensor
TAOS 2561 Light-to-Digital Sensor
16-bit Analog-to-Digital Converter QF4A512
External Analog Input 0 – 3.3V
I2C
SPI
1.8V
3.2V
Imote2

Figure 2. SHM-A block diagram

1.2 Pin descriptions

The SHM-A board connects to the Imote2 via two connectors located on the bottom of the board. In addition, the SHM-A board provides two connectors on the top of the board to allow the stacking of additional boards to interface with both the SHM-A board and the Imote2. Figure 3 gives the dimensions of the SHM-A sensor board, indicates the location of the connectors on both the top and bottom of the board, and shows the acceleration measurement directions. The pin descriptions are given in Tables 1, 2 and 3.

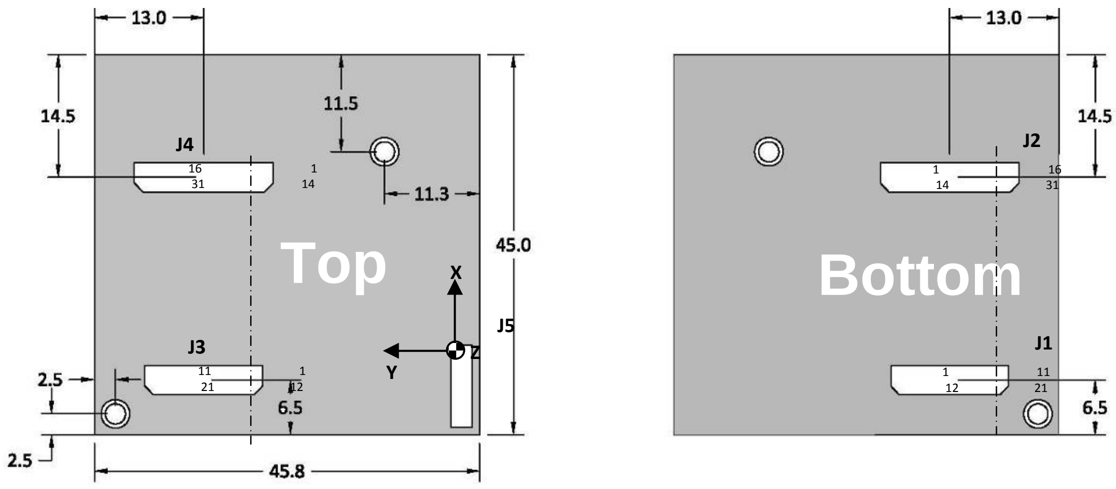


Figure 3. SHM-A dimensions (all dimensions in mm).

Table 1. Imote2/SHM-A 31-pin connector (J2 and J4) pin descriptions.

Pins	Group	Imote2 Description	SHM-A Functionality
1	UART1	Serial port communication or General Purpose I/O	none
2			none
3			SHT11 (Temp. and Hum.) data[1]
4			SHT11 (Temp. and Hum.) clock[1]
5-8	UART2	Serial port communication or General Purpose I/O	none
9	GND	Ground	GND
10	SPI2	SSPCLK2– SPI Clock	none
11		SSPFRM2 – Chip Select	none
12		SSPTxD2 – SPI Serial Data Input	none
13		SSPRxD2 – SPI Serial Data Output	none
14	GPIO94	General purpose I/O	TAOS2561 (Light Sensor) interrupt
15,16	Reserved	Reserved	none
17	I^{2}C	SCL – I^{2}C Serial Clock	TAOS2561 (Light Sensor) clock
18		SDA – I^{2}C Serial Data	TAOS2561 (Light Sensor) data
19	SPI1	SSPCLK1– SPI Clock	QF4A512 (ADC) SPI Clock
20		SSPFRM1 – Chip Select	QF4A512 (ADC) Chip Select
21		SSPTxD1 – SPI Serial Data Input	QF4A512 (ADC) Serial Data Output
22		SSPRxD1 – SPI Serial Data Output	QF4A512 (ADC) Serial Data Input
23	GPIO10	General Purpose I/O	QF4A512 (ADC) Data Ready Interrupt
24	GND	Ground	GND
25	SDIO	MMCLK	none
26		MMCMD	none
27		MMD0	none
28		MMD1	none
29		MMD2	none
30		MMD3	none
31	GPIO93	General Purpose I/O	QF4A512 (ADC) Chip Reset

[1]The humidity and temperature sensor cannot be accessed when the Imtoe2 is connected to the debug board (IIB2400) since it uses the same pins as the one of the two serial ports used by the debug board.

Table 2. Imote2/SHM-A 21-pin connector (J1 and J3) pin descriptions.

Pins	Group	Imote2 Description	SHM-A Functionality
1-2	VBAT	Drives power to processor (3.2 – 4.7V input)	none
3	GND	Ground	GND
4	PMIC_TBAT	PMIC battery temperature input	none
5-9	Reserved	Reserved	none
10		Available for expansion	Negative external analog input (0-3.3V) **J3 (top) only**[1]
11			Positive external analog input (0-3.3V) **J3 (top) only**[1]
12	Power	1.8V (programmable 1.8 – 3.3V)	1.8V supply
13		3V (programmable 1.8 – 3.3V)	3.2V supply
14	Reserved	Reserved	none
15	ALARM	Alarm input to PMIC	Connected to VRTC (18)[2]
16	RESET	Reset – manual reset	none
17	GND	Ground	GND
18	VRTC	Imote2 processor powered indicator - high if on or asleep	Connected to Alarm (15)[2]
19	nCHARGE_EN	Battery select (primary or rechargable)	none
20	STDUart	STD_RxD – Debugging with BLUSH	none
21		STD_TxD – Debugging with BLUSH	none

[1]The external input can come from either J5 or from J3. Populate R13 and R14 for connection via J5 OR populate R15 and R16 for connection via pins 10 and 11 of J3.

[2]VRTC is connected to the PMIC Alarm if R10 is populated. This connection causes the Imote2 to power on if a USB plug power source is inserted or the Imote2 is connected to a powered battery board without the need to press the reset button.

Table 3. SHM-A 4-pin external analog input pin descriptions.

Pin	Label	Description
1	VCC	Power supplied by SHM-A board to external device (typically 3.2V)
2	GND	Ground
3	SIG-	Negative external analog input (0-3.3V)[1]
4	SIG+	Negative external analog input (0-3.3V)[1]

[1]See note 1 on
Table 2.

2 Mechanical and electrical specifications

2.1 Mechanical characteristics

Table 4. Acceleration characteristics @ V_{SB} = 3.2V, T = 25°C unless otherwise noted.

Parameter	Min	Typ.	Max.	Units
Acceleration Range[1]		±2		g
Least significant bit (LSB)	0.133	0.143	0.152	mg
Sensitivity	6600	7000	7500	LSB/g
Zero-g offset	13300	14000	14600	LSB
Temperature sensitivity, all axes[2]	-0.08		0.02	%/°C
Zero-g change vs. temperature, x & y axes[2]		-1.25		mg/°C
Zero-g change vs. temperature, z axis[2]		-2.75		mg/°C
Noise floor, x & y axes	0.2	0.3	0.7	mg
Noise floor, z axis	0.3	0.7	1.2	mg
Maximum Frequency[1,3]	1158	1448	1736	Hz

[1]According to STMicroelectronics LIS344ALH Datasheet: http://www.st.com/stonline/products/literature/ds/14337/lis344alh.htm.
[2]Before on-board temperature correction
[3]This is represents the maximum analog frequency prior to digital filtering that results from 1nF capacitors on the output of the accelerometer

Table 5. Environmental sensor characteristics

Parameter	Min	Typ.	Max.	Units
Light Range[1]	0.1		40000	Lux
Light Resolution[1]		16		bit
Temperature Range[2]	-40		123.8	°C
Temperature Resolution[2]	0.04	0.01	0.01	°C
Temperature Accuracy[2]		±0.4		°C
Temperature Response Time[2]	5		10	s
Humidity Range[2]	0		100	%RH
Humidity Resolution[2]	0.4	0.05	0.05	%RH
Humidity Accuracy[2]		±3.0		%RH
Humidity Response Time[2]		3		s

[1]From TAOS Light-to-Digital Converter (TS2561) Datasheet: http://www.taosinc.com/getfile.aspx?type=press&file=tsl2560-e58.pdf. Not specifically tested on the SHM-A board.
[2]From the Sensirion SHT11 – Digital Humidity Sensor Datasheet: http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf. Not all characteristics were specifically tested on the SHM-A board.

2.2 Electrical characteristics

Table 6. SHM-A electrical characteristics.

Parameter	Condition	Min.	Typical	Max.	Units
Supply voltage			1.8		V
		3.2	3.2	3.3	V
SHM-A current draw[1]	3.2V, QF4A512 operating, 3-ch.		12.8	13.2	mA
	1.8V, QF4A512 operating, 3-ch.		79.6	82	mA
SHM-A + Imote2 current draw	SHM-A powered down + Imote2 @ 13MHz		41	56	mA
	SHM-A operating (3 ch.) + Imote2 @ 104MHz		169	184	mA
Analog external input voltage		0		3.3	V

[1]From the QF4A512 Programmable Signal Conditioner Datasheet: http://www.quickfiltertech.com/files/QF4A512revD7.pdf.

3 Typical performance characteristics

The following plots are intended to give a range of expected performance of the sensor boards. Unless otherwise noted, the boards were tested at ~25°C with V_{SB} = 3.2V.

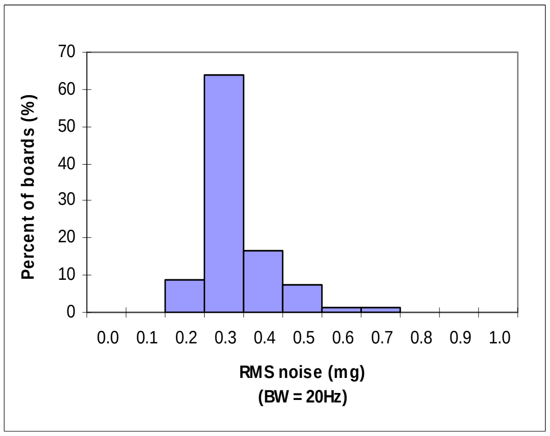


Figure 4. RMS noise for 20-Hz bandwidth, x and y axes.

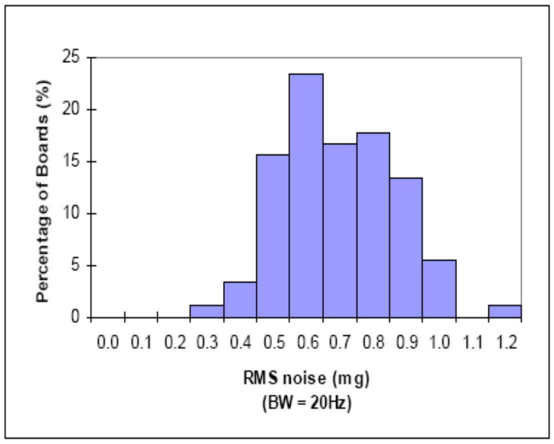


Figure 5. RMS noise for 20-Hz bandwidth, z axis.

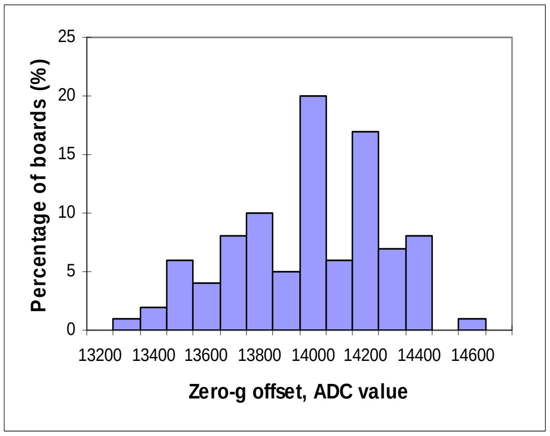


Figure 6. Zero-g offset ADC value.

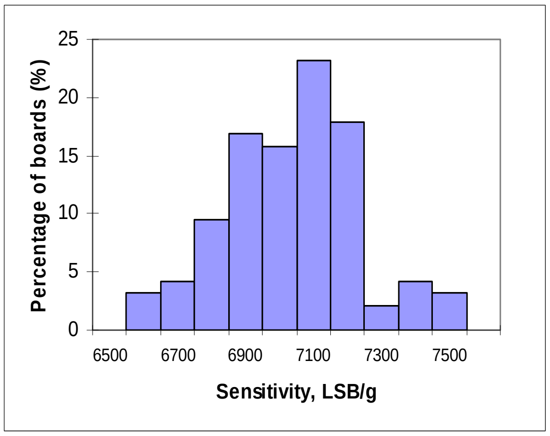


Figure 7. Sensitivity in LSB (ADC value)/g.

During sensing the SHM-A board self-heats (primarily due to the ADC). To account for the effects of this self-heating as well as any changes in the ambient temperaturon the mean values (zero-g offsets) of the acceleration output, the temperature measurements are used to provide on-board correction. The following plots show the typical change in the zero-g levels and sensitivity without temperature correction.

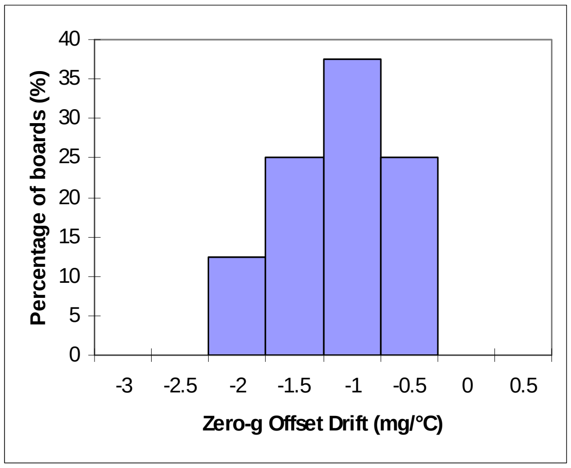


Figure 8. X-axis zero-g drift vs. temperature.

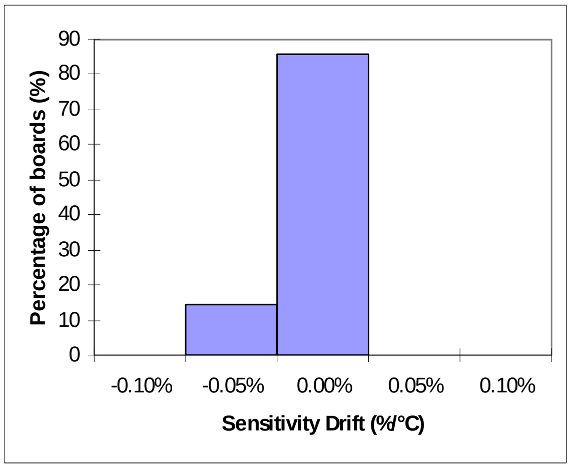


Figure 9. X-axis sensitivity drift vs. temperature.

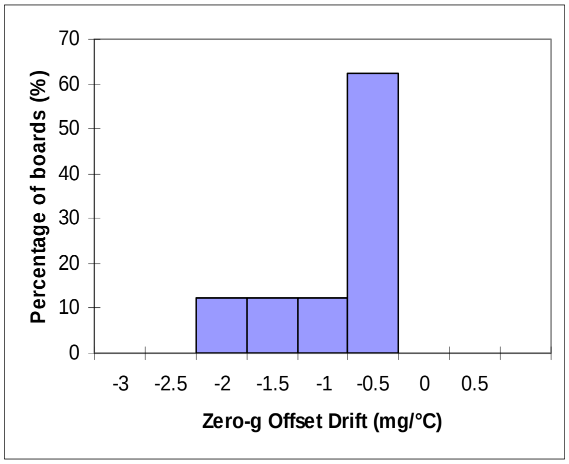


Figure 10. Y-axis zero-g drift vs. temperature.

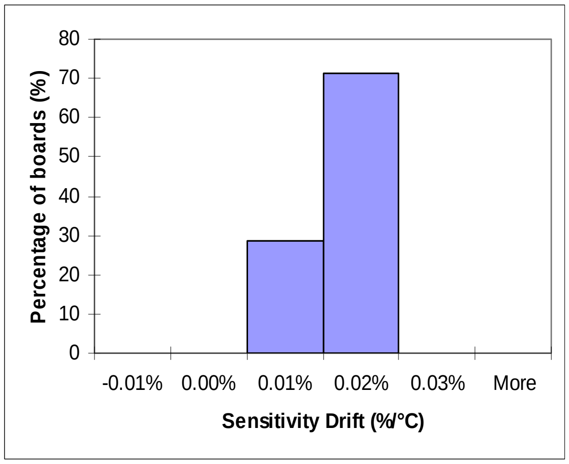


Figure 11. Y-axis sensitivity drift vs. temperature.

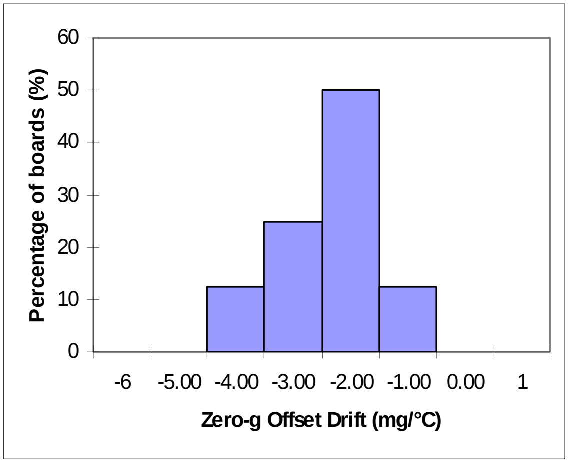


Figure 12. Z-axis zero-g drift vs. temperature.

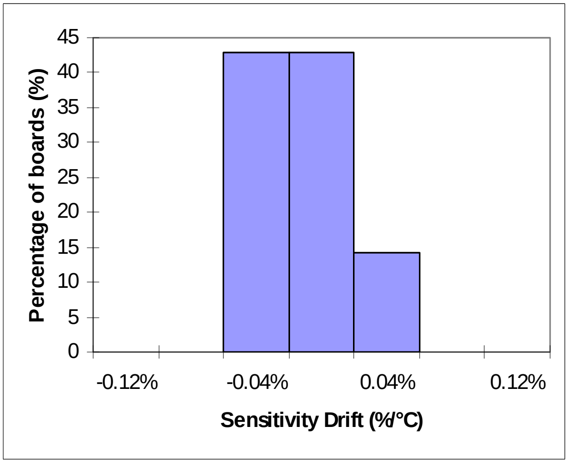


Figure 13. Z-axis sensitivity drift vs. temperature.

4 Software

The software required to operate the SHM-A board interfaced with the Imote2 is open-source and can be found at http://shm.cs.uiuc.edu/software.html. This software includes drivers for the QF4A512, the temperature and humidity sensor and the light sensor as well as a wide range of application software for acquiring data locally and remotely.

4.1 Driver

After installing the ISMHP toolkit, the driver for the SHM-A sensor board can be found in the shm/sensorboards/SHM_A directory. Included in this directory are the main driver for the QF4A512 (ADC) as well as drivers for the SHT11 (temperature and humidity) and the TAOS 2561 (light) components. Table 7 describes the contents of each file.

Table 7. SHM-A driver files.

Component	File	Description
QF4A512 - Accelerometer and external input ADC	AccelSensorM.nc	Module and configuration file for QF4A512 operation.
	AccelSensorC.nc	
	AccelSensor.h	Defines constants associated with QF4A512 operation.
	filters.h	Defines configuration files and filter delays to be loaded for QF4A512 operation.
TAOS 2561 – Light Sensor	LightSensorM.nc	Module and configuration file for light sensor operation.
	LightSensorC.nc	
	LightSensor.h	Defines constants and numerical relationships for light sensor
SHT11 – Temperature and Humidity Sensor	TempHumSensorM.nc	Module and configuration file for temperature and humidity sensor operation.
	TempHumSensorC.nc	
	TempHumSensor.h	Defines constants and numerical relationships for temperature and humidity sensor.
	hardware.h	Defines SHT11 I2C parameters.
SHM-A – whole board operation	SensorboardM.nc	Module and configuration file for SHM-A board management including channel setup, memory allocation for sensed data and temperature correction.
	SensorboardC.nc	
	sensorboard.h	Defines constants for SHM-A board including temperature correction factors.
	TempCalibrationM.nc	Module and configuration file for a utility to estimate temperature correction factors. Operates with BluSH command.
	TempCalibrationC.nc	

4.2 Channel configuration file

The channel configuration file specifies the number of active channels as well as the sampling rate, analog gain, and digital filter characteristics for each channel. There are four default configuration files included in the SHM-A sensor board driver in the ISHMP Toolkit. Table 8 gives the parameters for each of these files.

Table 8. Default QF4A512 configuration parameters.

Sampling Rate (Hz)	Cut-off Frequency (Hz)	Gain	Active Channels	Latency (points)	File Name
25	10	1	1,2,3	76	filter3ch_fs25Hz_fc10Hz.h
50	20	1	1,2,3	89	filter3ch_fs50Hz_fc20Hz.h
100	40	1	1,2,3	78	filter3ch_fs100Hz_fc40Hz.h
280	70	1	1,2,3	94	filter3ch_fs280Hz_fc70Hz.h

Please refer to “SHM-A Sensor Board: Advanced User’s Guide” for instructions on creating new configuration files.

4.3 Application software

The application software in the Illinois SHM Toolkit allows the acquisition of data from the SHM-A sensor board. Please see the associated documentation for further instructions on the use of the application software.

For inquiries, please contact:

Professor B.F. Spencer, Jr.
bfs@illinois.edu
University of Illinois at Urbana-Champaign
Department of Civil and Environmental Engineering
2213 Newmark Civil Engineering Laboratory, MC-250
205 North Mathews Ave
Urbana, IL 61801
USA

Or visit:

http://shm.cs.uiuc.edu

List of Recent NSEL Reports