INDIAN INSTITUTE OF TECHNOLOGY KANPUR

Design Issues and Experiences with BRIMON Railway BRIdge MONitoring Project

by

Nilesh Mishra

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree
BACHELOR-MASTER OF TECHNOLOGY (DUAL DEGREE)

Approved, Thesis Committee:

Dr. Bhaskaran Raman , Chair
Department of Computer Science and Engineering
IIT Kanpur

Dr. Rajat Moona
Department of Computer Science and Engineering
IIT Kanpur

Dr. A. R. Harish
Department of Electrical Engineering
IIT Kanpur

Kanpur,
U.P., India
208016
August, 2006

Certificate

This is to certify that the work contained in the thesis entitled *"Design Issues and Experiences with BRIMON Railway BRIdge MONitoring Project"*, by *Nilesh Mishra*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

August 2006

(Prof. Bhaskaran Raman)
Department of Computer Science & Engineering
Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh 208016

ABSTRACT

Structural health monitoring of railway bridges is an essential but currently expensive proposition. Current systems employing data loggers and analyzers are both labour intensive for setup and require trained manpower for deployment and usage. In this work we propose a system design for remote, on-demand railway bridge monitoring system which is easier to deploy and autonomous in its operation. The system is low cost, as off the shelf equipments are used and expensive sensing equipments are replaced with Micro Electronic Mechanical Systems (MEMS) based sensors and Wireless Sensor Networks (WSN) replacing the cabling and data logging system for data transportation and collection. Current structural health monitoring systems are essentially wired solution or proprietary single hop wireless solutions. These systems are non-scalable and difficult to deploy on most of the structures. In contrast our approach using independent nodes attached with appropriate sensors and batteries for power make them easy to deploy without hassles of cabling. Using wireless data transmission and multi-hop data transport our solution is highly scalable. Use of low power devices also enable our design to have a longer maintenance cycle than existing methods which may require separate power generation units such as generators.

Due to challenges in wireless data transfer and distributed nature of data collection any designed system should have both reliable data transport and synchronized operation for data logging requirements. A reliable data transfer protocol called PSFS (Push Slowly Fetch Slowly) and an application specific data routing protocol have been designed for this purpose in a separate related work. In this work we focus on the problem of time synchronization and event detection for data collection. Flooding Time Synchronization Protocol (FTSP) has been used for time synchronization to achieve microsecond level accuracy on the multi-hop topology. We have modified the existing implementation to adopt it for our platform and work on the linear topology generated due to deployment pattern over bridge. We show that these modifications reduce the average error at nodes farther from the root as well as improve the network wide synchronization stability. We employ a Wake-on-WLAN based event detection mechanism for detection of incoming train. In our design the train acts both as mechanical shaker for data collection as well as data transporter from remote bridge location to a data repository. The event detection method also turns on the MEMS sensor just in time for data collection thus saving power. Results reflecting the achievable distances for successful operation of Wake-on-WLAN and comparison of MEMS accelerometers with existing Forced Balance Accelerometers (FBA) show that our design is viable for future deployment. Overall our approach achieves more than 96% cost savings over currently used wired solutions.

Acknowledgments

This work is incomplete without the acknowledgement for the contribution of numerous people at different stages. Firstly, I will like to thank my supervisor Dr. Bhaskaran Raman for the excellent mentoring and guidance given throughout the work. Next, I will like to thank Dr. Kameshwari Chebrolu from Department of Electrical Engineering for an introduction to the problem and fruitful discussions with critical inputs from time to time. Dr. C.V.R. Murthy and Dr. K.K. Bajpai, from the Civil Engineering department, gave crutial domain knowledge and helped in perfection of the design. Mr Abhishek Singhal, from Structures Lab, took special measures and helped at numerous times during shake table related experiments. Along with Mr. Sukanto Kole he also provided important discussions for making and assembling electronic circuits used throughout the project. My batchmates, Saeeduddin Ansari and Rishabh Halakhandi, were helpful at times for conduction of experiements along with Mr. Phani Kumar Valiveti, Mr. Dheeraj Golcha and Mr. Rajkumar. The Department of Computer Science and Engineering needs special mention for the healthy and conducive enviornment offered by it which made working real pleasure. I offer my gratitude to my parents for supporting me at all points in life and also keeping me motivated throughout the process. Last but not the least, I thank Mr. Hemanth Haridas who apart from being a good friend was also a co-worker on the BriMon project.

Contents

Abstract ii
Acknowledgments iii
List of Figures vi
List of Tables viii

1 Introduction 1
1.1 Thesis Contributions . . . 2
1.2 Prior Work . . . 2
1.3 Solution Approach . . . 6
1.4 Overview . . . 12

2 Background 13
2.1 Structural Health Monitoring . . . 13
2.2 Wireless Sensor Network . . . 18
2.3 Time Synchronization . . . 21
2.4 Event Monitoring and Detection . . . 23

3 Related Work 25
3.1 Structural Health Monitoring and Wireless Sensor Network . . . 25
3.1.1 Wireless Sensor Network Applications . . . 25
3.1.2 Structural Health Monitoring using Wireless Sensor Networks 26
3.1.3 Use of MEMS Accelerometers . . . 28
3.2 Time Synchronization . . . 29
3.3 Event Detection . . . 33

4 Application Design: BriMon 35
4.1 Problem Definition . . . 35
4.1.1 Requirements . . . 35
4.1.2 Constraints . . . 36
4.2 BriMon . . . 37
4.3 Event Sequence . . . 39
4.3.1 Data acquisition cycle . . . 39
4.3.2 Time synchronization cycle . . . 40
4.3.3 Routing Cycle . . . 41
4.4 Hardware modules for BriMon . . . 41
4.4.1 Messaging and transporter module . . . 41
4.4.2 Frontier node module . . . 42
4.4.3 Data aggregator module . . . 42

4.4.4 Data collection module 43
4.5 Software modules for BriMon 44
4.5.1 BriMon Time-synchronization: FTSP 44
4.5.2 BriMon Event Detection: Wake-on-WLAN 47

5 Implementation **49**
5.1 Hardware 49
5.1.1 BriMon Hardware Modules 49
5.1.2 BriMon Hardware Components 52
5.2 Software 59
5.2.1 Flooding Time Synchronization Protocol(FTSP) 60
5.2.2 Event Detection 62
5.2.3 Duty-cycling in motes 62
5.3 Discussions 64

6 Performance Evaluation **71**
6.1 Accelerometer comparison 71
6.2 802.11 signal detection using 802.15.4 sensor motes 76
6.3 Mote's range with external antenna 77
6.4 802.11 data transfer in motion 78
6.5 Data transfer rates for Tmote-sky sensor node 78
6.6 FTSP related experiments 79

7 Conclusions and Future Work **89**
7.1 Conclusion 89
7.2 Future Work 90

References **91**

List of Figures

1.1 Different wireless sensor nodes (motes) 4
1.2 Architecture of a sensor node 5
1.3 Wired SHM solution 8
1.4 Single hop wireless data logging solution 9
1.5 Components for a bridge deployment of BriMon 11

2.1 Fall of the first Tecoma Bridge at 10:30 AM on 7th Nov.1940 17
2.2 Mote's component details 19

3.1 Multi-hop Synchronization using time translation in RBS 31
3.2 Spanning tree and level structure in TPSN 32

4.1 Location of different modules for BriMon application 38
4.2 Schematic diagram for a data aggregator node 42
4.3 Schematic diagram for a data collector node 43
4.4 Data communication protocols used between different modules 45

5.1 A prototype data aggregator node with individual components spread out 50
5.2 A prototype data collector node with individual components spread out 51
5.3 Sensor Node based on BIM433 radio and 89C52 micro controller . . . 53
5.4 The latching switching circuit 55
5.5 Soekris single board computers 57
5.6 The attenuator circuit 58
5.7 The SPI-RS232 protocol translator circuit 59
5.8 Component interaction and interfaces used for ClockTimeStamping in TinyOS 60
5.9 Component interaction for FTSP 61
5.10 Component interaction for event detection mechanism 63
5.11 Location of different modules for BriMon 67

6.1 Setup for the shake table experiments with FBA and MEMS based accelerometers 72
6.2 Accelerometer's response to a sinusoidal wave 73
6.3 Power Density curves for the accelerometer's output 74
6.4 Frequency response at 0.1 Hz 75
6.5 FTSP error graphs for original implementation with beacon period of 2 s 81

6.6 FTSP error graphs for modified implementation with beacon period of 2 s 82
6.7 FTSP error graphs for original implementation with beacon period of 10 s 83
6.8 FTSP error graphs for modified implementation with beacon period of 10 s 84
6.9 FTSP error graphs for original implementation with beacon period of 50 s 85
6.10 FTSP error graphs for modified implementation with beacon period of 50s 86

List of Tables

3.1 Comparison of wireless sensor network based applications 27
3.2 Comparison of wireless sensor network based SHM applications . . . 28
3.3 Comparison of existing time synchronization protocols 32

5.1 Connections for JK flip flop in the switching circuit 56
5.2 Cost of components needed for BriMon 69

6.1 Range results of WOW experiment 76
6.2 Range results of mote connected with different antennae 78
6.3 Effective data transfered for bulk data at different speeds 79
6.4 Least beacon id for stable synchronization 87
6.5 Number of packets received at data logger for different nodes at different beacon periods 88

Chapter 1
Introduction

Indian railway's network spans over 63,140 Kms and has over 120,000 railway bridges in service. On this network 14 million people and more than a million tonnes of freight moves daily [6, 31]. Thus it is essential that all the aspects of this network are up and running securely and safely round the clock. Railway bridges are expensive to construct. Most of the bridges are made at structurally challenging places such as river beds, mountains, etc. Indian Railways spends more than 7.5 billion rupees for their maintenance every year [50]. As per the data available amongst the 120,000 railway bridges 44% of bridges are more than 100 years old, and 74% i.e. appoximately 89,000 are more than 60 years old [30]. Thus it becomes essential to monitor the health of these structures on-demand and as frequently as possible.

The life of a bridge is not dictated by its age but rather by its physical state. To identify and expedite the maintenance of its bridges, Indian Railways has a system to mark bridges needing immediate rebuilding or rehabilitation as 'distressed bridges'. These bridges are put under frequent inspections and their rehabilitation process is prioritized. As of year 2001, a Special Railway Safety Fund was mobilized which earmarked a corpus of 15.30 billion rupees for five years to work on 2700 bridges gradually [31]. Rehabilitation and rebuilding process itself is time taking and costly (at times costlier than building a new bridge).

To assess the structural health of a bridge a number of techniques are available. Also there is no single parameter which can be used for getting the complete picture. The emphasis is to use non destructive techniques so as not to disturb the already fragile system. Some of the methods mentioned in the white paper released by Indian Railways [31] are:

- Under water inspections
- Mapping unknown foundations and integrity testing of foundations
- Non destructive testing techniques like ultra sonics, acoustic emission, strain gaging, and radar etc.
- Fatigue life and residual life assessment techniques
- Modern bridge management system (one associating bridge records, deterioration models and cost for maintenance)

Most of these methods need costly and specialized instruments. Also the use of these require highly trained and skilled labor adding to the cost [46]. To overcome

some of these difficulties we provide a viable solution for monitoring bridges which is scalable, portable and easier to install without requiring any on-site skilled support. The proposed architecture gives on-demand data retrieval capabilities and can be installed at remote sites with maintenance cycles spanning months. This solution uses relatively inexpensive MEMS accelerometers along with wireless sensor network (WSN) to achieve the goal.

1.1 Thesis Contributions

The goal of the system is to record the structural response of a railway bridge toward different type of vibrations. These vibrations are recorded using accelerometers attached with the piers or other strategic locations of the bridge, to be designated by an engineer/consultant. Depending on the length and design of the bridge, these accelerometers can be separated by a distance of 5-60 m. We need to recover these accelerometer generated data with high reliability and fidelity. For frequent data acquisition the equipment needs to be left on site and be capable of autonomous and on-demand data collection.

In this work we provide the following:

- Complete system design for fulfilling the requirements mentioned above.
- Auxiliary electronics circuits to complement the generic hardware being used.
- Customization of time synchronization protocol for linear networks with domain specific time error bounds.
- Event detection method for data gathering.
- Data transportation method for transfer of data from remote bridge location to data repository

The complete architecture can be seen as a collection of software and hardware components. We provide the solution using generic sensor motes, MEMS accelerometers, and 802.11 radio. Of the software components we provide implementation of event detection and time synchronization modules. The usability of these components is both justified and substantiated for the application via various evaluation experiments provided in Chapter 6.

1.2 Prior Work

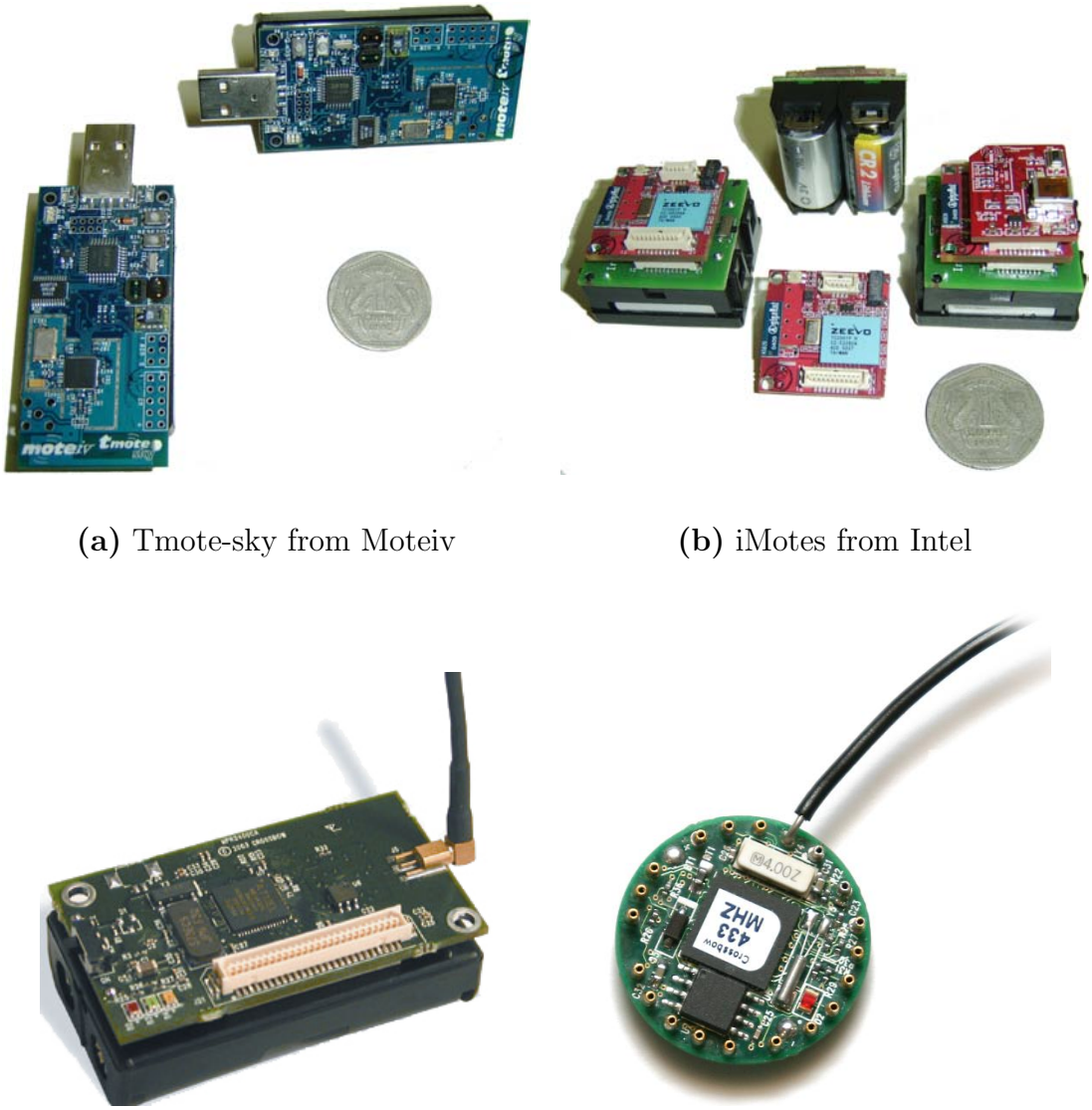
Structural Health Monitoring (SHM) systems are used for damage detection, damage localization and prediction of remaining lifespan for structures. SHM techniques

are used for monitoring air, space and civil structures (such as bridges and buildings). Typical SHM studies involve stress, pressure, displacement, temperature and acceleration measuring sensors. These sensors are placed at different locations on the bridge and the readings are taken from them. A given SHM system can be active or diagnostic based on the way it is used. It can be used as an advanced warning system which is embedded in the structure and continuously monitors the health and gives warnings based on change in values of parameters for the given structure. On the other hand it can be deployed occasionally on a structure to check for the values of some intrinsic property. These techniques fall under the non-destructive analysis techniques and the most common method is to use the vibration data from accelerometers [10]. After performing frequency domain transformations the data from all the sensors can be combined to get a model of the stress patterns in the bridge.

Accelerometer data can be used primarily for two types of studies. In the first case ambient vibrations are used to determine the eigen frequencies in the Fast Fourier Transformation (FFT) curve of the temporal data from the accelerometers. The second type of study tries to find these structural properties from the acceleration values obtained by actuating vibrations into the system [2]. Thus both forced and free vibrations can be used for getting such data sets [2]. Deployed systems used can be divided into data loggers or data analyzers [13]. The former collects time stamped data to be analyzed later while the latter provides both logging and analysis on-site.

A data logging system consists of a storage media, multiple channels to connect sensors and a clock for time-stamping the data. With advances in memory and wireless communication technology, such capabilities can now be seen in small assemblies called wireless sensor nodes. A wireless sensor node [63, 49, 42] is a small, low power, ideally inexpensive device which can be deployed in large numbers. Each node has a limited processing power, limited memory and battery supply as well as limited range of communication and region for which sensor data has been collected. A few generic and commercial sensor nodes are shown in Figure 1.1. A collection of such nodes co-ordinating amongst themselves constitute a wireless sensor network (WSN).

A typical WSN consists of sensor nodes (motes), corresponding sensors, and software to run and maintain the node and the network. Figure 1.2 gives the architecture of a sensor node with different subsystems. A general purpose sensor node has a microcontroller for computation and control, a radio for communication, a power source (usually a battery pack) and a set of interfaces/ports to connect to sensors, actuators or other application specific electronics or auxiliary circuits. These nodes are programmed either by connecting them to a computer or a special programming board. It is not uncommon to have certain generic sensors such as temperature, humidity and photo sensors located on the motes. Depending on type of application and cost involved it can also provide additional data memory (such as flash), analog to digital and/or digital to analog converters. The software on the mote can be application



(a) Tmote-sky from Moteiv

(b) iMotes from Intel

(c) MicaZ from Crossbow

(d) Mica2dot from Crossbow

Figure 1.1 Different wireless sensor nodes (motes)
(**Source:** images c and d `www.xbow.com`)

specific code or generic program with deployable modules to change the functioning of the nodes.

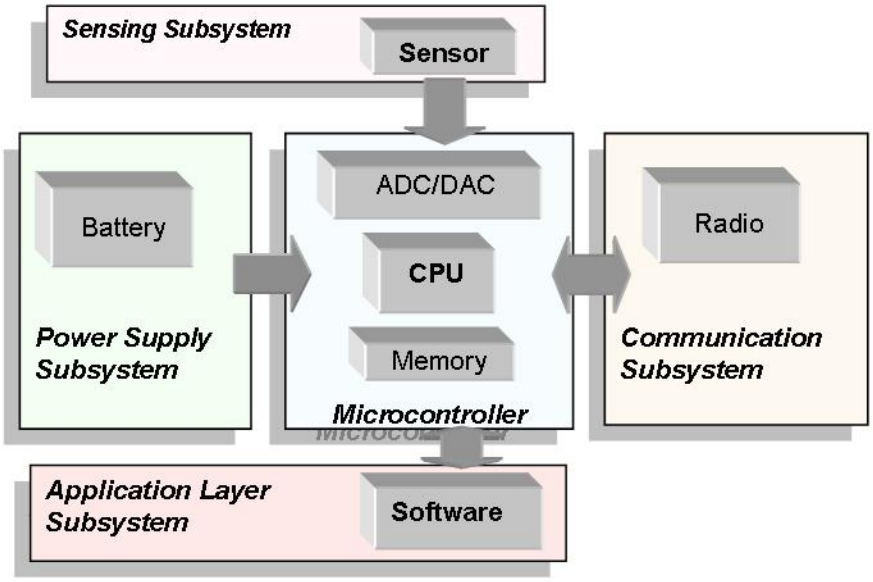


Figure 1.2 Architecture of a sensor node

Wireless sensor networks have been used for numerous monitoring applications. Foremost are the Habitat monitoring applications such as the great duck island experiment [52], studying the redwood tree macroscope project [25], Zebranet project [48] etc. Such monitoring applications have also demonstrated that systems based on sensor networks can be deployed over large duration of time and with minimal maintenance. Also deployments such as the Zebranet have shown that new node designs can emerge from application needs.

There have been numerous study of SHM applications in the sensor network domain. The sensor nodes fit in place of data loggers and transportation back end. A number of projects have given architecture for the complete application design. In [21] Sazonov et al give the architecture of SHM specific node. It has a two tier architecture to perform data collection and aggregation, finally to be transfered to a data repository for analysis. In [46] Xu et al give a SHM system with event detection and light weight time synchronization protocol. Newer model from the same team [36] offers re-programmability and other facilities for the nodes hence making the system more versatile. In the Golden Gate bridge monitoring project [58], a new node design with two MEMS accelerometer sensors has been given. The two sensors have different characteristics and sensitivity towards different ranges thus giving more detailed picture in comparison to single accelerometer based architectures.

Correlation amongst the collected data points during analysis for data logging applications need time-stamping of data points. Time synchronization for a distributed system is an old problem with many proposed and working solutions. But the always on/available and multiple packet exchange models such as Network Time Protocol (NTP) [20] used on uncertain mediums such as internet does not translate effectively to the WSN domain. The nodes have a duty cycle hence they are not always available. The cost of radio communication is higher than computation and node failure happens at a much higher rate than observed in wired networks due to deployment in challenging and harsh environments. Reference Broadcast Synchronization (RBS) [34], Time-sync Protocol for Sensor Networks (TPSN) [54] and Flooding Time Synchronization Protocol (FTSP) [44] are few of the newer generation of time synchronization protocols built for the sensor network domain. All these protocols achieve some accuracy by removing uncertainties from the message transmission/reception pathway of the time synchronization beacons. They also use linear regression for compensating against clock skew. Changes have been suggested in [53] which improves upon the clock model and introduces a variable beaconing rate for the time-synchronization packets depending on the permissible network error values. This allows the nodes to go for sleep over periods ranging from tens of minutes to a few hours maintaining synchronization.

For monitoring operations, detection of onset of the stimuli for the sensor (an event) is important. The easy approach is to keep the sensors on all the time so that every thing gets registered. By duty cycling and sampling the channel for activity we can save energy. This will require determination of correct thresholds and knowledge of event cycle to correctly calculate the sleeping period. This method is not suitable if the power consumption of the sensor is high or the turn on time is substantial. In this case either a second low powered but course grained sensor is deployed or some other low powered or fast turn up method to detect the onset of event is used. For SHM applications vibration is the stimulus for the accelerometers. In [46] authors keep the sensors always on and use a threshold value to decide whether the current data should be logged or not. In [36] authors propose use of a course grained sensor to monitor the channel and if and when an important event occurs switch on the main, more precise accelerometer and log the values.

1.3 Solution Approach

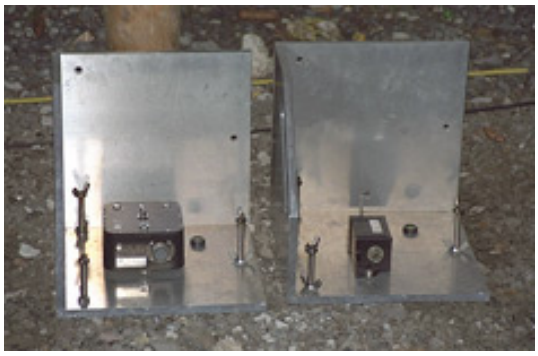
Figure 1.3 gives the various components of a wired SHM system. It employs a data logger (Figure 1.3 (b)) and data analyzer (Figure 1.3 (d)) connected to accelerometers (Figure 1.3 (a)) and other sensors such as velocity and displacement sensor. In most of the cases both data loggers and analyzers are bulky equipments and require considerable amounts of power to run. Laying down of cables and installation of sensors

at the correct location is a tedious job. For larger bridges the setup time can be up to two weeks [46]. Also to run the equipment in a remote location power supplies such as generator sets and batteries are required which again increase the amount of equipment required for the same (Figure 1.3 (e,f)).

With the cost of sensors and monitoring equipments coming down relatively, more and more structures are having provision for having embedded sensors for close monitoring. Increased security awareness has popularized the concept of smart structures which detect change and warn accordingly. Essentially all these systems are still wired solutions where wires run along the structure to the monitoring/data collection center. This not only increases the cost but also has limited lifespan due to various factors (such as chemical reaction of the wire insulation with concrete, rodent attacks on wires, etc.). Also it is not always possible to lay wires in an existing structure which requires monitoring, such as an old railway bridge. In such cases there is a need for an independent module which can take the readings and then transfer them (if possible) wirelessly to a base station. Alternatively, it can also store a local copy which could be retrieved manually and analyzed. Till now single hop wireless data transfer solutions exist, but they cannot be scaled to a large deployment [46]. Figure 1.4 gives example of two such systems available in the market. Although, these systems provide wireless data logging and convenient deployment, these systems are single hop i.e. data is transferred directly from the deployed device connected to the sensor to the central station connected to the data logger. Thus, these nodes can be deployed only on a structure with a span equal to or less than the radio range and cannot be scaled for deployment over larger structures.

Wireless sensor networks (WSN) are fast catching up as the favored method for monitoring and data collection systems in physically challenging environments [14]. They are used to capture an event or monitor an environment in a distributed fashion. Since the nodes are constrained in terms of resources, all subcomponents are energy aware and low power consuming. Further with proper duty cycling these nodes have lifespans (before battery replacement) as long as a year. Large scale deployments of such network under various projects [52, 25, 38, 48] have shown the robustness and viability of solutions based on WSN.

To overcome the issues such as high cost, specialized man power and long setup time we use WSN based solutions in the structural health monitoring domain. The sensors (such as accelerometers) are integrated with the motes to give a data collection module. These modules are then deployed on the strategic points at which the data needs to be collected. A collection of such nodes forms a multi-hop network. Such a system is highly portable, light weight, scalable and deployed very fast with little technical know how. Although this design is limited by the types of sensors which can be interfaced, advancement in MEMS and piezo-electric technologies have opened new avenues of low power and low cost sensors. Designing of such a system has challenges



(a) Accelerometer sensors

(b) Data logger

(c) Diagnostic tools

(d) Data analyzer

(e) Transport and housing

(f) Wired connectivity

Figure 1.3 Wired SHM solution
(**Source:** `www.brimos.com`)

Figure 1.4 Single hop wireless data logging solution
(**Source:** `www.eltekdataloggers.co.uk`,
`www.seaworthysys.com/industrial_internet/hanwell`)

in its own right.

SHM using WSN demands that a number of features are made available by the underlying networking and system software. Despite wireless medium being a lossy medium, one has to ensure that the data is transferred reliably and with high fidelity (application requirement). For tighter control, an application specific routing protocol is needed. The routing protocol must be able to handle the duty cycling and long maintenance cycle of the network. Because of this the routing protocol has to be power efficient and robust enough to take care of node losses. Apart from the routing and transport, the application also requires that one timestamps the data to an accuracy of a few milliseconds as per recommendations of domain experts. Since the nodes will remain idle for majority of time one has to come up with a duty cycle schedule. This should not only be able to conserve energy but also be able to detect the meaningful events, such as arrival of the train for data collection, relevant for registering/transferring data. Thus we effectively divide the complete application into four disjoint parts which when stitched together make a complete SHM system for railway bridges.

- Robust application specific routing
- Reliable transport with high data fidelity

- Time synchronization
- Event detection and data collection from deployment site

In this work we present a system design to achieve SHM using wireless sensor nodes interfaced with low cost MEMS accelerometers. The prototype considered is for monitoring a railway bridge. We try to address the structural engineering problem of finding the natural oscillation frequency of a bridge using vibration data from accelerometers. The system provides a scalable architecture to collect data from accelerometers attached to piers of a bridge, thus reaching longer spans of bridges and structures than possible using a single hop wireless solution. These accelerometers are switched on to collect data when needed and then data is transferred reliably to a central node. Thus the nodes in the network not only act as data collectors but also as data routers. The flow of data is from nodes farthest from the central node towards the central node which acts as temporary data repository. Figure 1.5 gives details of all the components for the system. The three distinct modules named frontier, data aggregator and data collection node are responsible for early detection of trains arrival, temporary data storage and data collection from accelerometers respectively. The frontier node has the capabilities of detection of 802.11 signals while the data aggregator node has a 802.11 radio attached with it to transfer data collected from all the nodes over to the train. Since the effective bandwidth offered by radio chip available on the Tmote motes used by us is in the range of 30 Kbps which is insufficient to transfer the data collected to the train. To achieve this, we need an 802.11 radio offering higher bandwidth (from 1-54 Mbps) connected with the data aggregator nodes. The data collectors are connected with accelerometers and are located on the piers for collecting vibration data.

The application design proposed by us covers the requirements of a SHM system for bridge monitoring. Our focus will be on time synchronization and event detection (using a novel mechanism based on Wake-on-WLAN [45]). Transport and routing are to be handled in a separate work [28].

In the application scenario we require a set of motes to be synchronized on a millisecond level. This is because the data analysis can live with few millisecond errors. The data collected from a node can be used either in isolation where we analyze data from each pier separately or along with data from other nodes. When we analyze the whole bridge as a system then the data collected from two different motes need to be correlated such that we can identify the common events (say vibration generated due to passage of train) in them. There are a number of robust synchronization protocols available as implementations. Looking at the profile of sleep cycle and application topology we chose FTSP [44]. We have ported the current open source implementation of FTSP available from Vanderbilt University for the Tmote sensor

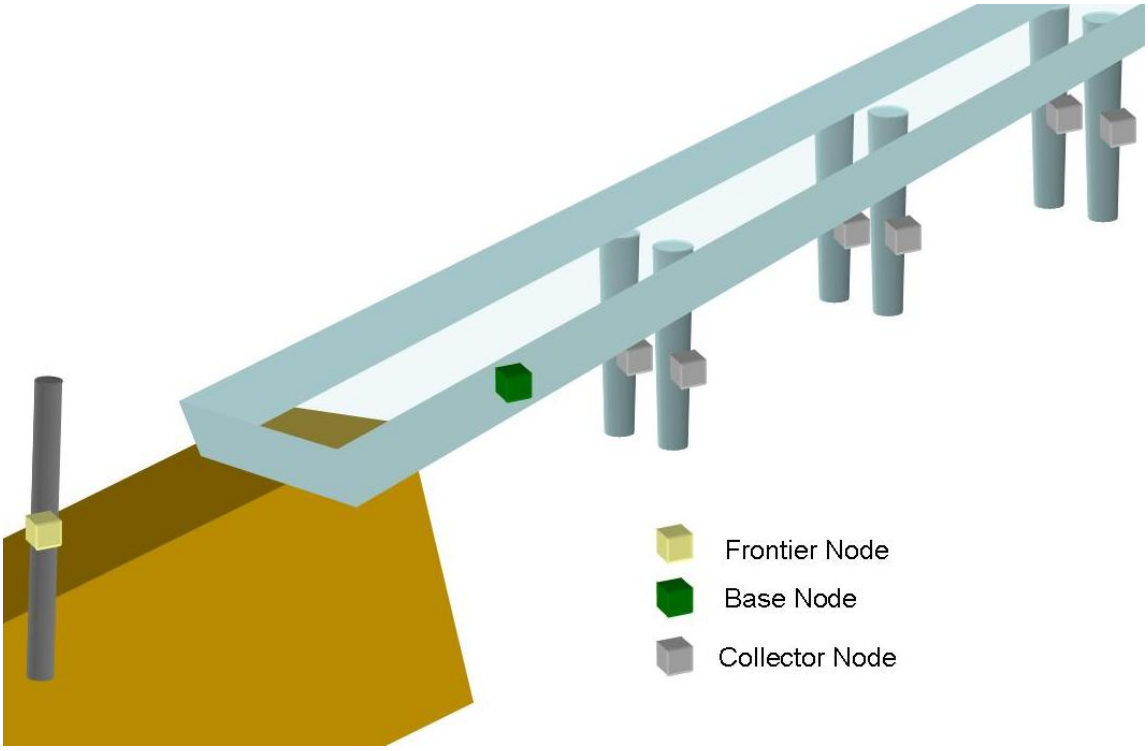


Figure 1.5 Components for a bridge deployment of BriMon

node used by us for application design. The results show that in the given application topology FTSP provides errors in the range of 0-2 ms over a multihop network.

Duty cycling is essential for the application keeping in mind the long maintenance cycle. The architecture design requires that the duty cycle keeps in mind the turn-on time of various components as well as the requirement for occasional beaconing and routing related packet exchange.

In [46] the authors have used a trigger based event (relevant vibration) detection for starting the data logging. Since they were using the setup in a lab environment keeping the sensor always powered on is not a problem. In our setting, accelerometer's current consumption is in the range of 700 μA-10 mA comparable to that of mote itself. To conserve power it makes more sense to switch it off and use it only while taking readings. We use the Wake-on-WLAN mechanism to construct an event detection method. This mechanism uses the feature available on the used platform, Tmote's CC2420 radio to detect 802.11b/g packets. Thus by employing a set of nodes we detect an approaching train and based on whether to take readings or perform data transfer, wake up the motes or transfer data by switching on the appropriate components. In our design the train acts both as vibration generator for taking the readings for forced vibrations as well as a data transporter for collected data from

the remote bridge location to the data repository.

1.4 Overview

Chapter 2 gives the background information on SHM, WSN, time synchronization and event detection. Chapter 3 has the related work in the above mentioned domains. Chapter 4 gives the application design for the BriMon application from hardware and software perspective giving details for different subcomponents and the architecture as whole. Implementation details are given in Chapter 5. Evaluation and results for various experiments done for comparison of MEMS and currently used accelerometers, validation of time synchronization, event detection and data transfer over moving target are provided in Chapter 6. Finally we conclude the work in Chapter 7 with some future work indicated for deployment.

Chapter 2
Background

Structural Health Monitoring (SHM) has been in existence for a century but has gained attention in near past for diagnostic and predictive study of structures for safe use or maintenance. Our solution approach for the railway bridge monitoring uses Wireless Sensor Network (WSN) domain extensively. This chapter provides necessary background details for both these fields. We first provide details about SHM and the problem in question about monitoring of bridges. Next we give background details for the WSN domain followed by time synchronization in WSN. Lastly we cover event monitoring and detection.

2.1 Structural Health Monitoring

By definition, in SHM we monitor, directly or indirectly, various parameters of a structure for identification of its current state. Based on the state we can assess whether the structure is damaged or not. Subsequently, locate the damage and estimate the remaining useful life for safe usage of the structure. Since SHM techniques fall under the non destructive analysis domain, they are gaining prominence amongst the structural engineering community with advancement in theory and equipment technology. The theory for identification of condition of a structure from its eigen-frequencies has been in existence for long. The linear finite element method, forced vibration method and the ambient vibration method were few breakthroughs which happened at regular intervals in the past century. With rapid advances in computer technology and sensing systems, there has been a spurt in SHM related studies [10]. A great number of researchers have shown the use of these methods to obtain useful and correct predictions about different type of structures. The primary use of SHM is for monitoring civil structures such as buildings and bridges, aircrafts, spacecrafts and satellites. As the cost and the risk involved with a particular structure increases, it makes more sense in deploying SHM techniques for monitoring and prediction. SHM is used for:

- Damage detection i.e. is there change in the state of the structure.
- Damage localization i.e if there is a damage, can we locate the site of damage.
- Damage assessment i.e. how extensive is the damage and what is its impact.
- Lifespan prediction i.e. what is the life remaining of the structure for safe use.

As emphasized earlier the life of a structure is not dictated by its age but by its physical state. In order to operate safely one needs SHM methods for assessment of risk from time to time.

On inducing vibrations a structure vibrates at a combination of its natural frequencies (known as modes). Each mode characteristically deforms the structure spatially and temporally. Since each structure has a characteristic mode set, change in the mode shape and frequency means change in the underlying structure. This property is used in most of the diagnostic and damage localization techniques. Also this mode set can be treated as the signature of the structure. On a Fast Fourier Transformation (FFT) graph this signature can be seen with distinct peaks as response to different vibration inputs. This vibration induced in the system can come from various sources. It can be artificially induced using mechanical shakers or by passing of vehicle/railway locomotive over or near the structure. Often it can come from natural sources such as wind and earthquakes. From the point of view of the civil engineers there are three types of vibration which are important.

1. *Forced vibration* is the vibration induced in the system when the source of vibration is still pumping energy into the system. This situation comes up when a moving train passes over a bridge and thus gives it energy.

2. *Free vibration* happens when the source of energy is removed from the structure in question but the residual energy remaining in the system keeps it vibrating. E.g. when a locomotive crosses a bridge, the latter keeps vibrating for some time even after the vehicle has moved away from the bridge.

3. *Ambient vibration* is the vibration in the bridge when no intentional source is pumping energy into the structure. The source of the vibration can be attributed to wind, passage of vehicle on a nearby road, seismic vibrations, etc.

Data from all these sources are used for calculating or finding different parameters associated with the structure. Most of the analysis is done in frequency domain after applying Fourier transformation on the vibration data obtained from accelerometer sensors placed at strategic locations. Since the region of interest lies in frequency values < 50Hz a low pass band filter is used for noise removal and trimming the unnecessary frequency values. The vibration data after FFT transformation is used to derive the eigen frequencies of the structure. This data can further be used to localize the faults in the structure which could have emerged due to aging, overuse or earthquakes, etc. The velocity and displacement values obtainable by subsequent integration of acceleration data are also used for certain analysis.

In practice a typical SHM system can be divided into three main parts:

1. Sensing subsystem

2. Data logger

3. Data analyzer.

Apart from these there are auxiliary systems such as cabling, power supply and instrument protection. Typical sensors getting used in these studies are to measure vibration/acceleration, stress, displacement, humidity, temperature, etc. The parameters monitored are the natural frequency of the structure, vibration signature of the structure, temperature and humidity at critical or vulnerable locations, etc. Data loggers simply collect the data without doing any data analysis. The data is timestamped and analysis is done later after all the data is aggregated. The data loggers have a fixed number of channels (i.e. number of sensors that can be connected with them) to collect data, usually in the range of 2-64. The sensors are connected through co-axial cables using these channel ports. Generally the data logger also powers the sensors. Data analyzers possess all the functionalities of a data logger plus additional features. Depending on the sophistication of the system one can analyze the data on-site using these systems. Normal functionalities provided for data analysis include FFT, integration and differentiation of data etc. It also has additional ports to connect computers/auxiliary storage devices. These days most of the analysis has moved to software as it allows faster development, more sophisticated algorithm implementations and larger library of analysis methods in comparison to hardware based analysis. There are numerous commercial and device specific softwares available. For algorithm development and simulation MATLAB and C find favor amongst the academia.

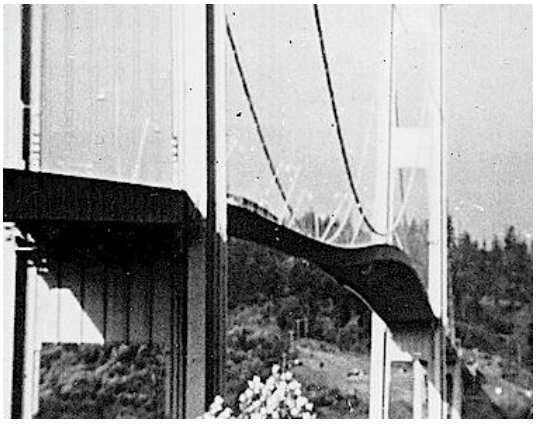
In a given setting first an engineer or consultant studies the structure and determines strategic locations on the structure where different sensors need to be placed. This can be done either by doing a dense deployment of sensors or by sectioning the structure and keeping sensors at each of the section corners [36]. Next the cabling and power supply is planned. This logging can be done at the sensor itself or can be collected at a central location. In most of the current systems data analysis happens at the central repository hence the data is routed to a data logger. Since communication cost is not an issue in wired systems *edge computing* is not given importance and all the data related redundancy or filtering happens at the central location itself prior to analysis. Similarly, since the synchronization issues are not present in the wired domain the time-stamping of the data happens at the data logger rather than at the sensor itself. The power for the sensors is also provided through the signal carrying ducts.

SHM for Bridges

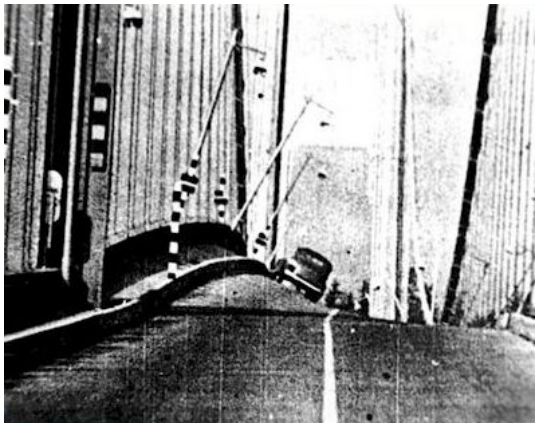
Each structure has multiple natural frequencies. At any time a given structure can be made to vibrate in one of these frequencies by inducing correct amount of energy into the system. If the source and the structure get in sync they can put the structure in resonance which in turn can produce standing waves in the structure. These standing waves have nodes and antinodes based on waves adding constructively or destructively. At the nodes the amplitude of the waves get added hence there is large displacement of surface from the normal state, whereas at the antinodes the waves cross each other hence the surface is still. Nodes and antinodes are the extremes and all other surface points are displaced from the normal position with magnitude in between the two. Thus a lot of stress can get generated in the system if it gets in sync with the external vibration which can cause damage to the structure. The most vivid and prevalent example for structure having standing waves is of First Tacoma Narrows Bridge. At that time it was the longest single span suspension bridge in the USA. The structure would get into sinusoidal motion for wind speeds as low as 6-8 Km per hr but stand still in stronger winds. The bridge failed and fell into the sea on 7th November, 1940. On that day winds blowing at 55-75 km per hour created vortices around the bridge which in turn generated standing waves in the bridge (Figure 2.1(a)). The amplitude of motion at the nodes reached 8.5 m [22]. Although the fall of the bridge is attributed not to the standing waves (Figure 2.1(b,c)) but still its failure (Figure 2.1(d)) gave lot of insight in bridge designing and resonance of structures.

Modern day bridges are made in such a manner that they do not get into resonance first hand and even if they do it causes minimal damage. A lot of factors are kept in mind while designing of a new bridge including traffic pattern, soil profile on which the bridge is constructed, material to be used, maximum wind velocity in the region, etc. To play with natural frequency of a structure a civil engineer can use various techniques pre-construction and post-construction of the structure. During design phase an engineer can change the inter-pier distance, adopt a particular design in preference to others, or change the material to be used for construction. For an existing bridge, weights and hydraulic suspensions can be added to specific regions to change the vibration pattern and hence the natural frequency. Also traffic can be regulated by imposing maximum weight of a passing vehicle and/or specifying the maximum and minimum speed for traffic. Since many of the bridges are more than 100 years old they were not built anticipating todays traffic pattern. With aging the bridge undergoes wear and tear. Additionally the material properties change with time; iron rusts, concrete changes its crystalline properties. Vibration and earthquakes also cause damage. The soil profile underneath also changes with time (certain rivers are known to shift courses during earthquakes and floods). All these change the natural

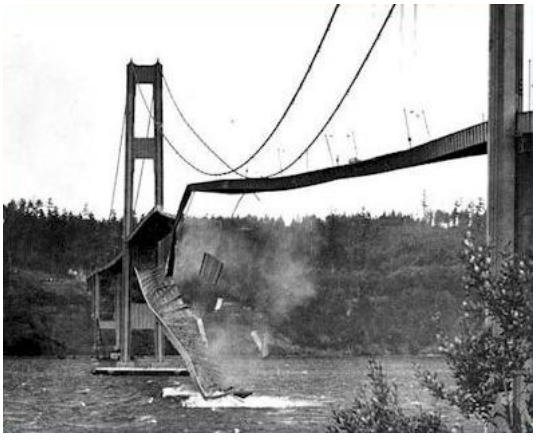
(a) A physicist walking on the antinode line of the standing waves formed on the bridge



(b) The twisted motion of the waves generated by wind which ultimately led to the fall of the bridge



(c) Twisting motion of the bridge from a different angle



(d) Fall of the mid section

Figure 2.1 Fall of the first Tecoma Bridge at 10:30 AM on 7th Nov.1940 (**Source:** `www.lib.washington.edu/specialcoll/exhibits/tnb/page4.html`)

frequency of the bridge and hence it needs to be continuously monitored in order to regulate the traffic or take corrective measures. Our work is to collect vibration data from accelerometers placed on the bridge piers and identify the natural frequencies of the bridge. In Chapter 4 we give the design of a WSN based system to collect such data.

2.2 Wireless Sensor Network

Wireless Sensor Network (WSN) have recently come into prominence as the suitable solution platform for a number of sensing and monitoring applications. Advancements in Application Specific Integrated Circuits (ASIC) and Micro Electronic Mechanical Sensors (MEMS) have led to introduction of low cost and low power devices such as microcontrollers, radios and sensors. Such devices which constitute the heart and soul of the sensor nodes have boosted WSN usage. They leverage their deployment in large numbers to give robustness and fault tolerance with closer monitoring of events and environment. This way they are advantageous over the use of single higher capability sensors. Potential WSN deployment scenarios can be grouped under four application domains [44] namely, monitoring, inventory management, smart systems and military domain. All these application have the common theme of distributed sensing, low on-site computation and availability of low power sensing systems. WSN hardware consists of sensor nodes (motes), sensors, (if any) actuators, and at times higher end devices such as Intel's Stargate gateway [32] to connect sensor nodes to WiFi/ethernet LAN's. The software can be broken down into application specific, network management and OS related code. Newer designs have systems where the application or network management code and also a large chunk of OS code can be changed/upgraded on the fly [19, 47].

Sensor nodes (motes) come in varied configurations based on the application for which they are to be used. They can be differentiated on number of parameters such as processing power, data bandwidth, battery power, etc. Thus they can be optimized on them based on application requirement. Generally, a sensor module consist of a microcontroller, a radio, a flash based memory, a power source (typically a battery), peripherals and connectors (Figure 2.2). Using the connectors one can connect a range of sensors to the mote. Since today's microcontrollers come with most of the functionalities such as ADC, DAC, UART, I2C, timers and program flash the number of peripheral chips are small in number leading to more compact designs. Some of the platforms also have onboard sensors for temperature, humidity or light. One can have external or internal radio antennae depending on the compactness of the design and the range of communication.

The typical sensors used in conjunction with low powered sensor networks are temperature, pressure, light, humidity, acceleration, velocity, displacement, sound,

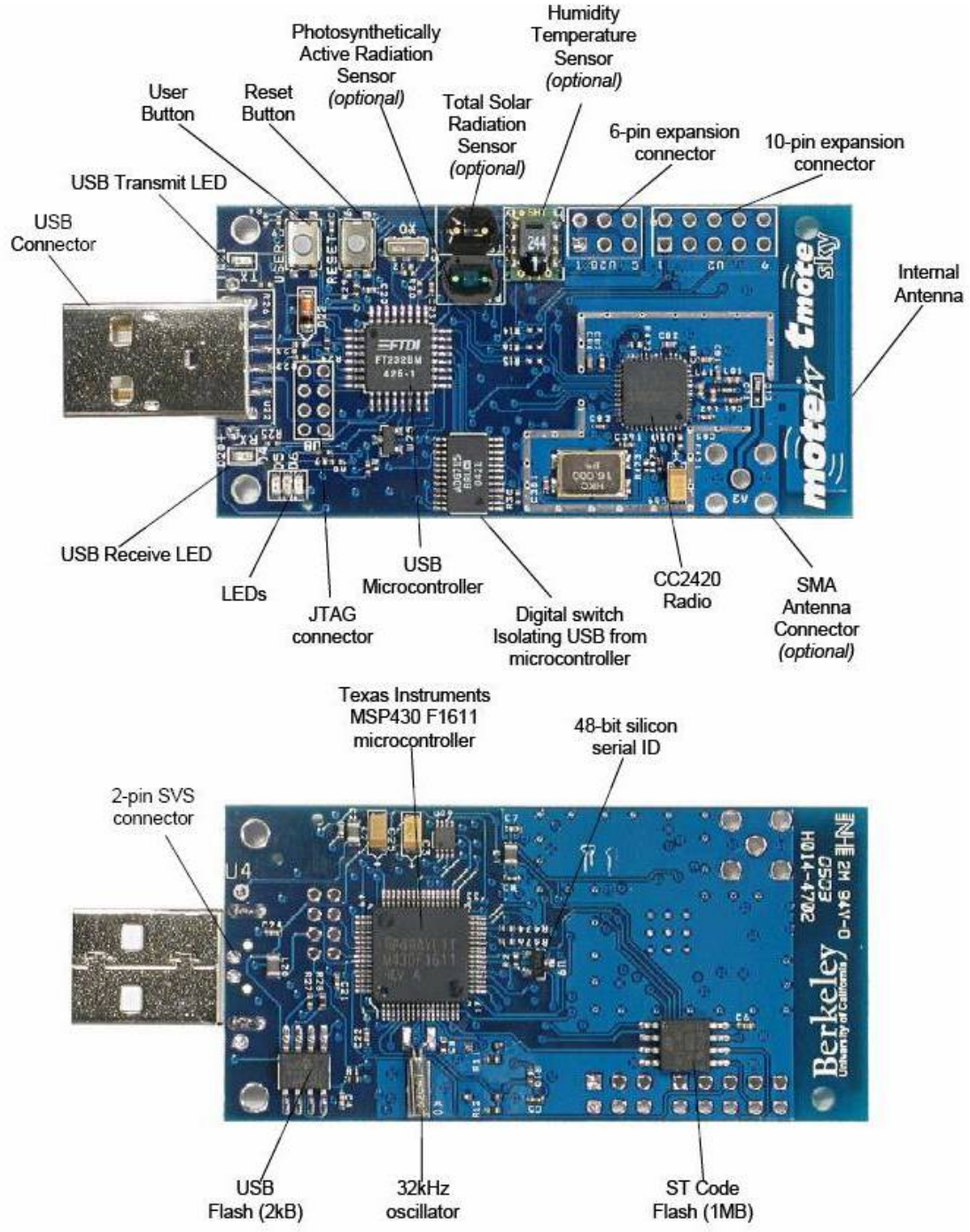


Figure 2.2 Mote's component details
(**Source:** Tmote-sky's datasheet)

chemical and radioactivity sensors. Generally these sensors have analog interfaces, thus one has to use a Analog to Digital Converter (ADC) for getting digital data. Some of the sensors also have digital outputs which can use the USART, SPI or I2C interfaces. With the advent of MEMS, the cost and power consumption of these sensors has reduced dramatically. Also these sensors require less footprint on the Printed Circuit Board (PCB) leading to more compact designs. Since most of the sensor nodes are deployed in harsh environments they need to be properly encased. Generally a waterproof and transparent (for making LEDs visible) casings are preferred. On the battery front one has a wide verity of choices. For application requiring compact designs one can go for expensive and compact Li-ion based batteries. For simpler application R6 alkaline batteries are good enough [52]. If one's maintenance cycle is frequent enough, rechargeable NiMh or lead acid batteries can also be used.

The software used for running the mote and the network has to be closely integrated with the hardware, as for maximum performance one needs to control all the aspects of the hardware. Radio communication is costly in comparison to data storage on flash memory and computation. Thus one thrust of todays techniques is to reduce the amount of communication happening in the network. Thus the routing, transportation, synchronization related components must be designed with minimal communication requirement and with longer periodicity. Similarly, data compression and feature selection techniques are used for removing redundancy and transferring only meaningful data. Since computation is order of magnitude cheaper than radio communication, techniques which can push the computation to these edge nodes and hence reduce amount of data being transferred make sense. But in certain domains this computation cannot be done on the constrained sensor platforms or are very costly e.g. FFT transformation of vibration data [21]. Thus there should be optimal balance between computation and radio data transfer which varies from domain to domain.

For our development we have used TinyOS. TinyOS is an operating system developed at University of California at Berkley (UCB) for programming sensor nodes. It has a component based architecture where only application specific components get compiled and transferred to the nodes during programing. The components have been developed for a number of common platforms and microcontroller architectures. It provides interfacing at various level including direct interfaces with hardware, interface for abstraction of common features and programming level application interfaces. These interfaces are bi-directional and hence can be used to communicate to and fro between the user and provider of interface. A given interface is in turn a collection of commands and events. The commands are implemented by the provider of an interface and the events are implemented by the user of the interface. The provider and user are linked using a process aptly called *'wiring'* in a configuration file. The programing is done in a derivative of C language called nesC.

Current off the shelf motes start from $40 and may go upto few 100s of dollars. An application specific mote developed and produced in mass can cost around $25. As the prices of ASIC goes further down with advances in technology, it is possible to produce these motes very cheaply and in large numbers. Such motes can be deployed densely for distributed sensing applications such as close environment monitoring. Also with proper duty cycling and battery capacity networks spanning months have been reported [52]. In certain applications the power can be scavenged or regenerated to charge the batteries which again improves life of the system [37]

2.3 Time Synchronization

Time synchronization requirements come into picture due to the need of proper time stamping of vibration data coming from different accelerometers. Time stamping of SHM data is necessary for accounting and event correlation. In case of traditional methods of data collection since all the sensors are directly connected to the data logger they can be timestamped using the same reference clock. The uncertainties of data transfer in case of wired solutions is very slight. Again chances of an error occurring or data loss during transfer from sensors to the central data collection point are negligible. When one moves to sensor network domain a new set of problems arise.

Time synchronization is required for sensor motes to perform a number of tasks. Based on the protocol used the motes may require synchronized time for scheduling packets for transmission and reception and/or gaining access over the medium at the MAC layer. If the motes are performing duty cycling they need to wake up in a synchronized fashion so as not to miss important beacon or data transmissions. Data logging applications require marking of collected data with appropriate time stamps for data correlation. In event driven applications it may also be important to maintain absolute time rather than local time, synchronized with the root node in a multihop network. There are multiple ways to perform time synchronization.

- Maintaining a separate and accurate global clock at each node. This can be performed by having a GPS device attached with each mote in question. This method provides nano second level precision at each mote. But this comes at the expense of increased cost (cheapest GPS module set cost $45 [23]). Also the interfacing of the hardware requires communication ports which are scarce on the present day sensor motes (Tmote, Micaz) as they have limited number of ports which can be used for interfacing sensors or other hardware. GPS in itself requires a clear sky to lock on to a set of satellites in order to accurately get the timing information. This may not be always possible in places such as inside a building or underneath dense foliage. Additionally there can be situations when the GPS signal can get jammed intensionally or due to noise.

- The second method is to maintain a global (precise) clock at one central location and then update the clocks of other motes periodically with messages containing sufficient information for the nodes to update and synchronize with the central clock.

- The third method of post-facto synchronization [33] marks the events with local unsynchronized time-stamps which are later corrected using information collected separately to calculate the offsets and drifts of individual clocks.

The reason for nodes not having synchronization can be attributed to two factors, offset and clock skew. Offset is the difference in reference time at the server and the local time at the node. Offset calculations are easy and can be calculated from reference and local time using synchronization points. Such points can be obtained by exchanging timing information of same event marked by both the nodes [34]. Otherwise if one can eliminate the uncertainties in the pathway of message exchange then such synchronization points can be just time stamp information of server [54]. Clock skew on the other hand refers to the difference in time measurement introduced due to clocks of different nodes running at different rates. This is because each node uses a crystal oscillator to generate timing pulses. These pulses are intrapolated or extraolated to get the clock for the node which is used for various operations. Now associated with each crystals are two characteristics as given in [34]:

- **Accuracy** i.e. how close is the oscillators actual frequency to the specified frequency claimed by manufacturer. Typical frequency errors (difference between the two clocks) range from 0-100 μs. Thus typically we will have the clock drifting at a rate of 40-50 μs per second.

- **Stability** i.e. does the oscillator change its frequency with respect to time. This is further divided into short-term and long-term stability. Short-term stability gets effected by factors such as temperature, supply voltage, shock, etc. Long-term stability is governed by crystal aging and usage pattern.

If the synchronization period is small and one can live with the drift during this period then drift compensation is not necessary. But if tighter precision is desired then the time values must be adjusted for the skew. To compensate linear-square regression is the most common method used [34, 44]. Clock skew is the slope of the graph between local time and corresponding offset values. Using this value for adjusting the clock assumes that crystal does not changes its frequency drastically i.e. has high short term stability. Also the table used for regression is refreshed with newer values from time to time.

2.4 Event Monitoring and Detection

Current sensor node platforms are designed such that they last for only four full days when run on full power using a pair of AA alkaline batteries [18]. To use them for a network lasting months without maintenance requires duty cycling and network planning. A given sensor network can have longer life span by careful management of the available energy resources. Most of the components do not need to be switched on all the time. If carefully designed a mote has a provision to invoke and initialize only application specific components as per need basis. Radio is used for transmission and reception, hence if we can predict when a radio communication is going to take place we can switch off the radio rest of the time. Similarly, we do not need the sensors (accelerometers in this case) to be switched on all the time. If one can decide in advance or predict when an event (in our case arrival of the train near bridge location) which is being monitored has to be captured, the concerned sensors can be switched on just in time.

For monitoring applications we can divide the events into two classes of instantaneous events (order of time span for the event in action lesser than turn-on time for sensor) and events with temporal and/or spatial span. These events can further be periodic or non-periodic. Based on the type of event being monitored the event monitoring strategies can be classified as:

- *Always on:* The sensors are always on and continuously monitor. It is energy inefficient with respect to other strategies but is the only solution if the event is non-periodic and instantaneous. Example event: volcanic eruptions

- *Periodic wakeup:* The nodes know or can predict the next happening of the event, hence can switch off the sensors and go to sleep in between the events. This assumes that there is sufficient time to turn-on, stabilize and calibrate the sensors. Example event: Monitoring for moisture content at dawn, noon, dusk and night.

- *Monitoring with rotation:* Since the event can span spatially, if more than one sensor nodes are in the event horizon they can rotate the monitoring responsibility and on the onset the monitoring node can intimate others to wake up and start gathering data. This again assumes availability of sufficient time for sensors to turn-on, stabilize and calibrate themselves. It can also require that the radio communication cost is non-significant which may not be always true. Example event: Monitoring for animals coming to drink water at a forest lake.

- *Associated event monitoring:* If there are associated events happening with the main event which can be monitored at lower costs, then the secondary sensor can monitor for associated event and as and when required turn on the main

sensor. Example event: Turning on the motion camera placed to capture animal's activity in underground burrow using temperature as an associate event.

- *Triggered switch:* Use of passive switches which get turned on with the onset of event. Here also the sensors must have time for stabilization, etc. Example event: Use of mass-spring based switches to monitor vibrations.

- *Two tier approach:* Use of a course grained low powered sensor which can monitor for the event. As and when required the higher resolution but more energy consuming sensor is turned on to capture the event. Example event: Low power MEMS sensors monitor for vibration. When a threshold is crossed the mote turns on a Forced Balance Accelerometer (FBA) to capture data.

The other aspects associated with event monitoring are memory usage, radio communication and edge computing. Since in the current platforms on-board memory is a constraint, event logging must be balanced with data transfer. For event happening in bursts the sensor can use a thresholding, averaging or other detection techniques to decide when an event starts from the data collected. Only the relevant portion of the data can be logged discarding others. Data compression and/or feature selection techniques are also used for reducing the amount of data being logged. They also reduce the amount of data to be transferred over wireless thus saving energy as well.

For monitoring applications event detection is the ability of the node to detect and capture the concerned event with high probability. As explained above the naive approach is to keep the sensors on all the time and logging all the data. This may not be feasible due to energy and memory constraints. Next we can apply duty cycling such that the nodes wake up periodically and on detection of event log it in the memory. But this requires that we can predict in advance the happening of the event or we do not require to record the complete span of the event. Thresholding and signature based event detection techniques are being developed [36] which can be used to detect event onset. Use of two tier sensor architecture has also been proposed by Chintalapudi et al [36].

Chapter 3
Related Work

A considerable amount of work has been done in the sensor network domain for monitoring applications. A number of these projects had the deployment spanning over months and in various environments from city, battle field, industrial floor, ships, on top of a tree to down inside the burrow of ducks [40, 24, 25, 38, 52]. All these have demonstrated that WSN has moved from playground type environment to more demanding commercial and scientific use. In the following sections, work related to SHM and WSN are described. The various approaches and design decisions adopted by them are discussed. Also we give an overview of time synchronization approaches employed by current WSN based SHM techniques and finally briefly touch upon the various event detection techniques used today.

3.1 Structural Health Monitoring and Wireless Sensor Network

In this section we first look at a few wireless sensor network deployments and then cover a few examples of SHM based WSN applications. We also look briefly into use of MEMS based accelerometers in some of these applications which gives BriMon low power monitoring capabilities.

3.1.1 Wireless Sensor Network Applications

Wireless sensor networks have picked up attention of late as the preferred method for monitoring and data gathering applications. Here we analyze a few long term deployment projects. An example of a prolonged deployment (4 months) of sensor network has been documented for the Great Duck Island project [52]. In this work the authors have documented the design, deployment and subsequent analysis of a sensor network used for studying the habitat and micro-climate of sea bird's habitat at Great Duck Island, Maine. The sensor network was a tiered architecture deployed in two patches with single hop and multi hop network. The complete system consisted of motes deployed as burrow motes and weather motes with gateway nodes for connecting to a base station. The base station had a laptop and a 2 way satellite connection for transfer of collected data and receiving commands from a remote monitoring station. Temperature, humidity and barometric pressure sensors were used for the project. Krishnamurthy et al [38] deployed a sensor network for predictive monitoring of equipments in a semiconductor fabrication lab and abroad an oil tanker

in north sea. The network was deployed for 19 and 6 weeks respectively. Accelerometers were used as sensors to collect vibration data from the two sites. Both Intel's imote [49] and CrossBow's Mica2 [42] platform were used for the deployment. The network is tiered network where the motes act as leaf nodes collecting data and route it via a cluster head. The cluster head sends this data to a gateway node which is a Stargate node [32]. The gateway transfers the data using 802.11 radio, ethernet and over internet to the data collection point. Werner-Allen et al [26, 27] use a WSN deployment for monitoring volcanic eruptions using low-frequency acoustic sensors. In this work [27] a sensor array of 16 nodes were deployed over 3 km and collected data for three weeks. Due to the nature of event (spontaneous and unpredictive) the system was used with 100% dutycycle i.e. sensor nodes and sensors are always on. The nodes are deployed in a manner to form a linear topology, each node was fitted with an external antenna and the last hop is using a low frequency long distance link. The data was logged on the sensor nodes and an event detection method (using thresholds) detected for interesting events. An interesting event is reported to the base node. If sufficient number of nodes report this the base initiates a data retrieval node-by-node. The data is kept in the memory till all of it is not transferred and any further logging is stopped.

A comparative table for a few well documented projects with long term deployment of WSNs is provided in Table 3.1 Most of these applications are tiered in nature. This may be due to scalability reasons or because of special service offered by the higher capability node. The single hop distance can vary from few meters to hundreds of meters. Thus different types of antennae are used. Depending on application needs the battery usage differs. In most of the applications battery is chosen based on the desired form factor, duration of battery replacement, cost and availability for recharge. Li ion batteries offer the best energy density where as alkaline batteries are the most cost effective.

3.1.2 Structural Health Monitoring using Wireless Sensor Networks

A number of architectures have been provided in literature for SHM using WSNs. In [46] Xu et al have proposed an architecture called WISDEN data acquisition system to stream MEMS based accelerometer data. The system has a reliable data transport protocol using end-to-end and hop-by-hop recovery. It implements a run length encoding based lossy compression mechanism using thresholds. Also a low overhead data time stamping protocol is mentioned which records the resident time of packets in each node and then calculates the time stamp at the base node. A separate 16bit ADC resolution, accelerometer data collection card is used with an on board microprocessor. The nodes do not perform duty cycling and continuously monitor and stream data. They also study the use of wavelet based data compression techniques

	Habitat Monitoring	Predictive Monitoring	ZebraNet	Volcano Monitoring	NetSHM	BriMon
Deployment Period (Documented)	4 months	19 weeks & 6 weeks	3 weeks	3 weeks	Short Term	Long Term
Sensors Used	Temperature, Humidity & Barometric Pressure	Accelero-meters	Global Positioning System (GPS)	Seismoacoustic sensors	Accelero-meters	Accelero-meters
Mote's Radio Range	10s of meters	10s of meters	100s of meters	100s of meters	10s of meters	100s of meters
Power Source (Battery type)	Li ion	Alkaline	Li ion (Recharge-able)	Alkaline	N.A.	Li ion (Recharge-able)
Time Synchroniza-tion Method	TASK	None	GPS	FTSP	FTSP	FTSP
Data Collection (Period)	Periodic (5 & 20 min)	Periodic (7 & 21 hrs)	Periodic (8 min)	Continuous & Event Based	Continuous	Periodic & Event Based (weeks)
Architecture	Tiered	Tiered	Flat	Flat	Tiered	Tiered
Mote Level Computing	Raw data Collection	Raw data Collection	Raw data Collection	Event Detection	Raw data Collection	Raw data Collection

Table 3.1 Comparison of wireless sensor network based applications (**TASK**: Tiny Application Sensor Kit, **FTSP**: Flooding Time Synchronization Protocol)

	WISDEN	Golden Gate Monitoring	Predictive	NetSHM	BriMon
Motes used	Mica-2	Mica-2	Mica-2 & iMote	MicaZ	Tmote-sky
Accelerometer characteristics	100 mV/g with 300 μgrms noise	ADXL203 & SD1221	Wilcoxon 786A	$\pm$2.5 g tri axial	ADXL203
Data compression method	Run length coding Wavelet (proposed)	None	None	None	None
Time synchronization method	In network residence time compensation	N.A.	None	FTSP	FTSP
Architecture	Flat	Flat	Tiered	Tiered	Tiered

Table 3.2 Comparison of wireless sensor network based SHM applications

in this setting. The same group has designed the NetSHM [36] architecture which is a programmable and re-usable architecture which can evolve with the change in sensor network technology. The system is designed such that the application level programing of SHM related algorithms can be done in C/MATLAB transparent to the intricacies of underlying sensor network. The requirements of these applications is broken down into tasks for the sensor nodes. Thus an interface layer can be designed such that changes in the underlying sensor network techniques, algorithms and hardware or application layer programs do not effect each other. The authors also implement a damage detection and localization method to test the applicability of the system. A few other projects [21, 17, 58, 43] have also addressed the use of WSN in MEMS accelerometer based SHM studies but they stress more on the mote building aspect rather than the complete structure of the application. Table 3.2 gives a comparison of a few WSN based SHM applications.

3.1.3 Use of MEMS Accelerometers

For any sensor network application, the first constraint is the availability of low powered sensors. MEMS based accelerometers are getting used in many SHM based applications [21, 17, 58, 43]. The DuraNode [17] and Smartmote [43] projects use single MEMS sensor to capture the vibrations. Kim et al [58] on the other hand emphasize use of two MEMS accelerometers (SD1221 [55] and ADXL202 [3] for capturing different range of frequencies. They show using shake table tests that SD1221 should be used to capture lower amplitude data (< 0.1 g) due to its low noise and greater sensitivity features, where as the ADXL202 is used to capture higher frequency vibrations due to its better performance in more dynamic situations.

3.2 Time Synchronization

Traditionally Network Time Protocol (NTP) [20] and its variants have been used in distributed networks for maintaining synchronization amongst independent computers spread over the internet with microsecond level synchronization. NTP achieves this by having a hierarchy of servers and clients which exchange time synchronization specific messages. Although NTP is highly scalable, robust to failure and self configurable in a multi hop environment, it fails when translated to sensor network domain. NTP becomes infeasible for application to sensor network as it does not have energy efficiency features built in it. It is modeled on an *'always on'* philosophy where the servers or relays are assumed to be always available. This is not true for sensor motes which depending on the nature of application can have duty cycle less than 1%. Moreover due to mostly unpredictable nature of the wireless medium NTP can introduce errors in tune of hundreds of milliseconds.

The reason for error in time synchronization using naive round trip time calculation based approaches such as NTP is due to uncertainty and non determinism in data transmission over wireless channel. Newer algorithms try to estimate these delays and compensate for them. Message delivery delays have been listed in [34, 54]. In general we can divide the delays into three classes

- Sender side delays (send time, channel access time, transmission time)
- Propagation delays (propagation time) and
- Receiver side delays (reception time and receive time).

Maroti et al in [44] lists additional delays such as interrupt delays (on both sender and receiver sides), and encoding and decoding delays. Amongst these the transmission time, propagation time, reception time and the encoding and decoding delays can be estimated easily and accurately. Send time and receive time are dependent on the system call overhead. This in turn depends on processor load at that time or whether the interrupts were disabled by some program. The maximum uncertainty is introduced due to channel access delays as they can range from 0-500 ms and cannot be predicted. To overcome these time stamping the message during transmission and reception time is used in [54, 44]. Elson et al [34] eliminate the sender side uncertainties as well as propagation delay by using receiver-receiver synchronization. For tighter synchronization FTSP [44] uses averaging of time stamps obtained after every byte transmission/reception of SYNC bytes in the data packet. This effectively removes the interrupt introduced uncertainty.

The available time synchronization protocols can be divided into two major classes: receiver-receiver synchronization [34] and sender-receiver synchronization [20, 54, 44]

based protocols. Both classes of protocols perform pair wise synchronization differing only in the way they try to eliminate the uncertainty and indeterminism from the system. In the former class of protocols exemplified by Reference Broadcast Synchronization (RBS) two receivers synchronize their clocks based on the common observation of senders broadcast. This inherently removes the sender side uncertainty. The second class of protocols use MAC level time stamping for this purpose exemplifies by Time-sync Protocol for Sensor Network (TPSN) and Flooding Time Synchronization Protocol (FTSP).

RBS exploits the property of the broadcast medium that irrespective of the uncertainty in the send and channel access time for the sender, time of packet reception is the same with respect to all the receivers. Thus any message received by all the receivers falling in a particular nodes transmission coverage area can use the message as a synchronization point and record the local time at that moment. This removes any sender side uncertainty and only leaves propagation and receiver side delays which can be estimated. By further message exchange the receivers can synchronize amongst themselves using these synchronization points. It further uses least squares linear regression over the offset values with respect to local time to calculate the clock skew.

TPSN and FTSP on the other hand use the conventional method of sender-receiver message exchange for time synchronization. They use MAC level time-stamps to remove the uncertainties in the pathway. In TPSN a node `A` time synchronizes with a second node `B` by sending a time stamped message. The receiver (`B`) sends an *'acknowledgment'* against this message. The original message is time stamped at the transmission time `T1`. The *'acknowledgment'* contains time stamps for reception of the original message `T2` and when *'acknowledgment'* is transmitted `T3`. On reception it is again time stamped `T4` at `A`. Assuming symmetry we get $drift = ((\texttt{T2} - \texttt{T1}) - (\texttt{T4} - \texttt{T3}))/2$ and $propagationdelay = ((\texttt{T2} - \texttt{T1}) + (\texttt{T4} - \texttt{T3}))/2$. These values are used to correct the time on the node `A` according to time at `B`.

In both RBS and TPSN protocols a number of delays remain unaccounted for thus they cannot give very tight synchronizations. RBS fails to remove jitters due to interrupt handling and decoding times which can introduce errors in the range of 5-30 μs and 100-200 μs. TPSN on the other hand does not estimates the clock drift accurately (it uses only one value while clock skew needs statistical estimation). Also in both these protocols there are additional message exchange required prior to synchronization. FTSP performs better by utilizing the good features of both the protocols. It uses broadcast nature of the medium to disseminate time stamps, sender-receiver type model for updates and regression based clock skew estimation. It also perform MAC level time stamping and betters it by averaging the time stamps put in the messages to remove interrupt handling related jitter errors.

For multi-hop networks RBS and TPSN require additional provisions to achieve network level time synchronization. RBS uses time translation by node falling in

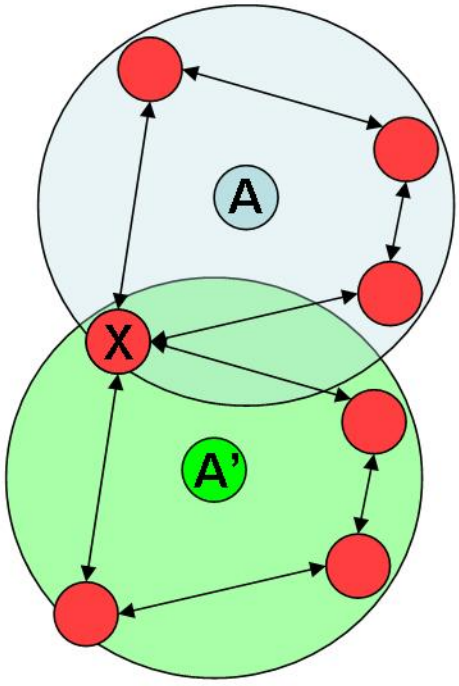


Figure 3.1 Multi-hop Synchronization using time translation in RBS

ranges of two time domains (sender's transmission range where a set of receivers can synchronize) E.g. Node X in Figure 3.1. Here the small circles are the sensor nodes and the bigger circle is the transmission range for `A` and `A'` respectively. The arrows indicate synchronization related message exchange amongst receivers in each domain. TPSN on the other hand maps a spanning tree over the network to divide it into levels (Figure 3.2). Nodes in each level synchronize with their parent in the level above. Both these features hinder the scalability of the protocols. Also in case of TPSN node mobility can draw considerable latency in synchronization of the node. Any node orphaned due to its parent node dying (or leaving the network) in the level above or any new node joining the network, need to wait till the network restarts level discovery (spanning tree construction). FTSP overcomes this by forming an ad-hoc topology to disseminate the time-synchronization packets. Since the message is broadcasted, multiple receivers can get synchronized at same time. Once synchronized these nodes can send there estimates for the global time further down the network. This approach is highly scalable and allows graceful handling of synchronization with mobility of the nodes. In all these protocols the periodicity of the synchronization related packets is dictated by the degree of tolerance for time-synchronization error acceptable by the application. Table 3.3 compares the three protocols.

In this work we are using FTSP to achieve time-synchronization. We have per-

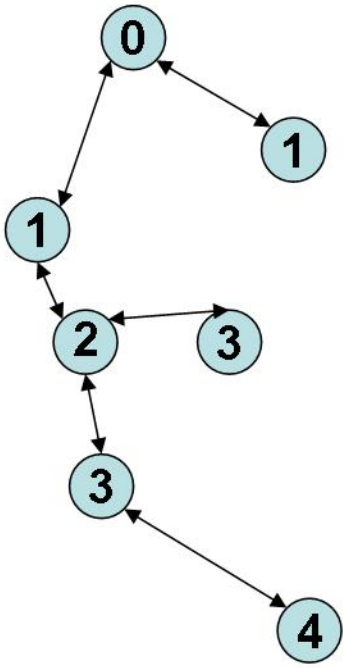


Figure 3.2 Spanning tree and level structure in TPSN

	RBS	**TPSN**	**FTSP**
Synchronization method	Receiver-Receiver	Sender-Receiver	Sender-Receiver
Property used	Broadcast nature of medium	MAC level time stamping	MAC level time stamping
Scalability	Small	Medium	Highly Scalable (dependent on permissible error)
Synchronization Error - single hop: Average (Maximum)	29.1 μs (93 μs)	16.9 μs (44 μs)	1.48 μs (6.48 μs)
Synchronization Error - multi hop (4 hops): Average (Maximum)	3.68 μs (10.7 μs) (using 802.11 radio)	21 μs (N.A.)	2 μs (9.3 μs) (estimates)

Table 3.3 Comparison of existing time synchronization protocols

formed modifications in the current available implementation to make it more suitable for our application scenario. Details are given in Section 5.2.1.

Approaches used by other WSN implementations

WISDEN [46] uses a consistent time stamping method at the base station instead of network wide time synchronization. In this scheme the nodes in the network calculate the resident time for each packet it spends time inside a node. Since propagation delay for radio packets falls in nano seconds range it is ignored. Thus if t_A^i is the resident time at the i^{th} node for a packet from `A` and T_A is the total resident time for the packet in the network before reaching the base then $T_A = \sum_0^n t_A^i$, here n is the number of nodes between root and `A`. Hence if the base node receives the packet at time τ_A then the time for generation of the packet is $\tau_A - T_A$. This approach does not require network wide establishment of a global time but there are number of problems with this approach. Although this algorithm can remove some sender and receiver side inconsistencies by using MAC level time stamping (not implemented) it does not remove all the sources of delays and error. First it does not take care for the clock skew between the nodes which can be in the range of 0-100 μs per second [34]. This gets accumulated for any packet as it spends time in the network. Also errors due to interrupt handling can creep into the calculations depending on the load on the nodes of the network.

The Zebranet [48] project uses low power Global Positioning System (GPS) modules for time stamping the data collected. Since for this work GPS module is used as a sensor as well to determine the positional information it also provides accurate time stamping device. In [25] Tolle et al and in [52] Szewczyk et al have used Tiny Application Sensor Kit (TASK) for achieving time synchronization of the nodes. Flooding Time Synchronization Protocol FTSP [44] has emerged as the protocol of choice for newer applications developed. It offers better estimation and subsequent correction for the delays and errors in the pathway of time synchronization. It has been adopted by Chintalapudi et al [36] in the NetSHM system architecture for SHM. Allen et al [27] use it in the monitoring application for volcanic eruptions.

3.3 Event Detection

Since SHM studies can be preplanned, most of the current methodologies do not have an explicit event detection mechanism built into them. In the wired methods since power is not an issue the sensors are either kept on all the time or started at the time of data capture using a mechanical switch. For the WISDEN application the authors do not mention duty cycling, hence we can assume that the equipment is kept on all the time and is streaming continuous data. In this case event detection is built in

order to perform compression of data prior to transfer. The nodes perform a basic run length encoding based on thresholds before performing wavelet based compression. In the Predictive Maintenance (PdM) application built by Krishnamurthy et al [38] since the application does not perform compression or mote level analysis hence no event detection mechanism is performed. The data is collected periodically on a timer expiration. In the volcano monitoring deployment, Werner-Allen et al [27] use threshold based event detection technique to mark event as interesting. If a group of nodes label a data set as interesting then the base starts data collection. Chintalapudi et al [36] describe both triggered and triggered with local computation models for accelerometer based viberation data collection. The former class of methods turn-on the sensors to perform data collection when it is triggered either by manual command, a timer fire or after significant ambient vibration. The latter method is similar except the data transmitted it analyzed locally on the motes and only relevant part (modes of FFT or significant vibration data above certain threshold) is transmitted. For detection of change in ambient vibrations when the mote is asleep the concept of using a secondary low power but course grained sensor is proposed by them. In our case we detect the arrival of the train using beacons sent by the train using Wake-on-WLAN [45] mechanism. Once the train is detected the accelerometers are turned on for data collection.

Chapter 4
Application Design: BriMon

In this chapter we give the complete Bridge Monitoring solution named as BriMon. The details are qualitative and where necessary we have shown why a particular design choice was necessary. First we describe the requirements and associated constraints derivable from the problem statement. Next we describe the BRIMON solution approach and give qualitative details about various modules. The various design choices and their necessity is explained in the subsequent section followed by discussion.

4.1 Problem Definition

Recall from section 1.1 that the goal of the system is to record the structural response of a railway bridge toward different types of vibrations. These vibrations are recorded using accelerometers attached with the piers or other strategic locations of the bridge to be designated by an engineer/consultant. Depending on the length and design of the bridge these accelerometers can be separated by 5-60 m. We need to recover these accelerometer generated data with high reliability and fidelity. For frequent data acquisition the equipment needs to be left on site and be capable of autonomous and on-demand data collection.

Thus any proposed solution needs to provide these features. Looking into details we observed that this solution needs to fulfill some additional requirements which come into picture because of the problem domain (SHM), suggestions from end users and usage scenario. Given below is the list of these requirements and constraints.

4.1.1 Requirements

- The accelerometer data collected from different accelerometers as well as different axes of given accelerometer need to be time stamped against a common global clock. This is required to correlate values in data collected from different independent accelerometers and axes.

- The equipment once deployed should have a long maintenance cycle because it could be used in far reaching areas with difficult and costly manual access. E.g. A single lane bridge on a deep gorge.

- The form factor for the complete module should be small and one that could be hidden easily below the bridge and difficult to catch casual eyes.

- SHM based techniques utilize vibration data with frequency < 50 Hz. For larger structures this comes to the range of 0.1-20 Hz. This is due to the high damping offered by the structure. To reduce the effect of noise and high damping of the structure, sampling frequency of 10 times larger than the most dominant mode is suggested [36].
- The minimum resolution desired is of 0.01 g.
- Data transfer must be reliable because even if a small chunk of data is lost it can result in wastage of complete data set for the cycle.
- Large scale deployment of any proposed scheme requires substantial automation and low cost.
- Data needs to be transferred to the data repository which need not be situated at the remote bridge site.

4.1.2 Constraints

The above requirements put the following constraints for the solution

1. The maximum allowable error for the time synchronization is in the single digit millisecond range.
2. Long maintenance cycle means there is limited energy for any equipment. Thus the equipment used needs to be low power and must save energy when not performing monitoring operation.
3. Small form factor requires using high density Li-ion polymer/alkaline batteries although such batteries can increase the cost of each node.
4. For sampling at 10 times the most dominant mode, we need to sample the accelerometer at frequency ≥ 200 Hz on each channel.
5. The accelerometer's resolution must be such that it is able to show a change for 0.01 g. Also the Analog To Digital (ADC) converter used must be able to convert the data at smaller resolutions than 0.01 g.
6. Self healing is required for autonomous operations. This requires built in measures to take care of component or equipment failure.
7. Since we need to gather data for all the three types of vibrations, lack of mechanical shakers leads us to use the train as a source for vibration. To capture the forced vibration we need to record data during the passage of train over the

bridge, for free vibrations we need to log data when the train has crossed the bridge and finally some routine recording to gather ambient data.

8. Distance between bridge and data repository spans over hundreds of km. To keep the cost low we need solutions other than satellite or long distance radio.

In Section 1.3, we noted that current solutions are primarily wired, with bulky design and require skilled man power for installation. These equipments are costly and are not designed to work autonomously. Their deployment is cumbersome, and requires lots of supplementary planning such as cabling and power supply related work. All these points has lead us to explore alternative solutions which could replace some if not all of the current solutions. We have designed the proposed solution using WSN and MEMS based low power accelerometers. This solution tries to cover the constraints thrown by the problem requirements and due to the use of wireless sensors.

Wireless communication channel is highly unreliable and ridden with data corruption and packet loss. Any system designed for WSN must be capable of recovering lost data for ensuring reliability. For Tmote-sky platform it turns out that radio usage for listening and transmission are roughly of the same order and equally costly energy wise. The current platforms do not have large memory banks, hence solution must be devised keeping a balance between data collected on the node and removing it by transferring to a larger storage site. The radio offers limited bandwidth for data transfer hence there must be enough time provision in the system to gather data from all the nodes before the next cycle begins.

A large portion of the high cost associated with the installation and usage of SHM instruments is due to the requirement of skilled engineers for this purpose. Any newer solution which could provide easy handling and deployment stands to get an upper edge over the other available methods. Similarly if the deployment requires programing and other similar facilities it must again be easy to learn and perform. Also for long term deployment the motes are likely to face harsh environmental conditions such as rain, wind, mechanical shock etc.

4.2 BriMon

We use a bottom up approach for our application design. All the modules are designed to work with the constraints of the domain and usage scenario. Optimizations implemented for the solution are application specific. We have listed a number of constraints above. In the next chapter on implementation we will show how these constraints are met by our implementation and application architecture.

An overview for the application can be taken from Figure 4.1. The messaging and transporter module is aboard the train and is used for beaconing and data collection from the bridge site for transportation to repository. An upstream frontier node is

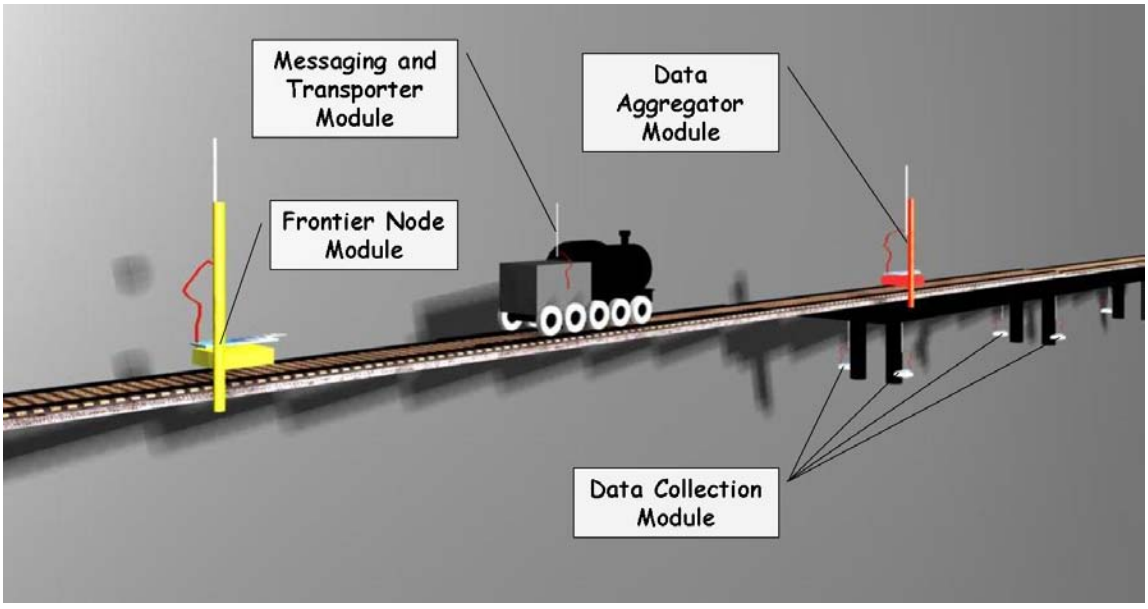


Figure 4.1 Location of different modules for BriMon application

used for early detection of the train. Data Aggregator node at the beginning of the bridge, which also takes up the role of a base node for the sensor network deployed on the bridge, acts as a temporary storage site for data and also transmits it to the train. The set of data collector nodes placed on the piers of the bridge are for collection of accelerometer data. The train and the data aggregator communicate over the 802.11 radio. The 802.11 packets are also used for detection of the arrival of the train. This event detection is crucial to our application from three view points:

1. **Data generation:** Since there are no independent source for generating forced vibration into the bridge, the passage of the train is used for this purpose. Once the train has passed, the residual vibrations is collected for free vibration analysis of the structure.

2. **Sensor turn-on switch:** The accelerometers are used only when data needs to be collected. Since the train's passage is the source for vibration to be collected by these sensors, arrival of the train is used to wake up the accelerometers.

3. **Long distance data transport:** To transport the data from the remote location of the bridge to the analysis center or the data repository we use the train as the transporter.

Details of these modules follow the event wise description.

4.3 Event Sequence

The nodes interact with each other for three purposes, data acquisition cycle, time synchronization cycle and routing cycle.

4.3.1 Data acquisition cycle

Data acquisition cycle is characterized by detection of train's arrival, turning-on the accelerometers for data collection, collection and time-stamping of data at the collector nodes and reliable transportation to the base node. These are described in an event wise fashion below:

- The train carries a 802.11 radio and initially keeps transmitting beacon packets for informing its arrival to the sensor nodes deployed at the bridge.
- The frontier node monitors the channel for the incoming train. It wakes up periodically to listen for channel activity. It detects the 802.11 beacon messages using the Wake-on-WLAN [45] mechanism.
- On detection of an incoming train the frontier node notifies the base station by continuously sending beacon towards it.
- The data aggregator node (base station) does duty cycling. It listens for the beacon packets generated by the frontier node for notification of train's arrival. If it detects the train arrival beacon, it calculates a time interval after which the collector nodes need to start the accelerometers. This interval information is written in a packet and it is flooded in the data collector network downstream. Additionally this beacon also carries the time interval for which the data needs to be collected.
- The data collectors do duty cycling and periodically listen for any beacons from the base node. These beacons can be either for notification of arrival of train, routing or time synchronization.
- If the beacon is for notification of train's arrival the node reads the time interval marked on the packet and transforms it to a local time value and sets a timer accordingly to start the accelerometers for data collection. A second timer is started to turn off the data collection process using the additional information on the beacon packet.
- On timer expiration the nodes turn on the accelerometers and sample the data. This data is written to the data memory of the node.

- On expiration of the second timer the nodes turn off the accelerometers and initiate the reliable data transport phase.
- The transport protocol offers end-to-end reliability as demanded by the application using hop-to-hop reliability. Also this protocol is initiated by the leaf nodes (i.e. nodes at the fringe of the network) and follows the store and forward strategy .
- Once data from all the nodes is collected at the base node, it updates the information regarding the next wake up duration and puts them to sleep.

4.3.2 Time synchronization cycle

We are using Flooding Time Synchronization Protocol (FTSP) for time synchronization. The time synchronization cycle is initiated by the base node. Here we are trying to synchronize the whole network with the time of a single node i.e. the root. The root sends periodic beacons which get used for estimation of the root's clock by other nodes. The non-root nodes once having sufficient number of estimation points start sending their estimates of the global time (i.e. the clock of the root), which are used by nodes further down the network for their own estimations. The periodicity of time synchronization beacons is dependent on the amount of time synchronization error acceptable by the application. The base node follows a dynamic beaconing method to perform synchronization. Initially when the nodes are unsynchronized the base node aggressively floods the network with time synchronization beacons. After a certain duration it decreases the beaconing rate assuming that all the nodes are synced. For time synchronization the cycle can be broken in to following events:

- The base station starts the process by transmitting the beacons marked with its time stamp (global time). This stamping is done just prior to sending the packet on air (i.e. after its MAC gets access to send packets)
- When any collector node receives a beacon it puts it's own local time in the packet just when the radio receives the packet. This pair of global-local time pair acts as a synchronization point for the algorithm and is used to calculate the offset using
$$Offset = Globaltime - Localtime. \tag{4.1}$$
- The time synchronization module creates a table of most recent beacon messages containing the global-local pair. These messages are used for calculating the clock skew of the mote with respect to the global clock. This table gets updated and refreshed as and when the node detects inconsistency or finds the information stored as stale.

- The information stored in the above mentioned table provides the offset and the skew which are used to get the global time estimate using the following set of equations.

  We have $Skew$ equal to the slope of the graph between offset and localtime. i.e.

  $$Skew = \frac{Offset - Offset_{Average}}{Localtime - Localtime_{Average}} \tag{4.2}$$

  This gives

  $$Offset = Offset_{Average} + Skew * (Localtime - Localtime_{Average}) \tag{4.3}$$

  Thus combining 4.3 and 4.1 we get

  $$Globaltime = Localtime + Offset_{Average} + Skew * (Localtime - Localtime_{Average}) \tag{4.4}$$

- When the node decides that it has sufficient number of synchronization points it starts sending out beacons marking it with its own global-time estimates.

- A node has a timer counting, which is reset after each successful beacon reception. If a node does not receive any packet withing this timer value it initiates the route detection mechanism.

4.3.3 Routing Cycle

During the initial phases the routing algorithm runs to detect and form the routes in the network. Once established the root sends periodic routing beacons to keep the node aware of the route. If a node does not receive these beacons for a certain period it initiates a route detection mechanism. Here we have divided the network into parent-child relationship which also gets used for the reliable-transport protocol. The details for both routing and transport mechanism are provided in my colleague Hemanth's work [28].

4.4 Hardware modules for BriMon

Hardware wise there are four independent modules in the application design. These modules are located at different locations and have assigned roles.

4.4.1 Messaging and transporter module

This module is located on the train. It has a fully functional 802.11 radio for data transmission. The radio is switched on prior to the approach to the bridge and sends beacon messages at a specified interval. These beacon messages are used to notify the frontier node about the trains arrival.

4.4.2 Frontier node module

This module is located upstream from the bridge. The role of this module is to perform early detection of the arrival of a train carrying the messaging and transporter module. This is early detection requirement comes due to long boot-up time of the Soekris box. This node has the ability to detect 802.11 channel activity using 802.15.4 radio on the lines of Wake-on-WLAN [45] mechanism. Once the train is detected, beacons are transmitted to the base node. Since the communication is essentially one sided, reliability is achieved using repeated message transmission. In light of the ranges achieved using a sector antenna (> 800 m), we can mearge the frontier node functionality with the sensor mote at the data aggregator node.

4.4.3 Data aggregator module

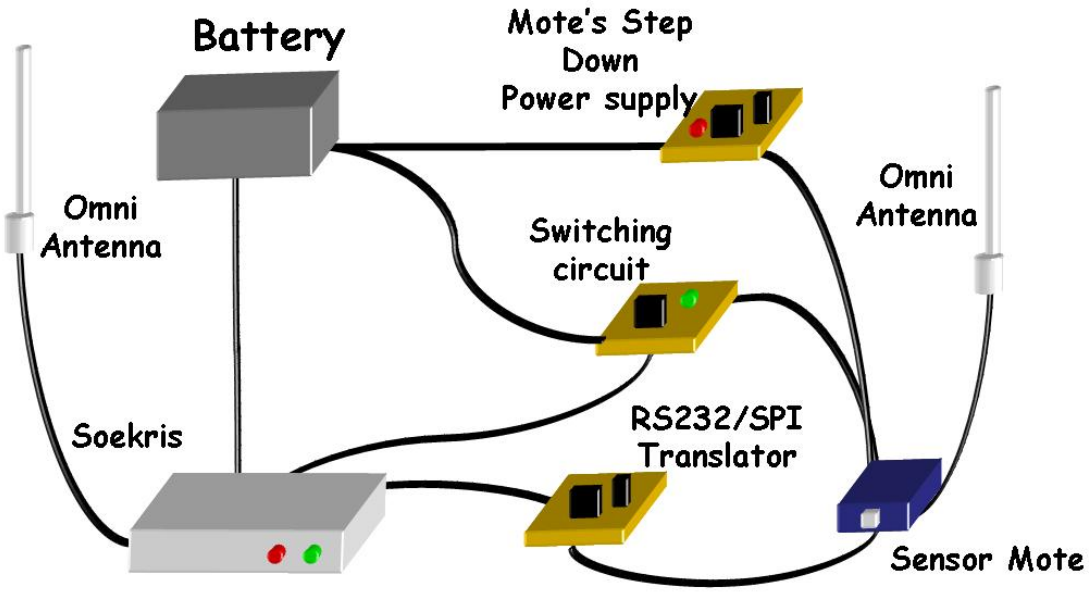


Figure 4.2 Schematic diagram for a data aggregator node

The module is located at one end of the bridge. It has the capability to talk to both 802.11 and 802.15.4 radios. The module acts as the base node for the network of sensor nodes located on the bridge. It also acts as a data storage for collected data before transfer to the train. To achieve high data transfer rates the node is equipped with 802.11 radio. The node is tasked to keep the collector nodes time-synchronized,

maintain the routing and command the nodes to initiate data transport as well as initiates data transfer to the train. The duty cycling of the node depends on the latency of trains arrival after detection by the frontier node, boot-up time of the Soekris (single board computer), duty cycle for the data collector motes, and periodicity of time-synchronization and routing protocols. This module can be connected with a GPS module to get accurate time and data information at the remote site. Figure 4.2 shows the schematics of a data aggregator node with the Soekris box, sensor node, battery, protocol translator circuit and switching circuit.

4.4.4 Data collection module

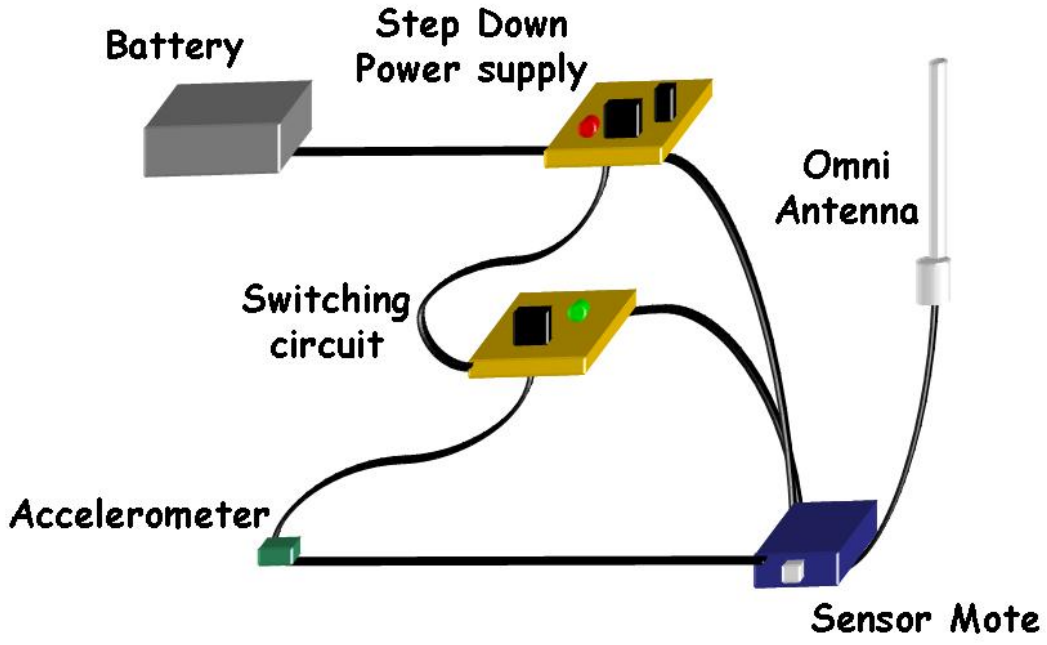


Figure 4.3 Schematic diagram for a data collector node

The module is located on the piers and other critical locations of the bridge designated by the structural engineer or consultant. The module has an accelerometer connected with a mote. The mote switches on the accelerometer to collect vibration data. This module acts as a slave to data aggregator module. All the functionalities are timer driven and are controlled by the data aggregator module via message ex-

change. Each data collection module logs and time-stamps the data. It is also a part of the multi-hop network, and hence acts as router for packets generated or consumed by nodes downstream. The data collected is transmitted to the base node using this multi-hop network. Data for a given node is only transferred when it collects all the data from its children. The nodes are time-synchronized with the base node via FTSP protocol. They also participate in the route detection mechanism during the routing cycle.

The sensor mote does duty cycling and is asleep most of the time. The duty cycle for the node is dependent on the periodicity of the time-synchronization protocol, latency in detection of train's arrival and stabilization time for the accelerometer. The sleep cycle in this case is scheduled by the root mote (data aggregator module) to get synchronized wake up amongst all the deployed collector nodes. The nodes wake up periodically to listen for beacon messages from the root node. These beacon messages are for route maintenance, time synchronization or command beacons indicating the nodes in the network to start collection of data after a given period of time. Once it receives the beacon, the mote first finishes the indicated job and then reschedules the sleep cycle. Since the root does not reach all the nodes in the network the beacon packets are forwarded by all the nodes after required modifications and processing. Figure 4.3 shows the schematic of a data collector node with individual components such as accelerometer, sensor node, battery and switching circuit. Current platforms (Tmote, MicaZ) do not support low power listening, and the energy consumed for idle listening is comparable to transmission energy. Thus duty cycling and synchronized wake up provide better power savings.

Figure 4.4 gives the different wireless communication protocols used amongst the modules.

4.5 Software modules for BriMon

Overall the approach is broken down into four independent modules of routing, transportation, time synchronization and event detection. On top of these sits the application which uses these modules for the solution. The time-synchronization, event detection and scheduling methods are explained next. For discussion and details regarding the routing and transport protocols please refer to [28].

4.5.1 BriMon Time-synchronization: FTSP

For bridge monitoring application to find the natural frequency for the bridge the time synchronization error desired is in the range of few milli seconds. The naive solution approach is to put the global time of the root and flood the network with these packets. Nodes can read from the packet and calculate the offset to update

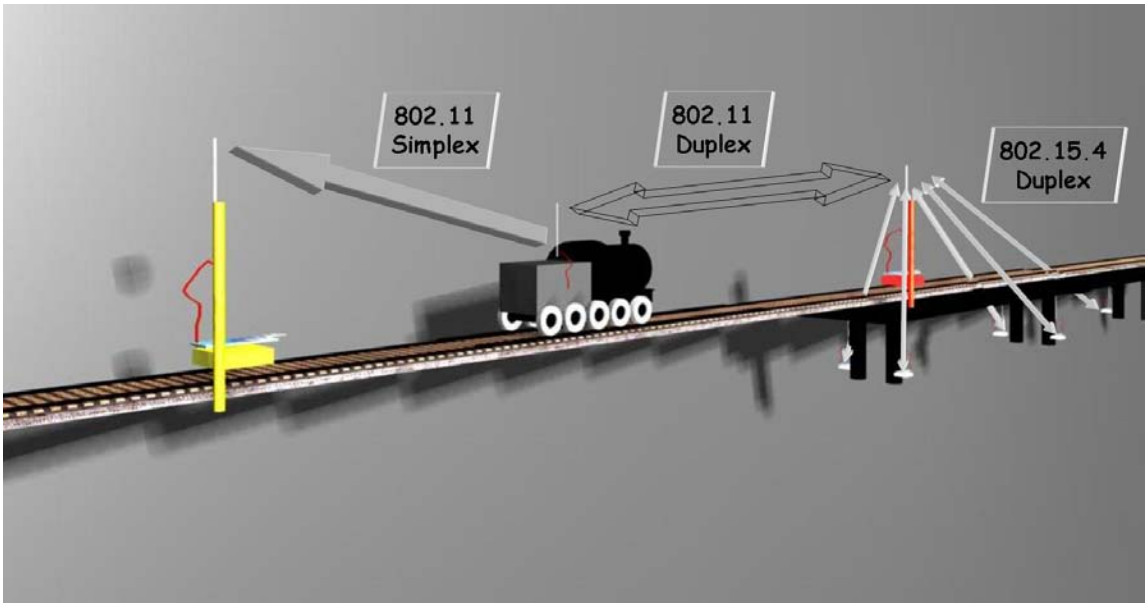


Figure 4.4 Data communication protocols used between different modules

their clocks. It turns out that this approach will encounter substantial errors on the message pathway. As mentioned in [44] the packet encounters milli second delays for channel access and send and receive times. Thus we will need to do better than this. Looking at the available protocols NTP, RBS, TPSN and FTSP come into picture. For NTP no sensor network implementation was available and also it will fail to register delay values just from the round trip time estimations as channel access time is not predictable. RBS although implemented in sensor network was not used due to its indirect way of keeping global time synchronization. Also its drawbacks have been listed in [54]. Thus we need to decide between TPSN and FTSP. The reason for choosing FTSP over TPSN was based on better results reported in literature [44], wide usage amongst different applications [27, 36] and availability of implementation code in open source public domain. Also we have made appropriate changes to this implementation to better suit our application scenario.

Flooding Time Synchronization Protocol (FTSP) [44] uses network flooding to disseminate timing information of the root node. Each time-stamping related packet is time-stamped at the time of transmission and reception at the sender and receiver respectively. Thus it overcomes with the send, access, and receive time delays. The propagation delay is negligible and interrupt delays are handled by averaging time-stamps generated at the time of sending and receiving SYNC bytes. Since the other delays mentioned in Section 3.2 can be estimated very accurately, these measures provide tight time estimation. For clock skew compensation it uses the least square

regression method from [34]. Although the original design available as TinyOS application has provisions for graceful transition from one root node to second node and other root selection methods, in our setting these are not used. Flooding is a direct consequence of broadcasting which is exploited to synchronize multiple receivers with the sender. FTSP forms an ad-hoc structure in the network which is dictated by transmission ranges of the nodes. Although it can be argued that flooding is an inefficient mechanism, in this case this need not be the case. To achieve any form of reliability, a node needs to know who all it is in touch with. This requires imposition of a structure on top of the network which requires additional packet exchange between the nodes. For achieving tight synchronization the nodes have to periodically update information in any case. Thus, instead of this two tier approach, FTSP's approach is for directly sending time synchronization related packets over the network. It can be argued (not provided here) that this will be more efficient in terms of number of messages used than link-by-link or node-to-node synchronization approaches such as TPSN and RBS.

The protocol described in it's current form [44], flushes the synchronization point table having the global-local time pairs after a certain time interval. This is done in order to refresh the values being used to calculate the clock skew. In our experiments with the code ported for Tmote-sky sensor node platform it turned out that this approach created trouble to synchronize nodes as well as overall synchronization stability in the network. We observed that a number of packets were carrying only the offset and skew compensated value in there global time estimate. Due to faster transmission of data or due to more load on the microcontroller the local time was not added to the timestamp as is required in Equation 4.4 for the global time estimate. Thus the root node was transmitting all zeros as its global time estimate occasionaly (this is because difference for global and local time at root node is zero). This phenomenon was also observed at other nodes which suddenly sent a packet with either a very small or else a negative value. Packets of this kind were inconsistent and hence resulted in flushing of the synchronization point table. This was not a periodic happening but was occurring randomly throughout the network.

We changed this approach to a newer method where instead of cleaning the table, we create a circular array of synchronization points. Thus newer values replace the older ones. This approach worked and has been demonstrated in the results section. Our justification for the approach is that since our application is tolerant to time synchronization errors in the single digit milli- second range we can relax a few constraints built to achieve μs error rates. Also since the short-term stability of the crystal is more or less constant, newer values do not affect the clock skew values derived earlier except to remove the jitter. This is achieved in our case as well albeit with few older values. Any error introduced on the other hand will get removed with replacement by newer values.

It can be argued that the μs error rate is an overkill, but we believe it has future use. Although right now we are using the accelerometer data for natural frequency detection, the same data can be used in other SHM applications such as damage detection and localization. These usages for data demand better synchronization and tighter error rates [46, 36]. Thus if the provisions are present the data can be used as it is without requiring recollection.

4.5.2 BriMon Event Detection: Wake-on-WLAN

As explained above event detection (arrival of train near the bridge) is essential for our application. We have used Wake-on-WLAN's feature for detection of 802.11 packets and implemented it on Tmote for this purpose. The 802.15.4 specification requires three modes for Clear Channel Assessment (CCA) to be implemented by any compatible radio transreceiver chip. Amongst them the first mode gives a clear channel when the energy at the mote's antenna is below a certain threshold. We use this mode to raise interrupts by the Tmote's radio when the energy is above this threshold. The energy in this case is generated by 802.11 packets transmitted in the 2.4 GHz range rather than regular 802.15.4 packets. Thus we can detect presence of 802.11 traffic around the mote.

Once the mote detects channel activity it can take further action as per application requirements. In our case the 802.11 packets are generated by the messaging and transporter module on the train. The mechanism is implemented at the node being used as the frontier node. Once the 802.11 traffic is detected the frontier node sends beacon messages toward the base station for further action. Since the threshold value which is used to decide for clear channel access is programmable, the sensitivity of this mechanism can be changed as and when required. Additionally, since the application implements an on-demand data generation and retrieval, we collect data only when desired. This is better approach than threshold based event detection methods which would have required the accelerometers to be kept on all the time.

We now argue for the usage of Wake-on-WLAN based event detection mechanism. An alternative method is the usage of mechanical switches. Firstly, we do not have proper domain experience for using them. Next, any new device which needs to be attached with critical equipments (railway tracks) needs proper certifications etc. In our case the proposed Wake-on-WLAN mechanism is a non-contact mechanism and does not disturb any existing equipment. Also we do not incur substantial increase in equipment apart from the frontier node. As we will discuss in the next chapter this additional node is required due to delay caused by booting up of Soekris board rather than one introduced because of usage of Wake-on-WLAN based mechanism.

Another disadvantage with the use of mechanical switch is that, we will not be able to differentiate between two trains, one being ordinary train and other carrying

the messaging and transporter module, without switching on the Soekris. As we will show in the battery power consumption calculations 5.3 that the highest power consumption is by the Soekris box. Thus we will need to start the Soekris with passage of all the trains and this will reduce our deployment's life substantially.

Chapter 5
Implementation

In this chapter we give implementation details regarding various hardware and software modules developed for the BriMon application. First we give details for the four different hardware modules in action namely: messaging and transporter module, frontier node, data aggregation module and data collection module. These modules are made from components such as sensor nodes, single board computers, application specific circuitry etc. which are explained next. In software modules we have the time synchronization and event detection modules. We provide component interaction for the time-synchronization and event detection modules and also explain the power saving features built within Tmote and TinyOS.

5.1 Hardware

In this section we first look at the four modules which constitute the application hardware. Next, we look at individual components of the modules. Then we show how the current set of hardware is able to fulfill the requirements and constraints mentioned in Section 4.1.

5.1.1 BriMon Hardware Modules

In Section 4.2 we gave the qualitative details for the four hardware modules used in the project. Here we give the implementation details with constituents and how they are connected, for the same.

Messaging and Transporter

This module consists of a computing device equipped with an 802.11 radio. The computing component in this module can either be a laptop or a Soekris box. Based on the speed of the train and location of the bridge a sectoral or omni directional external antenna can be used on this module. The computing device runs a script which transmits beacon packets, over 802.11, used by the frontier node for train's arrival detection. It also acts as a beacon message for the 802.11 radio connected to the data aggregator node to initiate data transfer sequence if there is uncollected data stored on the aggregator node. Once a HTTPS over TCP connection gets established the node collects data from the HTTPS server running on the data collector module via a script from a predefined location. We have used HTTPS as it offers authentication and encrypted data transfer. TCP has been used to give reliability to data transmissions.

Frontier Node

This node wakes up periodically to check for channel activity. The incoming 802.11 packets from the train are detected using the Wake-on-WLAN [45] mechanism. The method is implemented on the mote available at this module. If it receives beacon messages from the messaging and transportation module aboard the train, it starts sending messages towards the data aggregator module near the railway bridge. The message for the aggregator node contains the timestamps for the train's arrival (i.e. when it was detected by it) and time of message transmission.

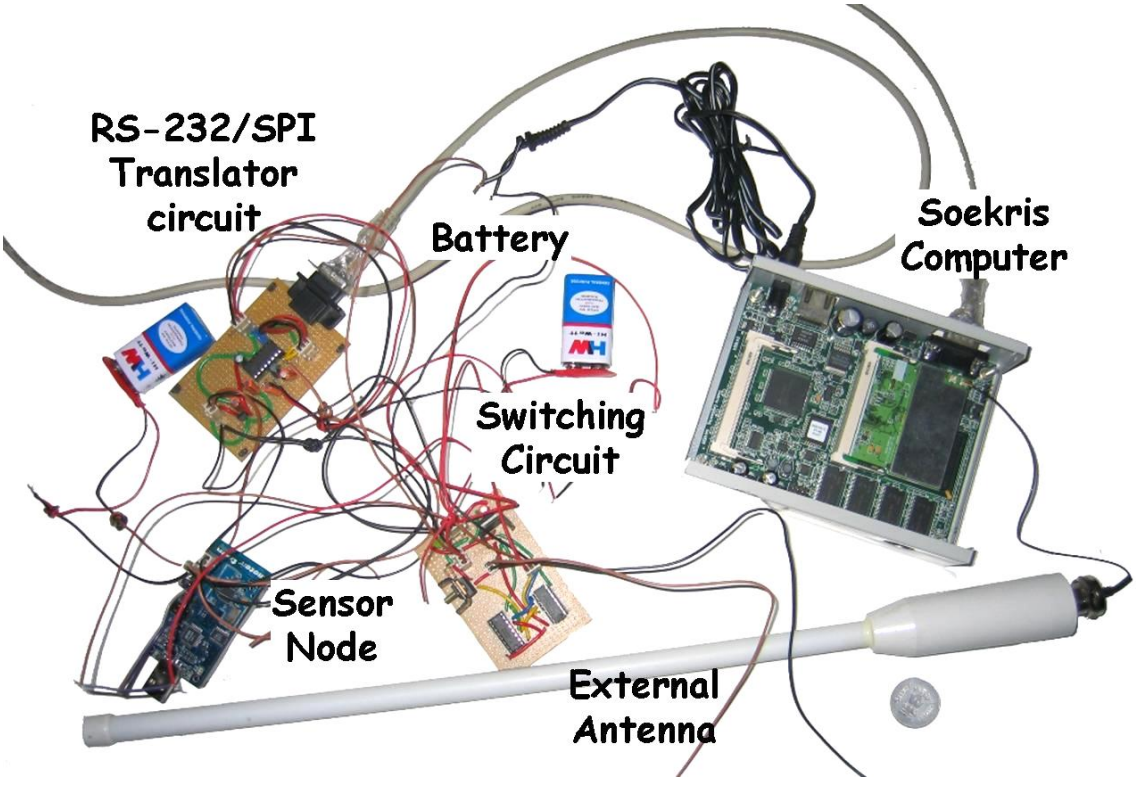


Figure 5.1 A prototype data aggregator node with individual components spread out

Data Aggregator

This node also acts as the root node for the sensor network located at the bridge. The module may or may not be connected with an accelerometer. Figure 4.2 gives the schematic diagram of a data aggregator node. Figure 5.1 on the other hand shows the prototype for the data aggregator node. The data aggregator module has a WSN mote and a single board computer (e.g. Soekris) connected to each other.

The sensor node is connected to the Soekris on pins 2 and 4 of the 10 pin extension header (via a RS232-SPI interface circuit) and to the switching circuit on pins 3 and 4 of the 6 pin expansion header. The switching circuit controls the power flow to Soekris. The mote is connected to external antenna for receiving transmissions from the data collector modules on the bridge and upstream frontier node. The Soekris board can share this antenna or have a separate antenna for 802.11 traffic. Since the node has the capability to receive and transmit both 802.11 and 802.15.4 traffic it acts as a transitory storage for data collected at the collector nodes due to larger flash memory available with the Soekris before getting transported to the repository by the messaging and transport module. Once the node receives beacons from the frontier node it sends appropriate beacon messages to the data collectors indicating the period after which they have to collect data. The module also takes care for time-synchronization and route maintenance of the network.

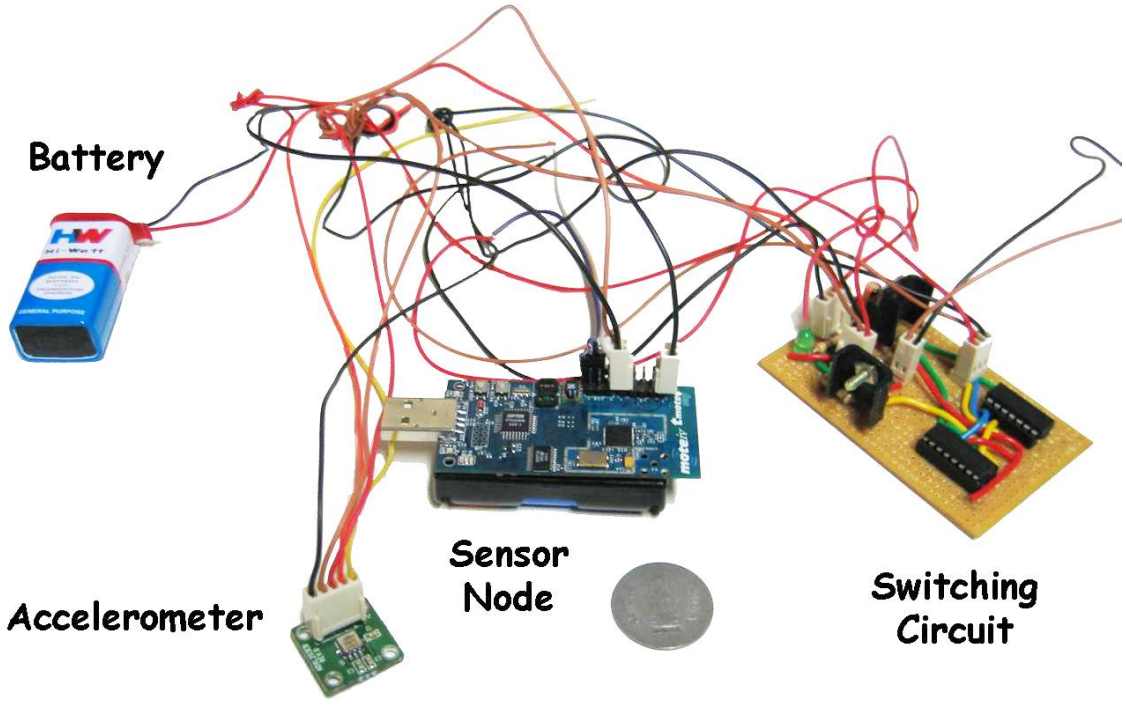


Figure 5.2 A prototype data collector node with individual components spread out

Data Collector

Figure 4.3 gives the schematic diagram and Figure 5.2 shows a prototype for a data collector node. These are the motes attached with accelerometers. The motes are connected to the accelerometers on the ADC pins 3 and 5 of the 10 pin expansion header. The switching circuit for controlling the power flow to accelerometer is connected on the GIO pins 3 and 4 on the 6 pin expansion header. Based on the distance between the two piers, placement of the motes on the piers and the surrounding environment the motes can be connected to an internal or an external antenna. When not collecting data, the accelerometer is switched off using the switching circuit.

5.1.2 BriMon Hardware Components

In this subsection we describe various components that have been used in the construction of different modules above.

Sensor Motes

The heart of BriMon is the sensor node. We initially intended to make our own sensor node platform. In an earlier work [41] a sensor node around 89C52 micro controller and BIM-433-40 [8] radio module was made (Figure 5.3). Programing for the module was done using C and assembly language. The success encouraged us to go to the next level and make more complex module around newer and sophisticated micro controllers and radio components. Most of the schematics for it was available online but the problem came in terms of programing the nodes. Developing software for different software components required lots of effort and development time.

Reuse of components in software reduces the development time. TinyOS [65] is an open source operating system specifically designed keeping in mind the constraints of sensor nodes. Through gradual contribution and evolution it has a large component library of both common and not so common software modules. Also once learned it offers fast development and testing. Development of new hardware platform and associated software requires considerable effort even in TinyOS. Thus we moved to the best available cost effective solution available in market and with good support in TinyOS community.

Tmote-sky

Tmote-sky (Figure 2.2) is the latest sensor node platform available from Moteiv Corporation. The design is based on the open source hardware specification for Telos platform [60]. The Telos platform offers many new features not available in the Mica-2 platform. Notable features for Tmote-sky are:

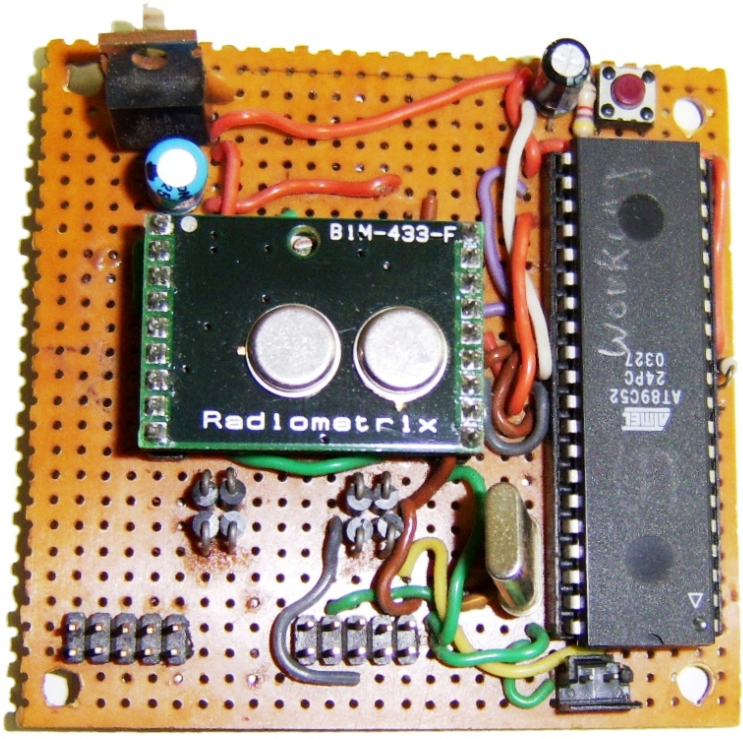


Figure 5.3 Sensor Node based on BIM433 radio and 89C52 micro controller

- Use of newer and more powerful TI's MSP430 [62] 8 MHz 16 bit low power micro controller with 10 KB RAM and 48 KB program flash memory. Using external crystals it has a wake up time of < 6 μs which makes Tmote-sky ideal for applications with small duty cycle but frequent and short wake-up.

- Use of 802.15.4 compliant TI's CC2420 low power radio transreceiver. CC2420 works in the 2.4 GHz ISM band and can work in coexistence with 802.11b/g. It offers radio data transfer rates upto 250Kbs. Observed connectivity distances were $\approx$ 125 m for line of site and 15 m for non line of site using 3 dBi internal antenna. The transmit power of the radio can be configured and is 0 dBm when at maximum. The receiver sensitivity is programmable and can go as low as -94 dBm.

- It has two expansion heads of 10 and 6 pins respectively, which support multiple protocol data connectivity (SPI, UART, USART, I2C), ADC and DAC support

and General I/O pins.

- 1 MB data flash memory for data logging and other applications.
- Onboard temperature, humidity and photo sensors and two internal 12 bit ADC.
- USB connectivity for programing and/or powering of mote.
- Working range of 1.8 - 3.3 V.
- Option for connecting external antenna.

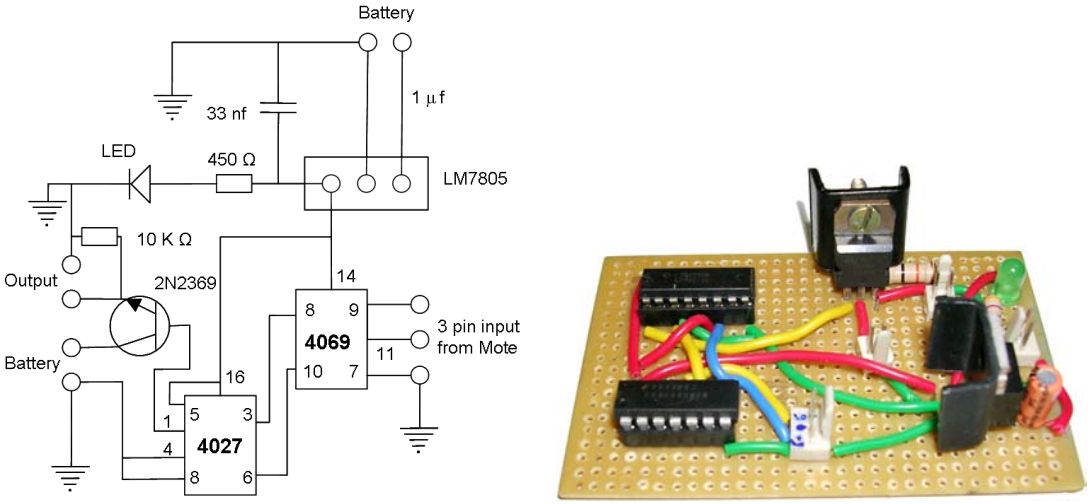
In BriMon the 12 bit ADC is used for converting analog signals coming from the accelerometers to digital values before logging it in memory. CC2420's listening energy in idle state is comparable to transmission energy and does not possess low power listening capabilities. Thus the radio is turned off when not in use. Most of the TinyOS software has already been ported for Tmote.

MEMS Accelerometers

We started with the 5V Silicon Microstructures' SM-7130 capacitive accelerometers which were used by the Structural Lab, IIT Kanpur, for its studies. The accelerometer offered ±2 g range with full scale output span of ±1 V (effectively 500 mV/g resolution). The current consumption is between 6-8 mA. These were superseded by ADXL203 MEMS accelerometers from Analog Devices. ADXL203 offers ±1000 mV/g resolution and noise as low as 110 $\mu g/\sqrt{Hz}$. Also we can configure the accelerometer's working range from 3-5 V which offers greater flexibility in terms of interfacing with Motes. The current consumption of the accelerometer is 700 μA offering order of magnitude less consumption compared to SM-7130. We also use a tri-axial accelerometer MMA7260Q from Freescale. This accelerometer IC offers a configurable range from 1.5-6 g with a sleep mode. The current consumption is 500 μA in normal working and 3 μA in sleep mode. The resolution offered is 800 mV/g but the noise floor is 350 $\mu g/\sqrt{Hz}$ making them less superior in comparison to ADXL203.

Latching Switching Circuit

Since the sensors and Soekris do not have a low power mode and also do not have a digital switch, we needed to control the power supply in order to turn them off when they were not needed. For this a digital power switch was designed using power transistor and logic gates. The schematic of the switch is given in Figure 5.4(a). TIP 31C is a power transistor capable to transfer currents up to 3 A. As mentioned in [45] the Soekris can withdraw upto 0.4 A of currents which cannot be handled by normal



(a) Schematic diagram **(b)** Prototype

Figure 5.4 The latching switching circuit

transistors. An alternative would have been to use relays but they have greater chance of failure due to mechanical parts as well as have higher power consumption (0.2 W) hence we preferred the transistor based switch. This switch latches the state of inputs from the mote, so the mote can sleep during the period when the state of the switch does not need change. A JK flip flop (4027) was used to store states. Additionally, we added a 4069 hex inverter to boost the logic voltage. Without it the circuit was not working as the voltage level for input signal from mote was between 0-2.8 V and 4027 seemed to ignore the change in voltage. After passing through 4069 the voltage was varying between 0-5 V DC hence the logic was working fine. For powering the ICs L7805 voltage regulator circuit was used.

There are two signal input pins from the mote and are connected to the GIO pins on mote side and to the clock and J pin on one of the JK flip flop (after passing through inverter for voltage boost). The connection for the other pins is given in Table 5.1 below.

The state of the switch changes only on a LOW to HIGH transition of the clock irrespective of the change in the state of the J and K pins. The setting shown in Figure 5.4(a) offer two states for the circuit. When J is set to LOW the output is LOW, with every clock transition. When the J pin is set to HIGH then with each subsequent clock transition the switch toggles between the LOW and HIGH states.

Pin Type	State
K	HIGH
S_d	LOW
C_d	LOW

Table 5.1 Connections for JK flip flop in the switching circuit

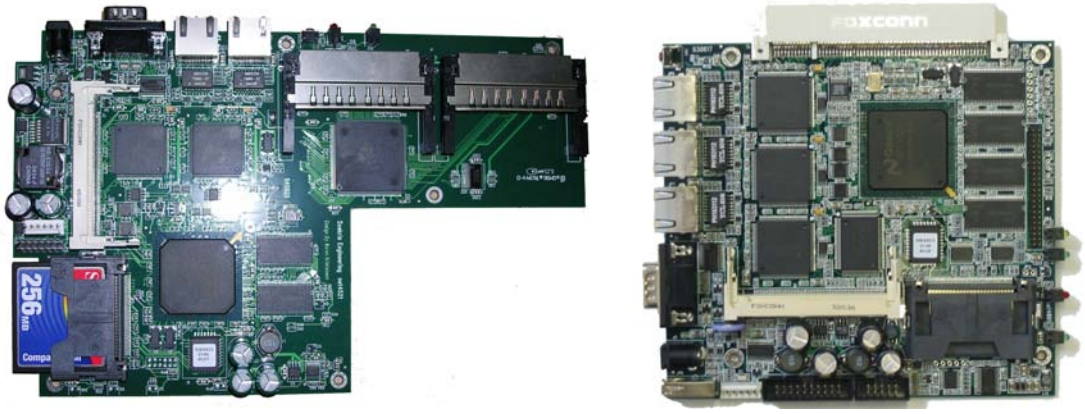
Thus we can always start by keeping the J pin LOW in a known state (output LOW) and reach the desired state by putting J pin at HIGH and toggling the clock signal desired number of times. Figure 5.4(b) shows a circuit prototype for the switch.

Single Board Computer

The data bandwidth offered by sensor motes is not sufficient to transfer the complete data within the small time period for which the train is available within the range of the root node (see Section 5.3). Data transfer between moving 802.15.4 capable radios has not been studied and verified to the extent it has been done for 802.11 radios. Also the amount of data flash available at the root mote is only 1 MB so it cannot save data if we are monitoring a dense deployment. To overcome both of these obstacles we decided to use a single board computer on which a WiFi card can be attached. This computer is able to communicate with the mote in order to collect and store all the data. We used Soekris net4826 and net4501 [67] boards for this purpose. Figure 5.5 shows the net4501 and net 4826 Soekris boxes.

A smallish linux kernel (here Metrix pebble) runs on these boards and they can be programed as any other linux machine. Some salient features of these boards relevant to the application are:

- Mini PCMCI slot (1 or 2) for insertion of WiFi 802.11 cards.
- Internal (net4826) or external (net4501) flash memory support. Typical memory used are 128-512 MB which are sufficient for our purpose.
- Variable power supply support with 11-56 V (for net4826) and 6-20 V (for net4501). There is an optional 5 V supply option for net4501 which can be useful in case higher voltage power supplies are not available (e.g. Solar Cells).
- Hardware watchdog for reset in case of software crash.
- Wide range of operating temperature.
- 64 MB SDRAM.
- 1 DB9 serial port and 1-3 RJ-45 ethernet port.



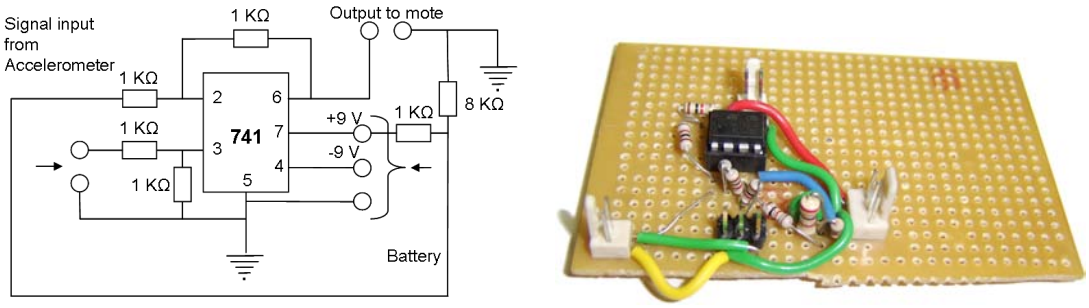
(a) Net 4501 **(b)** Net 4826

Figure 5.5 Soekris single board computers

The boot-up time is an important factor which dictates the application design (Section 5.3). Typically it takes 45 s [45]. Also power consumption values for these boards is in the range of 5-8 W so it is switched off when not needed by the associated mote using the switching circuit described above. We ran a HTTPS server on the Soekris to allow data transfer to take place between the data aggregator and the messaging and transportation module. The mote on the data aggregator module communicates with the Soekris using serial data transfer via the SPI-RS232 translator(Figure 5.7) connected to the DB9 port. The data collected is stored at a specific fixed address accessible via the HTTPS connection, thus enabling transfer to the messaging and transportation module aboard the train.

Antennae

For this application we needed different types of antennae depending on the use. Both directional and omni directional type of antennae find use in it. In our evaluations we have used Tmote-sky's 3 dBi gain internal antenna, Hyperlink's HG2409U 8 dBi gain omni directional antennae and 17 dBi gain 90° sector antenna. A combination of these antennae give different ranges with respect to train detection and 802.15.4 data transfers (see Sections 6.2 and 6.3).



(a) Schematic diagram **(b)** Prototype

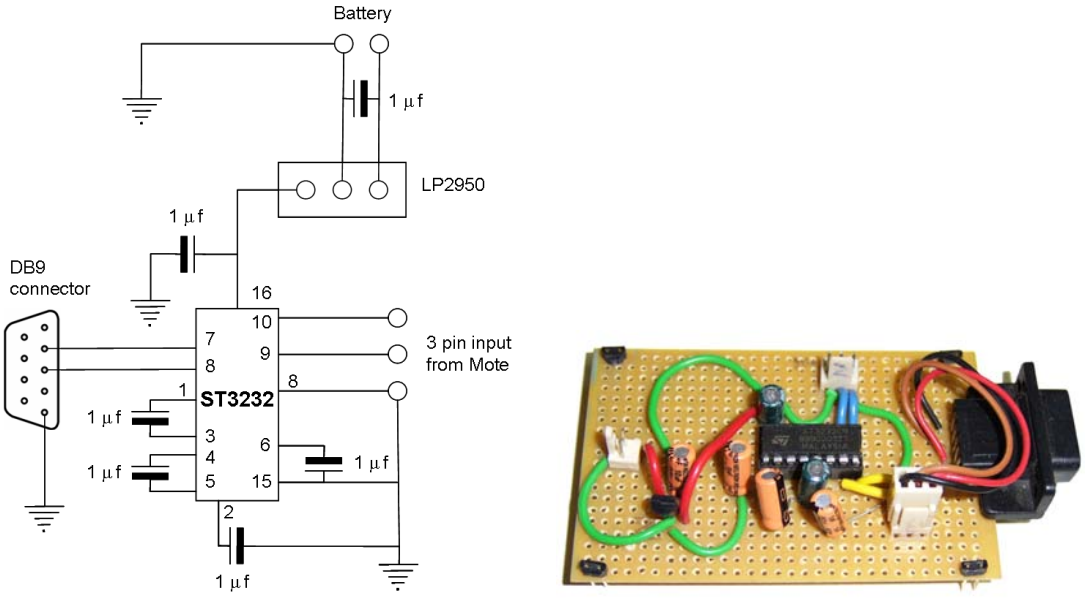
Figure 5.6 The attenuator circuit

Attenuator Circuit

The output of some of the accelerometers is in the 0-5 V range with 0 g base voltage at 2.5 V with resolution of 1000 mv/g. The output of such circuits cannot be fed directly to the 12 bit ADC of Tmote-sky. Hence, we use the attenuator circuit for reducing the dynamic range of values from 1.5-3.5 V to 0.5-2.5 V for 1 g measurements. The attenuator circuit is based around LM741 op amp IC. The schematic and prototype are given in Figure 5.6.

SPI-RS232 protocol translator

The Soekris board has a DB9 serial port which follows the RS232 protocol (working voltage range ±15 V). The motes work on 3.3 V CMOS logic. Thus we need a protocol translator to translate between the two different logic levels. We built a circuit Figure 5.7(a) which performs this translation. The heart of the circuit is the ST3232 IC which acts as the communication interface between the 3.3 V CMOS logic and ±15 V RS232 logic. Since for this chip to operate in the 3.3 V domain we required a constant power supply of 3.3 V, we have used LP2950CZ-3.3 as the voltage regulator. LP2950CZ-3.3 is a low dropout and accurate regulator as compared to L78XX series of regulators. Figure 5.7 gives the schematic and prototype circuit of the translator.



(a) Schematic diagram **(b)** Prototype

Figure 5.7 The SPI-RS232 protocol translator circuit

Battery

In most of our tests we have used two Sunturn AA size LR6 batteries [59] with 1.5 V marked capacity. These batteries deliver approximately 2 AH of power which when used with Tmote-sky nodes at maximum current (23 mA) will last for 3.6 days. Instead if we use 1.5 V LR20 D size batteries [59] offering $\geq$ 11 AH the performance increases 6 times i.e. approximately 20 days. With proper duty cycling scheme we increase the life of the system over months.

5.2 Software

Given below are the details for the two modules responsible for time-synchronization and event detection. For time-synchronization we have used the open source implementation for FTSP. The available code was modified to run on the Tmote-sky platform using the 32KHz clock. Changes are also done to provide improved stability against what is offered by the default implementation. The heart of the event detection method is the ability to detect 802.11 signals. We have used the lower level radio

functionality to generate interrupts on clear channel availability. Details for putting the mote in sleep mode are given in the end.

5.2.1 Flooding Time Synchronization Protocol(FTSP)

The FTSP implementation available as a part of TinyOS is primarily written to be used on the MICA-2 platform. It has the provision to generate timestamps using either the 32 KHz reference clock crystal or the 8 MHz system clock generated by the MCU. We used the 32 KHz reference clock crystal generated timestamps for our application because it allows the processor to be kept in low power modes which is crucial for duty-cycling in our application. This offers time synchronization in multiples of 30.5 μs. The faster 4 MHz system clock was not used, although it can provide even tighter synchronization, as it requires the MCU to be powered-on all the time. The low power mode allows the motes to go in sleep hence saving power when not in use. The current implementation uses a component called `ClockTimeStamping`(Figure 5.8) to insert 4 byte timestamps in the message just prior to transmission from a sender and on arrival at the receiver. These time-stamps are the number of crystal ticks recorded by the MCU from boot-up or roll over of counter. The other delays in the

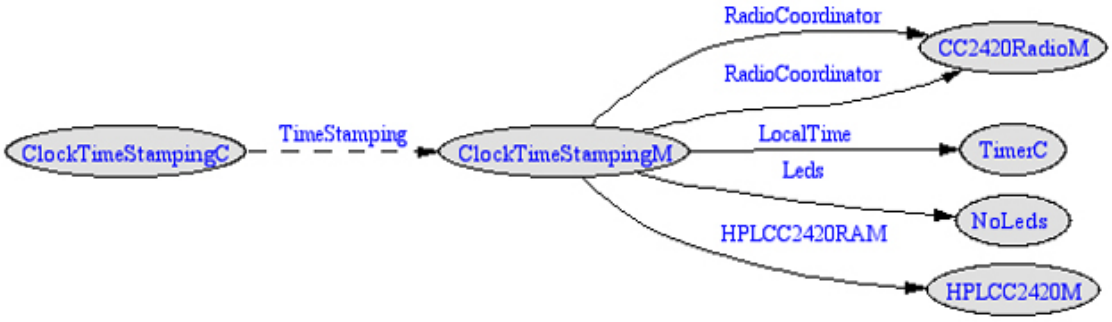


Figure 5.8 Component interaction and interfaces used for ClockTimeStamping in TinyOS

message sequence are either predictable or calculated using averaging. This pair of sender and receiver timestamps are used as a synchronization pair for synchronizing the receiver mote with the sender. Since multiple receivers can lie in the broadcast domain of the sender, we can achieve synchronization at multiple nodes using a single message which is not provided by other protocols (RBS and TPSN).

In the implementation the time-stamps are stored in an array table and used to generate average offsets and skew of the mote's clock with respect to the sender's

clock. The sender's time-stamp in the message is its estimate of the global time, thus knowing the offset and skew values, the receiver can generate its own estimate for global time. For skew calculations the least square regression method is used for which at least four points are required to get good estimates. A node starts sending the estimates when it has at least three synchronization points and declares itself synchronized when it has at least four such points.

In the original implementation this array table is cleared when there is discrepancy in the time stamp i.e. the sender and receiver timestamps differ more than a threshold. We observed in our test runs that at times the sender's time-stamp was having values either very small or very large (negative values in 2's complement representation of binary time-stamp). Recall Equation 4.4. Here we have the global time estimate consisting of three components of local-time, offset and skew. The packet prior to transmission and before using `ClockTimeStamping` component was inserted with the skew and offset values. The `ClockTimeStamping` TinyOS component added the local-time to them. On investigation it turned out that `ClockTimeStamping` missed to put the local-time at times due to fast transmission rate of radio and load on micro controller. Thus we received small or negative values as time-stamps. We have refined

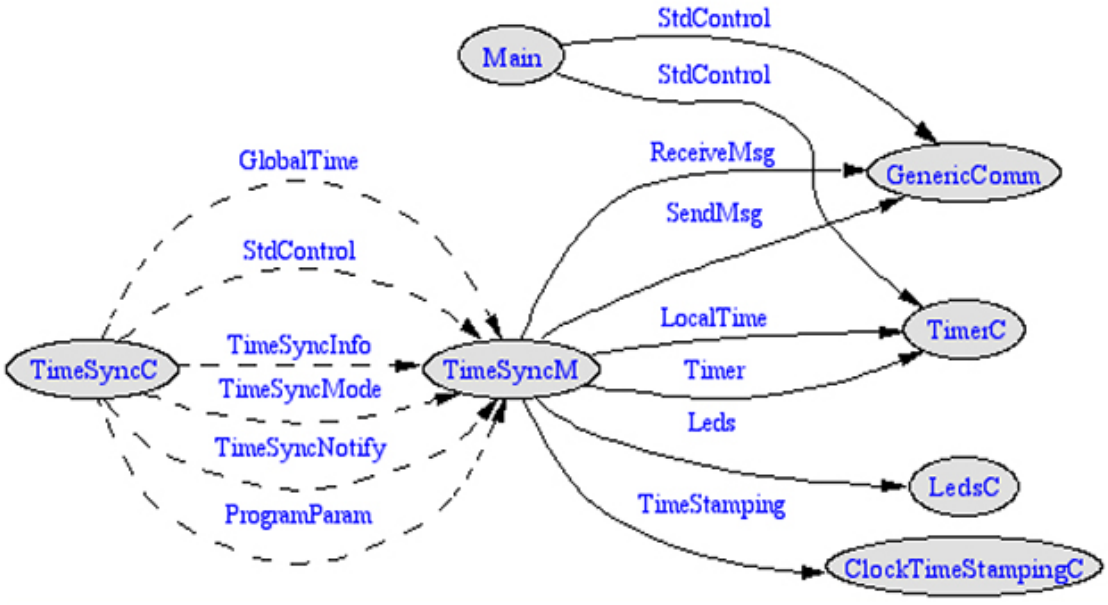


Figure 5.9 Component interaction for FTSP

the original approach and instead maintain a circular array of points. Only packets with timestamps too large or too small get rejected. Thus any small discrepancy gets

averaged and does not affect the current time-stamps broadcasted in the network. The evaluation for both original and modified implementation are provided in Section 6.6. The architecture diagram for the interaction of different components via interfaces used is shown in Figure 5.9. Details of each interface/component can be taken from the SenSys'04's FTSP paper [44].

5.2.2 Event Detection

In our application design we use the train both as a mechanical shaker for data collection and as a data transporter from the remote site to the central data repository. Detection of the approach of the train thus becomes essential. We are not interested in all the trains passing the bridge but only in trains carrying the messaging and transporter module. To detect such a train we use a mote with the ability to detect RF activity in 2.4 GHz generated by 802.11 radio aboard the train. This functionality is generated from the provision of clear channel assessment modes in 802.15.4 specifications. As per the specifications the compliant radios need to provide three modes for clear channel assessment. We use the first mode which gives a channel free interrupt on a specific radio pin (in CC2420 radio on Tmote) when the energy in the channel is below a threshold (called CCA threshold). In our implementation we poll the chip and check whether it is high or low. Based on this we can get whether the channel is free or busy at a given time. Since the radio only checks for energy in the channel we can detect energy generated by sources other than another 802.15.4 radio. In this case we detect energy generated from a 802.11 radio.

The CCA threshold is programmable in steps of 1 dB leading to different sensitivities with which the detection can be done. Similarly, the frequency is programmable in steps of 1 MHz hence we can span over the complete 802.11's 2.4 Ghz range. We have earlier proved that this method is able to detect the 802.11 channel activity correctly in Wake-on-WLAN [45]. Details for the method are in [45]. Figure 5.10 gives the component interaction for the mechanism. We modified the `CC2420Control` interface to accommodate commands to provide capabilities to read and change the CCA threshold via programing.

5.2.3 Duty-cycling in motes

Tmote-sky platform uses the MSP430 series of micro controller. Since the turn-on time for the micro controller is < 6 μs the TinyOS components are written in such a manner so as to put the micro controller in sleep when not needed. All the other components for controlling other hardware modules of the mote, in the TinyOS tree, are written to provide the StdControl or SplitControl interfaces. These interfaces can be used to turn-on and turn-off these components. The StdControl

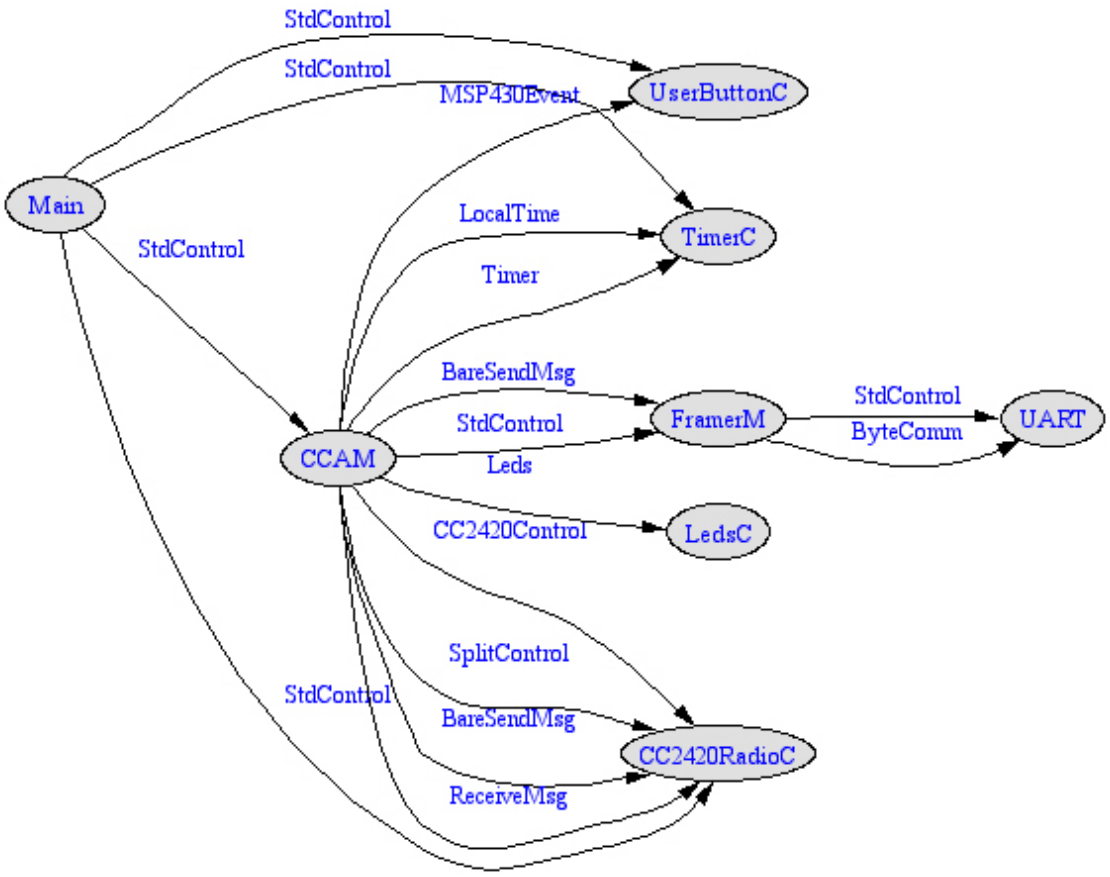


Figure 5.10 Component interaction for event detection mechanism

interface has three commands, `init`, `start` and `stop`. For any component providing this interface, these commands can be called in any of the fashion generated by the sequence `init`(`start`‖`stop`)*. This means once `init` gets called the `start` and `stop` commands can be called any number of times. Thus, to put the node in sleep we need to call `StdControl.stop()` command for each of the hardware component initialized to be used (say flash, radio, UART). To wake up the node we need to call the `StdControl.start()` command on all the hardware components needed. The StdControl interface is being replaced by SplitControl interface in newer components and newer versions of TinyOS will use the latter for controlling the hardware.

5.3 Discussions

As listed in Section 4.1 there are a number of requirements and constraints which needed to be satisfied by our design for being useful. A few points also emerge due to our design choices. In this section, we reason how they get fulfilled by our BriMon application design and the hardware and software modules used.

How 12 bit ADC is sufficient for 0.01 g resolution

The analog to digital converter (ADC) on the MSP430 has 12 bit resolution. Also the reference voltage for the ADC is fixed at 2.5 V thus the range of the permissible input analog voltage is from 0-2.5 V. We use the ADXL203 MEMS accelerometer with input voltage at 3 V. This results in reduction of the resolution of accelerometer to 560 mV/g and the 0g voltage (i.e. at rest) comes down to $V_{cc}/2$ i.e. 1.5 V. Thus for the ±1.7 g range the voltage varies from 1.5±(1.7*0.56) = 0.548 to 2.452 V which is within the range of the ADC. With this resolution a change of 0.01g will result in a change in voltage of 0.56 mV/g*0.01 g = 0.0056 V = 5.6 mV. The ADC's 12 bit resolution employs it divides the 0-2.5 V range in 2^{12} steps i.e. 4096 values. Thus the minimum change in voltage registered by the ADC is 2.5/4096 V = 0.6 mV. Hence the current setup can register a change of 0.01 g as demanded by the application requirements.

Data transfer rates and why 802.11 is necessary

Consider the model with each collector node equipped with a tri axial accelerometer with data sampled at rates not less than 200 Hz and each sample consisting of 2 bytes of data (12 bit ADC data). Also with each sample there is a 4 byte time stamp. Since an average train takes 66 s to cross a bridge of 600 m length [28], we need to capture data for roughly 90 s to capture both the free and forced vibrations. If `n` is the number of axis for the accelerometer, `f` is the sampling rate, `b` are the number of bytes of data collected per sample at each axis, `x` is the number of bytes used for time-stamp and the data is recorded for total of `t` seconds, we have the total amount of data generated per node $Data_{node} = (n * b + x) * f * t$. Using the above values we get $Data_{node}$ `= 180000 B` or 1440 Kb. With the claimed maximum transfer rate of 250 Kbps this will take 5.76 s. Since the observed transfer rate for actual data is much less (≈31.6 Kbps [28]) this data takes roughly 46 s. With inter-pier distance of bridges being in the range of 30-60 m [15] we will have nine such nodes deployed on the bridge. The total amount of data thus which needs to be transferred over to the train would be 9*843.75 Kb = 12960 Kb or 12.96 Mb. Using only 802.15.4 for data transfer, this amount of data will require 410 s of communication contact which may not be sustainable in this scenario. Using 802.11 for data transfer

from the data aggregator node to train this transfer can be done from 0.24 s to 13 s based on possible transfer bandwidth varying from 54 Mbps to 1 Mbps [51]. Thus for a working deployment we need to use an 802.11 based data transfer mechanism to transfer the collected data from the aggregator/base node over to the messaging and transportation module aboard the train.

Long maintenance cycle

Assuming that the battery offers `N` AH of battery power and the mote does `X%` of duty cycling which means that it consumes full power (23 mA) for `X` fraction of time and goes to sleep (i.e. low power state with micro controller at standby and radio turned off) with current consumption of 5-21 μA [63] rest of the time. Also the accelerometer is turned on for `Y%` of duty cycling and consumes 450 μA at 3 V. Let us assume that the Soekris is duty cycled for `Z%` and consumes 5 W power when turned on at 5 V. If the auxiliary circuits consume 1 mA of current when used we have for the data collector nodes the power consumption given by:

$$Power_{Collector} = X*23*10^{-3}*3+(100-X)*21*10^{-6}*3+Y*450*10^{-6}*3+1*10^{-3}*5 \tag{5.1}$$

Similarly power consumed by the data accumulator module will be.

$$Power_{AccumulatorMote} = X*23*10^{-3}*3+(100-X)*21*10^{-6}*3+1*10^{-3}*5 \tag{5.2}$$

$$Power_{Accumulator} = Power_{AccumulatorMote} + Z*5 \tag{5.3}$$

For the mote at the data aggregator the duty cycle depends upon the distance of the frontier node, wake-up time of the Soekris, periodicity of time synchronization and routing protocols. Each time the node wakes up for a fraction of a second and sends appropriate beacons and then goes to sleep. Since reliability is based on transmitting multiple copies of a message, the time for which the node is awake is fixed. For the data collector nodes the duty cycle is decided by the root node and can be assumed to be the same as the duty cycle for the aggregator node. Assuming that data is collected from the site once a week we have for the uptime of the aggregator node as:

$$t_{up} = t_{datacollect} + t_{wakeup}. \tag{5.4}$$

Here, t_{wakeup} is the time for which the node wake-up in between two sleep cycles. If `T` is the total number of seconds in a week then t_{wakeup} = `n * T`, where `n` is the ratio of mote's wake-up duration versus mote's sleep duration. Let `n` be 1/10 i.e. The mote wakes up for one second every ten second and sleeps for nine seconds. $t_{datacollect}$ is the time for which the mote is up during data collection period. Since, Soekris takes 45 s to boot up and it will take 10*46 = 460 s to transfer data over a four hop network.

The data is collected for 90 s. Thus $t_{datacollect}$ = 460+90+45 = 595. Net duty cycle for mote will be t_{up}/T. Using the above values we have X=0.101 or 10.1%. Similar calculations for Soekris gives Z=0.1%. Assuming we have D size LR20 batteries with 11 AH total available power and average voltage as 2.4 V for combination of two batteries we get from Equation 5.1, $Power_{Collector}$ = 0.012 W. Hence the collector node will work for

$$Life_{Collector} = \frac{11 * 2.4}{Power_{Collector}} \tag{5.5}$$

i.e. 2200 Hrs or approximately three months. This is also the power consumed by the mote at the data accumulator module. For the Soekris box using an array of batteries and the fact that the Soekris is used not very often we can extend the life of the complete data accumulator module to match the lifetime of the data collector module. Power will not be an issue for the messaging and transporter module aboard the train. Power consumption of the frontier node will be of the same order as that of collector node. Thus we can assume the maintenance cycle of the deployment to be of 2-3 months. In the above calculation for simplicity of calculations we have assumed the working range from 1.6V to 0.9V for a battery. For Tmote the flash cannot be used for writing if the voltage drops below 2.7V. Hence, to use flash 3 batteries will be needed with appropriate regulators such as LP2950CZ-3.0 for giving power to the mote.

Small form factor for deployed modules

The frontier node module consists of a sensor node, D size batteries and a sector antenna. Although the first two can be fitted in a box of size 15 cm x 15 cm x 5 cm the sector antenna is very much visible object. Also the frontier node will be located upstream of the bridge so it cannot be hidden underneath the bridge. To prevent theft we will need to put this module at a height, preferably on electric or signal pole. The data aggregator and collector modules on the other hand can be easily hidden beneath the bridge with only omni directional antenna visible which in turn can again be put on the electricity supply poles.

Timely detection of train: location of the frontier node module

Referring to Figure 5.11 we have frontier node at A having a specific range in which it can listen for the train with a given high probability. Let it be d_1. Let us assume that the train has a constant speed of v_1. Thus the train will cover d_1 distance in time $t_1 = d_1/v_1$. So if duty cycle for node at A is T_1 then

$$T_1 \leq t_1 \tag{5.6}$$

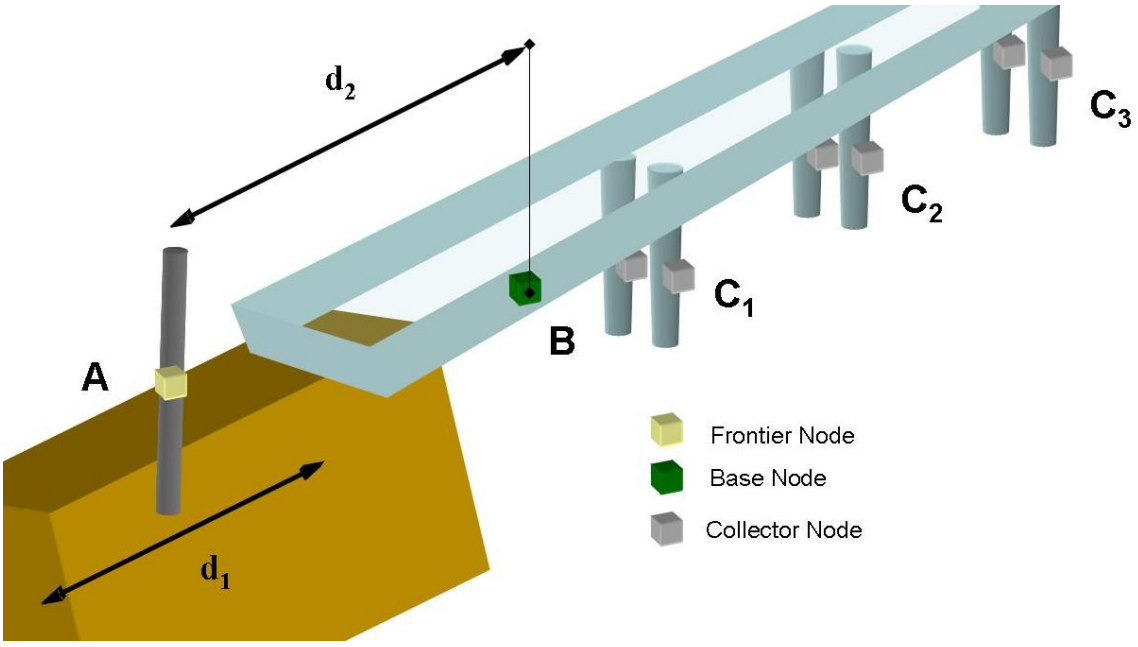


Figure 5.11 Location of different modules for BriMon

In the worst case when $T_1 = t_1$, the train can get detected when it has already covered a distance of $d_1/2$ i.e. it is at a distance of $d_2 - d_1/2$. Thus time taken to cover this duration is

$$t_2 = 1/v_1 * (d_2 - d_1/2) \tag{5.7}$$

Assuming instantaneous communication speeds then Equation 5.7 gives the time between which

- The Soekris must boot up for data transfer
- The data collector module on the bridge piers are up and receive the time at which they will start gathering data.

But the mote at `B` also sleeps with duty cycle T_2. Thus if T_3 is the time taken by the Soekris to boot up then

$$T_3 + T_2 \leq t_2 \tag{5.8}$$

For motes at C_i there are two constraints. Firstly, they need to remain synchronized hence they must exchange beacons for synchronization at a fixed rate T_x governed by the permissible error and the synchronization algorithm employed. Secondly, they should be able to capture root's beacons and transmit it to the complete network.

Let T_4 be the duty cycle for motes at `C`. These motes need to catch the beacon packets of base node, hence they should be awake only when root is on. Thus

$$T_4 = T_2 \tag{5.9}$$

The timer for sleep is reset once a particular job is done, i.e. the root reallocates the wake-up schedule each cycle it gets some work done (routing, synchronization or data collection). The motes are assumed to be synchronized. Also T_4 will get dictated by the routing and synchronization. So taking values for the variables above, $v_1 = 36$ Km/hr, $T_3 = 45$ s, $T_2 = T_4 = 10$ s and $d_1 = 150$ m putting in Equation 5.8 we get $d_2 = 625$ m, i.e. the required range is 625 m. For $v_1 = 72$ Km/hr, $d_2 = 1275$ m. As per our results the first value is within the range of sectoral and omni directional antenna communication for 802.11 radios. Also this is the worst case scenario and we can reduce the distance between the frontier and base node by reducing the boot-up time for Soekris and ensuring detection prior to the train crossing the frontier node. Although in this setting we will need to use two different antennae either at the frontier node or at the base node.

Low cost of the equipment

Current forced balance or piezo electric accelerometers from Kinemetrics cost over $8,000 each [12]. Adding to it the cost of data loggers and cabling we get a system which costs more than $10,000 for a single node. For a bridge deployment of say nine such accelerometers the cost will be approximately $75,000. This does not include labour costs or consultancy fees for the structural engineer or cost of the power supply needed to run the equipments in a remote location.

The cost of individual components needed in our system is given in Table 5.2. For our system the cost for the sensor node is $78 but it can further be reduced by bulk buying or self fabrication (due to open source design). Thus cost of a data collector node with a sensor node, 3 accelerometers, an external omni antenna, auxiliary circuits and 4 batteries will be $78 + 3 * 8 + 50 + 10 + 4 * 0.7 = \$165$. Cost of the frontier node will be $78 + 50 + 4 * 0.7 = \$131$ while for data aggregator node with sensor node, external antenna, auxiliary circuit, 10 batteries, a Soekris box with 802.11 radio and 2 external omni antennas will be $78 + 50 + 10 + 10 * 0.7 + 200 + 60 + 50 = \$455$. The cost of messaging and transporter module can differ based on what is used as the computing platform. Using a laptop computer and a sector antenna will make the module cost at $500 + 160 = \$660$ while using Soekris box, 802.11 card and sector antenna will cost $200 + 60 + 160 = \$420$. Thus for a deployment with 9 accelerometers will cost $9 * 165 + 131 + 455 + 420 = \$2491$. Also the cost of the messaging and transporter module can be distributed between a number of deployments. Thus we are able to achieve cost reduction greater than 96%.

	Cost (in $)	Used in
Sensor Node	78	FN, DA and DC
ADXL103 Accelerometer	8	DC
HyperLink 8 dbi omni directional antenna	50	FN, DA, DC, and M&T
HyperLink 17 dbi 120° directional antenna	160	FN, and M&T
D size alkaline Batteries	0.7	FN, DA, and DC
Soekris net4826 single board computer	190	DA
Mini PCI 802.11 WiFi card	60	DA
Laptop with 802.11	500	M&T
Auxiliary circuits	< 10	DA, and DC

Table 5.2 Cost of components needed for BriMon [**FN** = Frontier Node, **M&T** = Messaging and Transporter, **DC** = Data Collector and **DA** = Data Aggregator] (**Source:** Internet Search)

Reliable data transfer

In the application design a reliable data transfer mechanism is required to allow data integrity. Loss of data can compromise the accuracy of the system and also desired fidelity. Thus we need a reliable data transport protocol to transfer data from the collector nodes to the temporary storage at the aggregator node. In BriMon we deviced a NACK based hop by hop reliable data transport protocol. Here each node is responsible to reliably transfer its own data as well as data for its children, grand children, etc i.e. each node transfers data for the tree rooted at that node. This tree graph in turn gets constructed during the routing phase. The data is substantially large to be able to reside in the RAM alone. We store the data in flash memory at each node before transmission. Thus we follow a store, wait and forward strategy. Each node first gets all the data for the tree rooted at it, stores in the flash memory and then forwards it to its parent. They also use a header and tail message to indicate beginning and completion of data transfer from a node. The details for the protocol are given in Hemanth's work [28]. For transferring data from the aggregator node to the train we use TCP over 802.11 which again is a reliable protocol.

Automation and self healing

In the application we have used dynamic routing to allow it to be able to recover from node failures. The routing protocol is a root initiated process where the root sends a message soliciting child nodes. All nodes which are able to listen to this

message choose themselves as the child of the root and convey this message to the root node. This acknowledgment also acts as the child solicitation message for the next layer of nodes. This message also has a field indicating the distance of the node from the root. Thus the complete network is given a parent-child hierarchy. This hierarchy allows data routes to be setup which is subsequently used in the reliable data transfer method.

In case of node failure the parent child relationship will get broken. To handle this case the root transmits a periodic routing beacon message. If a node does not receives this message for a given time it initiates a route detection mechanism which associates the node with a new parent node. Details of the approach can be read from Hemanth's work [28].

Data transportation from remote site

In our application we use the train itself as the data transporter to collect data from the remote site and deposit it to the data repository. This is a novel feature not used in other similar applications. This method is both simple and low in cost and complexity in comparison to other methods such as satellite communication or long distance radio link.

Chapter 6
Performance Evaluation

In this section we evaluate different parameters and modules associated with the BriMon application design. We first demonstrate a comparative analysis of the MEMS accelerometer with Force Balance Accelerometer (FBA) at different frequencies. FBAs are used in current structural health monitoring systems and are bulky piece of sensitive equipment. Next we quantify the range achieved by a node for detection of WiFi signals using Wake-on-WLAN type mechanism. Range values for motes connected with external antennae and possible data transfer rates, using 802.11 in motion are provided. A set of measurement results are quoted for the achievable data transfer rates for 802.15.4 which justify the use of 802.11 usage for data transfer to a moving train. Finally, we test the FTSP protocol on a linear network and compare the results obtained for the original implementation strategy with a refined method which gives better network stability.

6.1 Accelerometer comparison

Currently, the accelerometer used by the Civil Department of IIT Kanpur for taking structural analysis related data is the single axis Forced Balanced Accelerometer (FBA11) [56] from Kinemetrics which costs upwards of $1000. Similar accelerometers also find use in commercial structural monitoring systems also (Figure 1.3) [10]. In our BriMon application we use Analog Device's ADXL203 dual axis MEMS based low power accelerometers. In order to do so we need to verify whether the MEMS accelerometers are suitable for structural engineering purposes. The aim of this experiment is to establish the fact that the MEMS accelerometers can effectively replace the expensive FBAs. Figure 6.1 shows the setup used in the experiment. The FBA accelerometer is powered by a ± 12 V supply. The MEMS accelerometer is given a supply of 5 V. In the experiment, we mounted the two accelerometers on the same platform i.e. shake table and provided external vibrations to the table in calibrated frequencies. The response of the accelerometers to the vibrations are recorded using LABVIEW DAC (Data Acquisition Card).

The amplitude of the shake table vibration was variable and was set by controlling a voltage regulator. This voltage was reduced for smaller frequencies so as to control the amplitude which may otherwise damage the setup. The signal fed to the shake table was a sinusoidal wave with calibrated frequencies generated using a function generator. Since the MEMS accelerometer was a dual axis accelerometer, we calibrated it to align itself with the axis of vibration by using the output at the second axis. The graphs obtained below have the signal from FBA passed through a

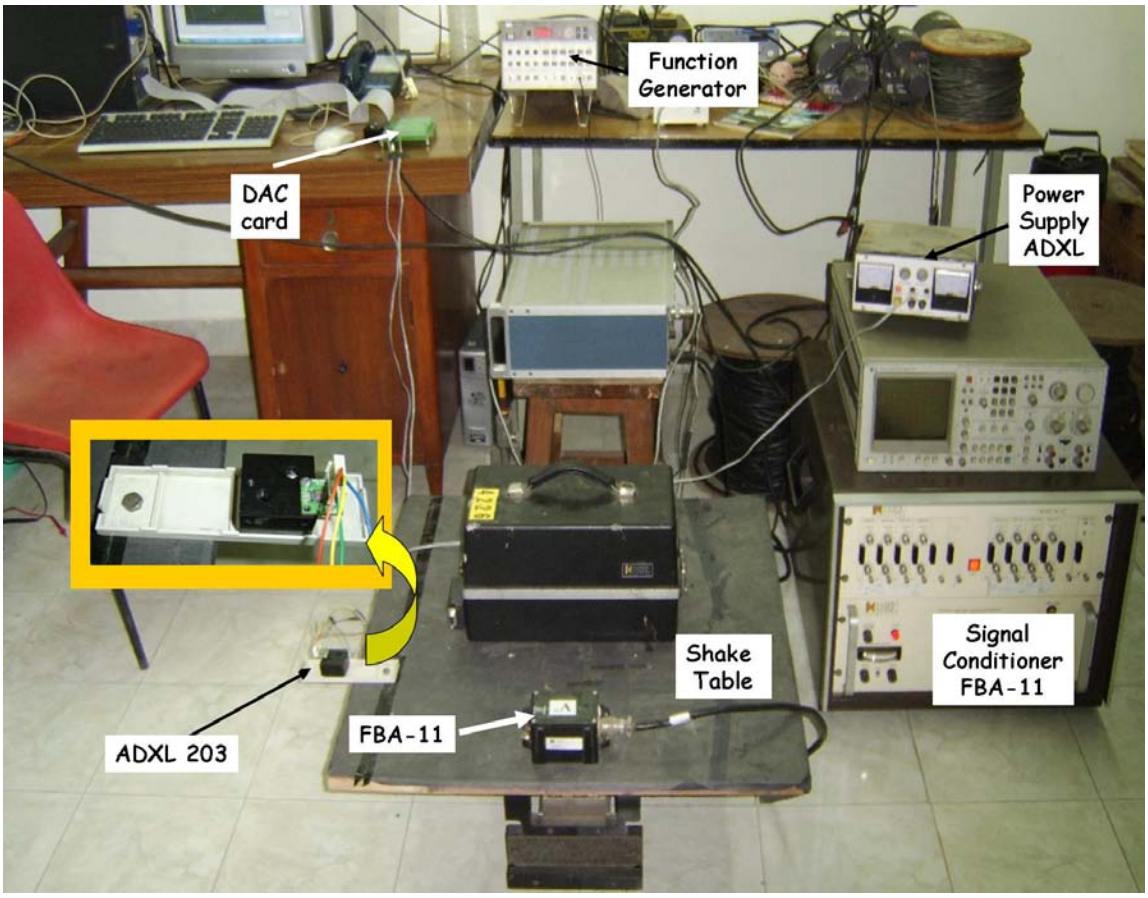
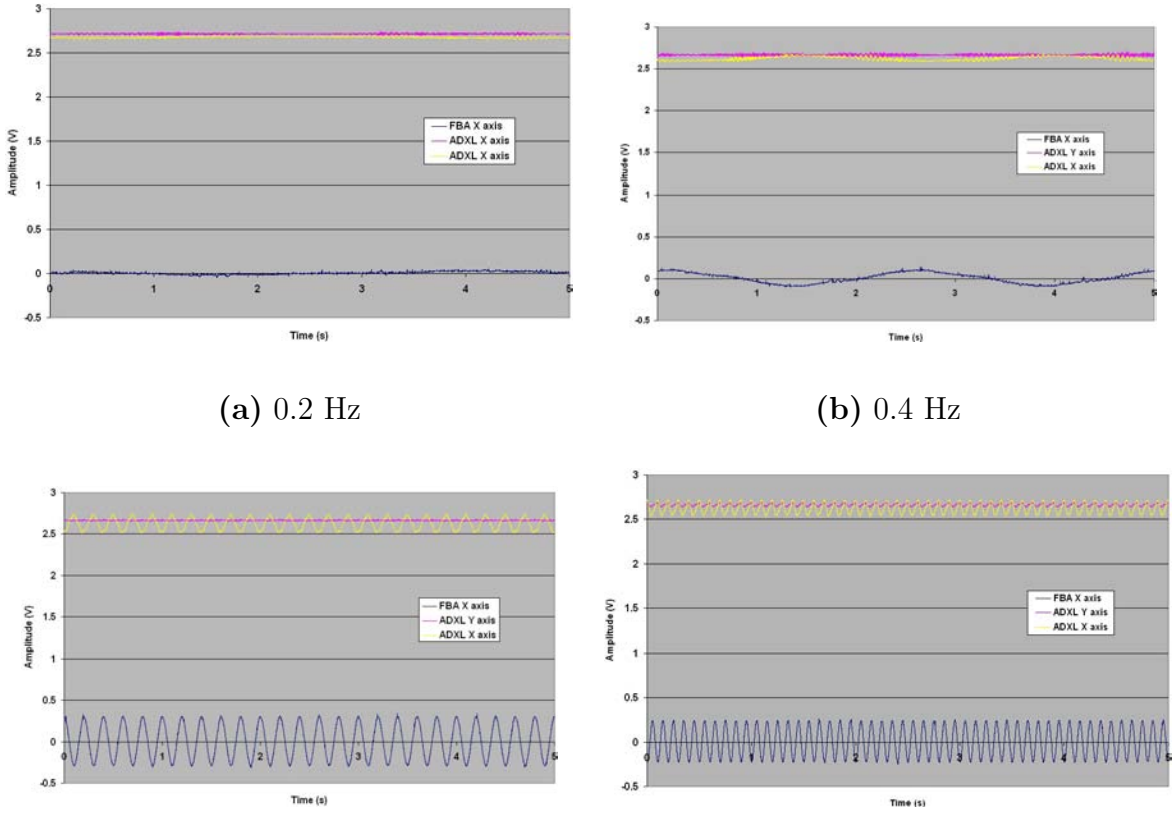


Figure 6.1 Setup for the shake table experiments with FBA and MEMS based accelerometers

signal conditioner before being fed to LABVIEW card. The signal from the MEMS accelerometer on the other hand is directly given to the LABVIEW card. We tested the equipment for the following frequencies: 0.1, 0.2, 0.3, 0.4, 0.5, 1, 1.5, 3, 5, 10, 15 and 20 Hz. At 25 Hz we observed the shake table going into resonance and hence did not proceed beyond this limit. In Figure 6.2 we give the accelerometer readings from both the accelerometers for frequencies of 0.2, 0.4, 5 and 10 Hz. In Figure 6.3 we give the frequency response of the signal after getting power spectral density (PSD) curves. Due to space constraints we do not give all the frequency graphs.

Based on the graphs we can make the following observations.

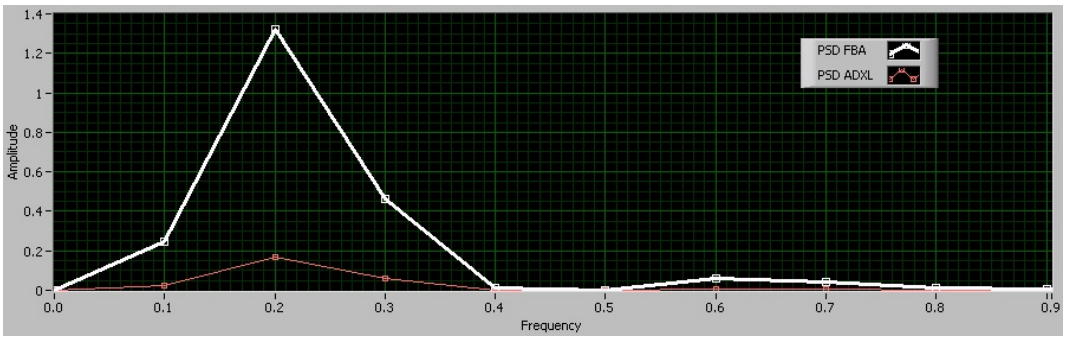
- Both the accelerometers are effective for acceleration readings ≥ 0.2 Hz. Their characteristics and response are the same for frequency ranges > 0.2 Hz.



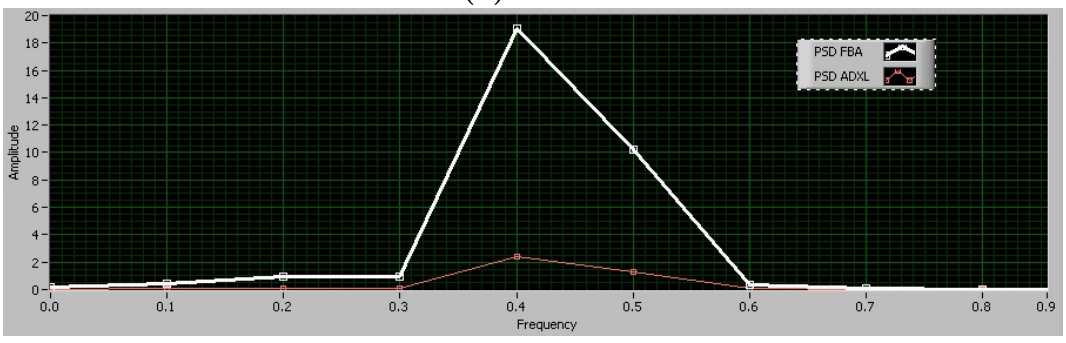
(a) 0.2 Hz **(b)** 0.4 Hz

(c) 5 Hz **(d)** 10 Hz

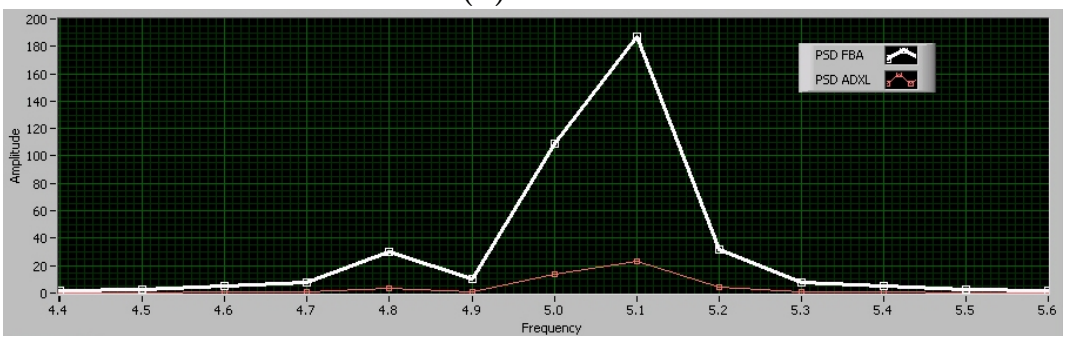
Figure 6.2 Accelerometer's response to a sinusoidal wave



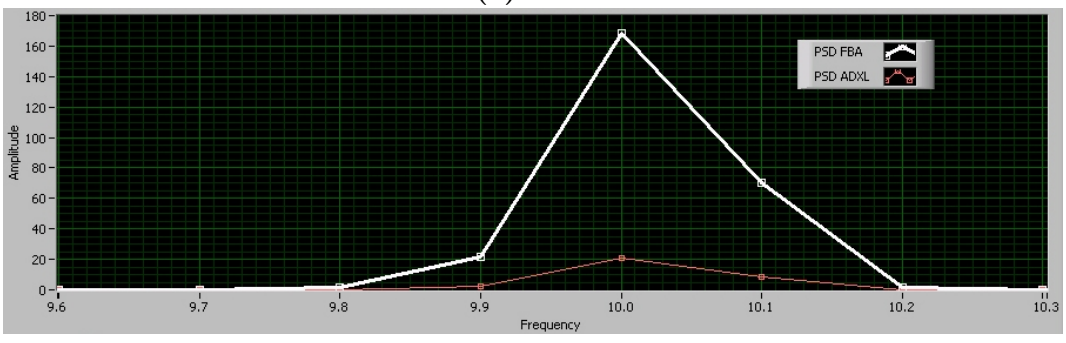
(a) 0.2 Hz



(b) 0.4 Hz



(c) 5 Hz



(d) 10 Hz

Figure 6.3 Power Density curves for the accelerometer's output (refer Figure 6.2)

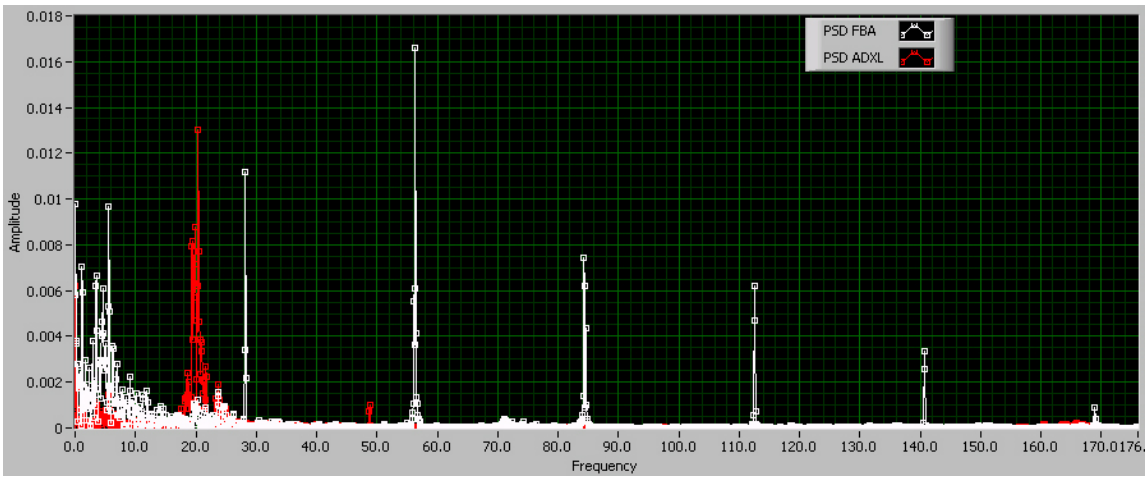


Figure 6.4 Frequency response at 0.1 Hz

- Although the claimed frequency response range for FBA is from 0.01 Hz onwards we observed that both of the accelerometers fail to give proper response at 0.1 Hz (Figure 6.4), in this setup. The dominant frequencies observed in this case are the natural frequencies of the respective accelerometers. The argument given for failure is that the amplitude of shake table was very small to allow the accelerometers to work correctly. A future experiment is planned with the same equipment on a larger shake table, currently being set up at the Structural Engineering Laboratory.

- On application of a low pass filter we could capture the 0.1 Hz response as well. The domain expert [15] argues that such a method will not work on a real structure. As we do not know the correct natural frequency of the structure before hand, one cannot decide for the filter in advance.

- Due to greater resolution (± 2.5 V/g) the PSD graphs observed from FBA accelerometers have more information than the curves from the MEMS accelerometer. Hence we see higher peaks of the FBA plot in the PSD graph for the corresponding frequencies.

- Due to signal conditioning and filtering FBA signal is smoother and less affected by noise. But we were also able to achieve the same results for MEMS graph by designing appropriate filters in LABVIEW (not shown here).

- Adjacent channel affect was observed for higher frequency shaking (> 10 Hz). We found that the software correctly determined the frequency of the shake

table from the PSD curve even for the Y-axis data. The Y-axis was aligned perpendicular to the direction of shaking (not shown here).

Thus the ADXL accelerometer can be used currently in measurements for buildings with structural response greater than 0.2 Hz. We still need to validate results for frequencies less than 0.2 Hz using a newer setup.

6.2 802.11 signal detection using 802.15.4 sensor motes

This experiment aims to quantify the range at which a sensor mote can detect any 802.11 packet when the channel has been flooded by WiFi packets from a certain distance. The experiment was conducted on IIT Kanpur's air strip in order to avoid any radio interference. We scanned the location prior to experiment for any radio activity in the 2.4 GHz range. A stationary laptop, kept at one end of the air strip, is transmitting 802.11 beacons of size 1400 bytes at the rate of 1 Mbps. At this rate the channel is effectively busy for 11.2 ms for each beacon. A beacon is transmitted every 20 ms. We use a mote connected to a laptop to measure the maximum distance at which the signal level (channel energy level) can be detected by the mote. The antenna on the fixed side was mounted on top of a tripod stand or a mast, depending on required height. The ranges observed with different antenna pairs connected at both ends is given in Table 6.1. In the experiments, the mote was moved manually and was held in hand. The experimenter observed an LED on the mote which indicated reception of 802.11 signal. When the LED stopped blinking for sometime and we started observing rate of blinking going down we stopped and marked the location as the range for the setup.

Antenna at laptop end	Antenna at mote end	Distance
8 dBi omni directional	3 dBi internal	240 m
8 dBi omni directional	8 dBi omni directional	380 m
17 dBi 90°	3 dBi internal	540 m
17 dBi 90°	8 dBi omni directional	> 870 m

Table 6.1 Range results of WOW experiment

From the above results we can observe the following:

- The values do not agree with the free space path loss model given by the Equation 6.1

$$P_r = P_t + G_t + G_r - (32.5 + 20log_{10}(f) + 20log_{10}(d)) \quad (6.1)$$

Where P_r is the power at the receiver, P_t is the power at the transmitter, G_t and G_r is the antenna gain at the transmitter and receiver respectively, with f and d as the frequency of operation in MHz and distance between the transmitter and receiver in Km. Thus a different path loss model needs to be used.

- The long range observed with the use of sector antenna can lead to simpler design with merging of event detection capability of frontier node with the base node.
- The method for deciding the range is crude but no other better strategy was available.
- The above experiments were done at the default CCA threshold value of -78 dBm hence the ranges can be increased further by decreasing the CCA threshold which can be programed to value as low as -94 dBm.

We observed during the experiments that the ground plane for the runway was not planer but of a mound shape with a bulge in the middle. When observed from the other end we found that the ground was coming in between the two antennae. The above range values suggest the use of a sector antenna over the train for the messaging and transporter module. The range is sufficient to work at train speeds of 36 Kmph. In future we would like to redo this experiment with the following changes for better results.

- Increase the height of the mast in order to reduce the effect of ground reflection.
- Mounting the moving antenna on a proper platform or a mast so that the plane of the antenna is aligned with the fixed antenna's plane. Also we want to remove the human experimenter from affecting the radio wave prorogation.
- Conduct the experiment in motion to simulate the actual condition.

6.3 Mote's range with external antenna

Table 6.2 gives the observed ranges with different external antennae connected to the mote. In our own experiments with internal antennae and omni directional antennae, we observe that the ranges observed are sufficient for deployment over a bridge with line of sight settings. Although the actual results can be seen only after a deployment, these values strengthen the belief that such a deployment can be achieved with motes separated by hundreds of meters. Details for the experimental settings can be taken from [9]

Antenna mounted at end-1	Antenna mounted at end-2	Range(in m)
3 dBi internal	3 dBi internal	75
8 dBi omni directional	3 dBi internal	75
17 dBi 90° sector	3 dBi internal	210
24 dBi 8° grid	3 dBi internal	500
8 dBi omni directional	8 dBi omni directional	90
17 dBi 90° sector	8 dBi omni directional	500
24 dBi 8° grid	8 dBi omni directional	800

Table 6.2 Range results of mote connected with different antennae (**Source:** [9])

6.4 802.11 data transfer in motion

We will be needing data transfer over to a moving train via 802.11. Although motion should not have any affect on the data transfer rates, this needs to be verified. In [51] Gass et. al. show that the effect of speed between the sender and receiver radios is primarily due to the lesser time of contact rather than data transfer rates. They show that only due to less contact time (i.e. receiver radio residing for less time in the effective radio range of transmitter), one is able to transfer less data rather than due to lowering of transfer rates. They also conclude that the majority delay comes due to the way programs/protocols perform authentication, handshake and data transfer, rather than bulk data transfer. The effect of such application level and protocol related delays have been documented in [51]. The bulk data transfer (using TCP) achieved by [51] are given in Table 6.3. From these values we can conclude that the current model and amount of data transfer needed (Section 5.3) will work without any problem for train speeds as high as 120 Kmph. Use of higher rate 802.11g protocol will definitely increase the quoted values. Also with availability of more powerful radios (up to 400 mW) we can extend the effective radio range and hence the amount of data transfer possible.

6.5 Data transfer rates for Tmote-sky sensor node

Tmote-sky uses CC2420 radio for communication. The radio works in 2.4 GHz ISM frequency range and implements the 802.15.4 specifications. The claimed maximum data transfer rate is 250 Kbps. In our experiments we found that the actual data transfer rates were much less than the claimed maximum rate. To verify we tried the following experiment. In TinyOS to send a data packet you need to use the `SendMsg` command. On successful sending, the calling component gets an event

Speed (mph)	Effective transfer time (s)	Total data transferred (MB)	Effective data transfer speed (Mbps)
5	219	92.3	3.4
15	89	44.2	4
25	58	26.9	3.7
35	37	14.5	3.1
55	19	7.9	3.3
75	12	6.3	4.1

Table 6.3 Effective data transfered for bulk data at different speeds (**Source:** [51])

called `sendDone`. The fastest rate possible will be when the next message is sent as soon as `sendDone` is received. Using this approach we find that the maximum rate of sending is 155 packets of 44 bytes each per second. The packet contains 34 bytes of data payload. This translates to 155*44*8 = 54.56 Kbps of data rate and 155*34*8 = 42.16 Kbps of effective data rate. On the receiver side though we could only receive 102 packets which gives 102*44*8 = 36 Kbps or 102*34*8 = 27.74 Kbps effective data rate. Similar results have been quoted in [28] and [61]. In [28] the values obtained (31 Kbps) are for a reliable data transfer method. The reason for this low rate of reception has been attributed to the implementation of receive buffers in TinyOS for Telos platform [61]. The implementation is such that it gives buffer overflows leading to lower receive rates. Lowering down the data rate by using a timer based sending on sender side improves the receive rate but the effective maximum rate observed is again only 40.46 Kbps [61]. Thus for all our calculations we use these values rather than the maximum claimed transfer rates.

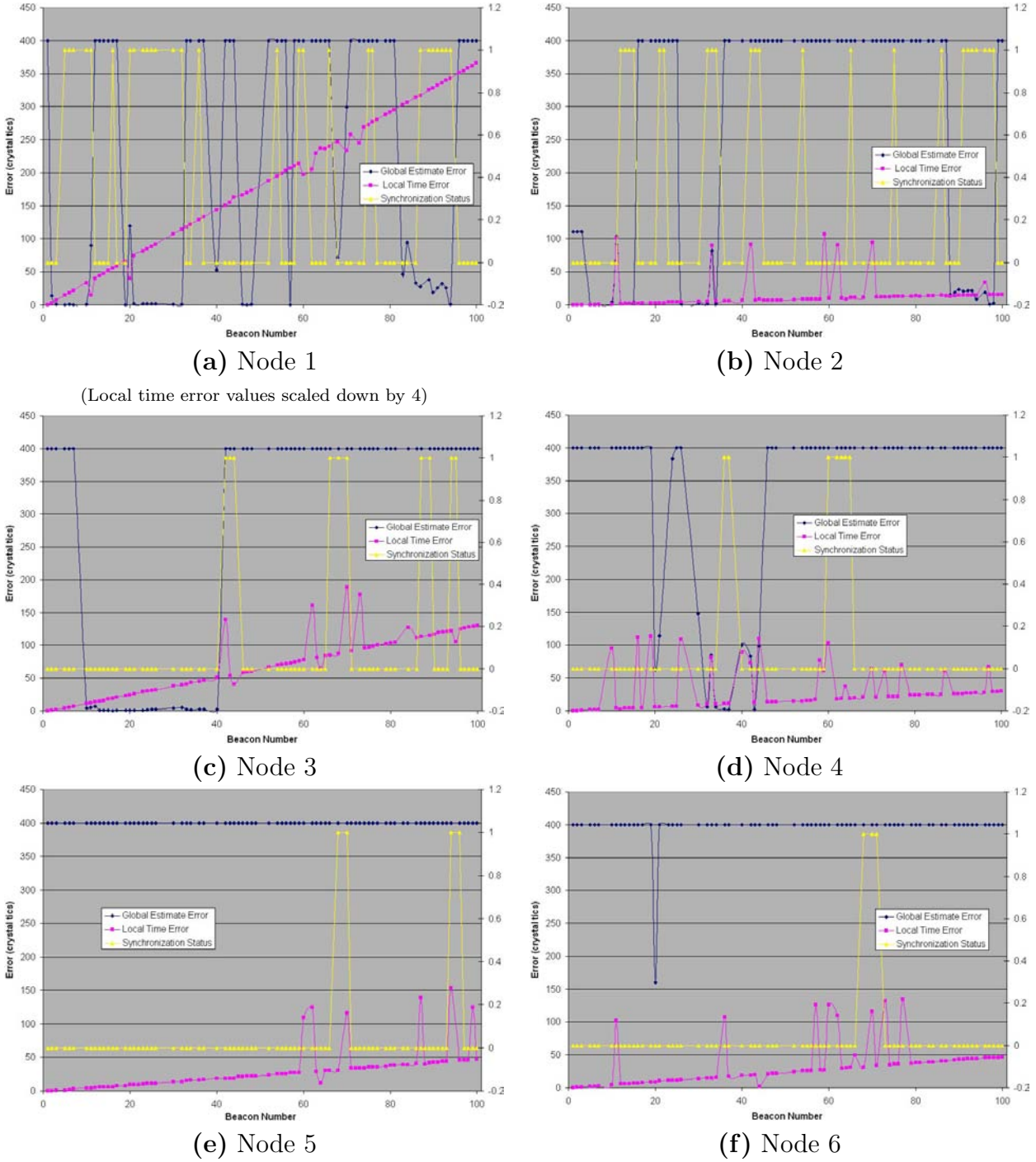
6.6 FTSP related experiments

For evaluation of the FTSP we forced a linear topology on the network. Nodes were given static addresses in a sequence and a node could only communicate with another node having address one less than its own address (called parent of the node). Thus a mote could receive synchronization packets only from its parent node. The root node was the node with the lowest address. A separate node acted as experiment coordinator sending beacon messages (called poller message) and configuring the participant nodes for each round. A second node was programmed with the `TOSBase` application and was used to listen to packets being exchanged. These packets were recorded in a data log and were used during the analysis.

The nodes in the network exchanged time-sync packets to perform synchronization. The synchronization beacon interval is kept as a programmable feature which can be changed from the coordinator node. The poller message beacon was transmitted at regular intervals. Since we captured 100 data points for each round, for our post experiment analysis, we used a variable poller message rate, equal to the rate for FTSP time-sync packet exchange rate. On reception of the poller message the nodes record the global time estimate, local time, clock skew, number of synchronization points in the table and sequence number for the time-sync message last received, against the poller message's sequence number. Between transmission of two poller messages we retrieve the recorded data from each participant node reliably at the coordinator node and pass to the computer using the `TOSBase` application. In this method we are able to capture the timing information almost simultaneously at all the participant nodes. This method does not have sender side delay and mostly calculable receiver side delays.

In our experiment run we had a network of 7 nodes with six hops. The nodes were addressed from 0-6, with node number 0 as the root. The coordinator node with address 12 was connected to a laptop. A second mote was listening to all the network traffic and sent the data to a computer using `TOSBase` application for data logging. We ran the experiment for time-synchronization beacon interval of 1, 2, 3, 5, 10, 30 and 50 s. The experiment was run for both the original implementation and the refined version (Section 5.2). Figures 6.5, 6.7 and 6.9, give the graph for the original implementation's error for global time estimate and local time difference, and whether the node is synchronized or not with respect to the root node, at the respective nodes. Node number 1-6 are nodes at a hop distance of 1-6 respectively from the root node. Figures 6.6, 6.8 and 6.10, give the same graphs for our refined approach. A node is synchronized when it has sufficient (three) synchronization points from the root node. In Table 6.4, the first time when a node gains stable synchronization with root (global time estimate error less than 2 ms for five consecutive data points) is given. This becomes important in case we use a dynamic synchronization approach of using a fast beacon rate to achieve synchronization and later switching to a longer beaconing period.

From Figures 6.5, 6.6, 6.7, 6.8, 6.9, 6.10 we conclude that the refined method gives more stable and consistent network wide synchronization. Now as per the definition of synchronization any node should be considered synchronized when it has sufficient (three) synchronization points from designated root node. The *'Synchronization status'*, in figures above indicate whether this condition is satisfied or not. A value of one indicates that a node is synchronized while a zero indicates that a node is not synchronized. As can be seen from the figures above our refined method offers more stable network with nodes spending more time in the synchronized state as compared to the original implementation. Also from Table 6.4 we can conclude that more nodes



(a) Node 1

(b) Node 2

(Local time error values scaled down by 4)

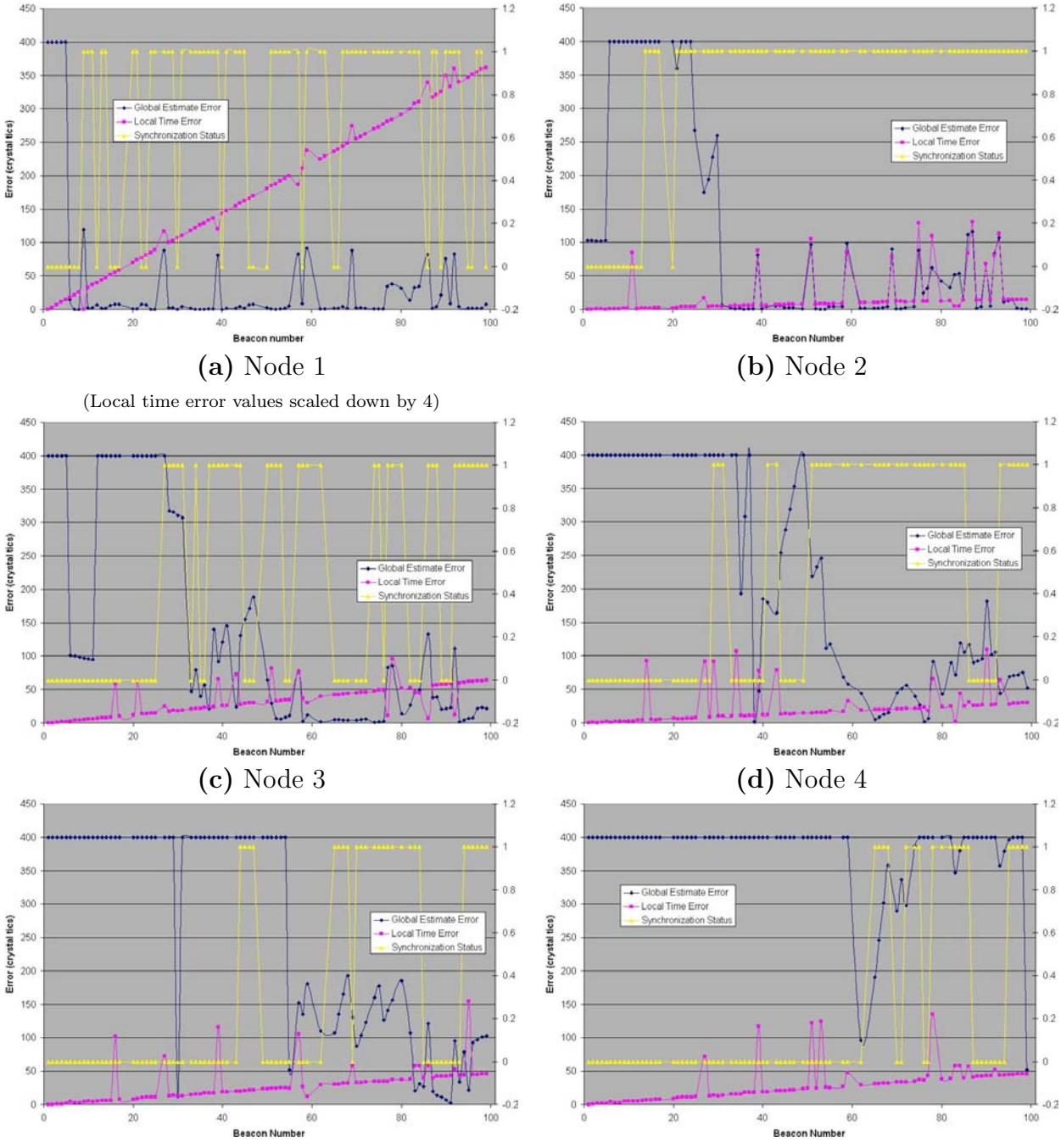
(c) Node 3

(d) Node 4

(e) Node 5

(f) Node 6

Figure 6.5 FTSP error graphs for original implementation with beacon period of 2 s (1 crystal tic = 30.5 μs; global estimate error values greater than $\geq$400 were clipped at 400)



(a) Node 1

(b) Node 2

(Local time error values scaled down by 4)

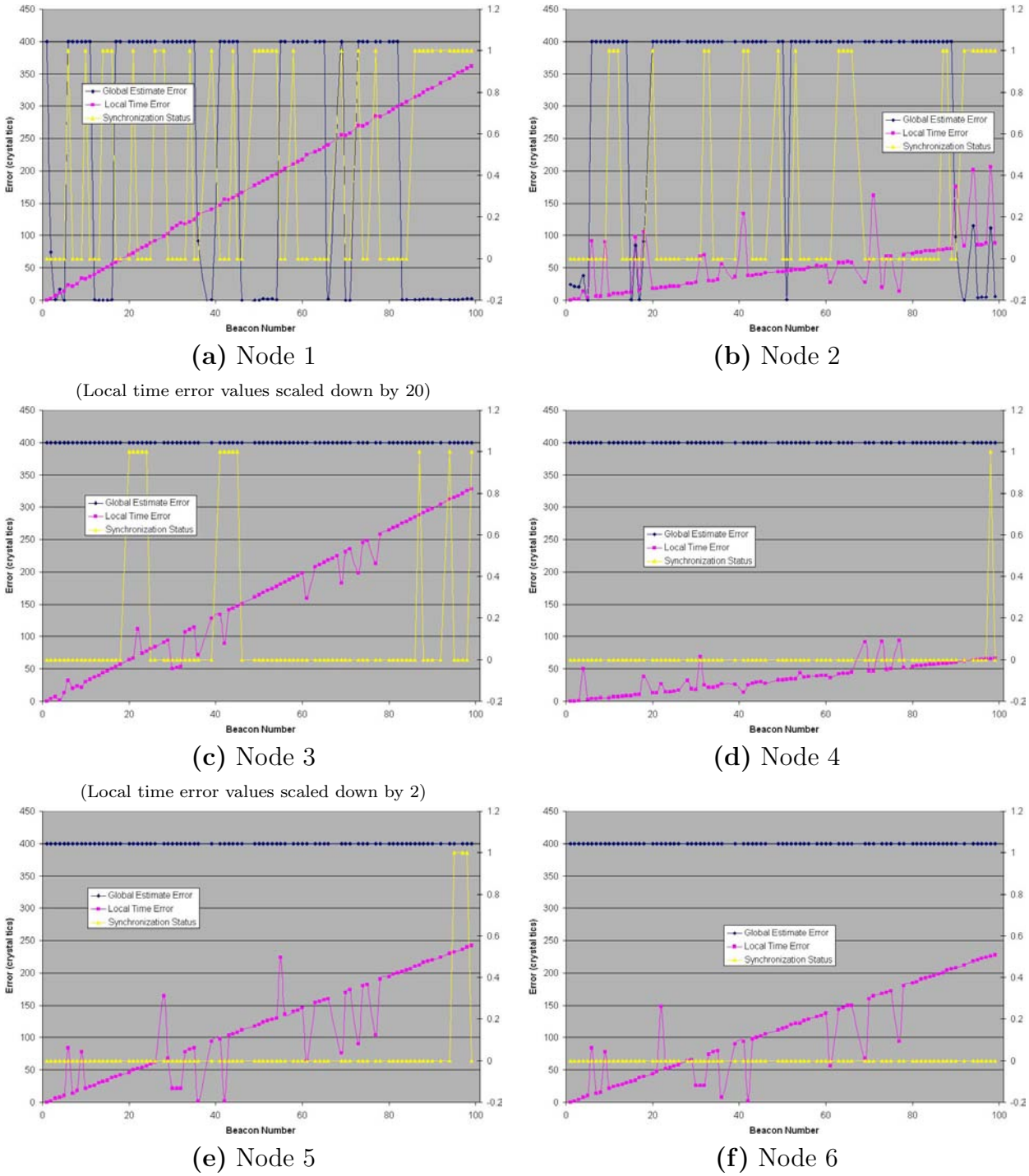
(c) Node 3

(d) Node 4

(e) Node 5

(f) Node 6

Figure 6.6 FTSP error graphs for modified implementation with beacon period of 2 s (1 crystal tic = 30.5 μs; global estimate error values greater than $\geq$400 were clipped at 400)



(a) Node 1

(b) Node 2

(Local time error values scaled down by 20)

(c) Node 3

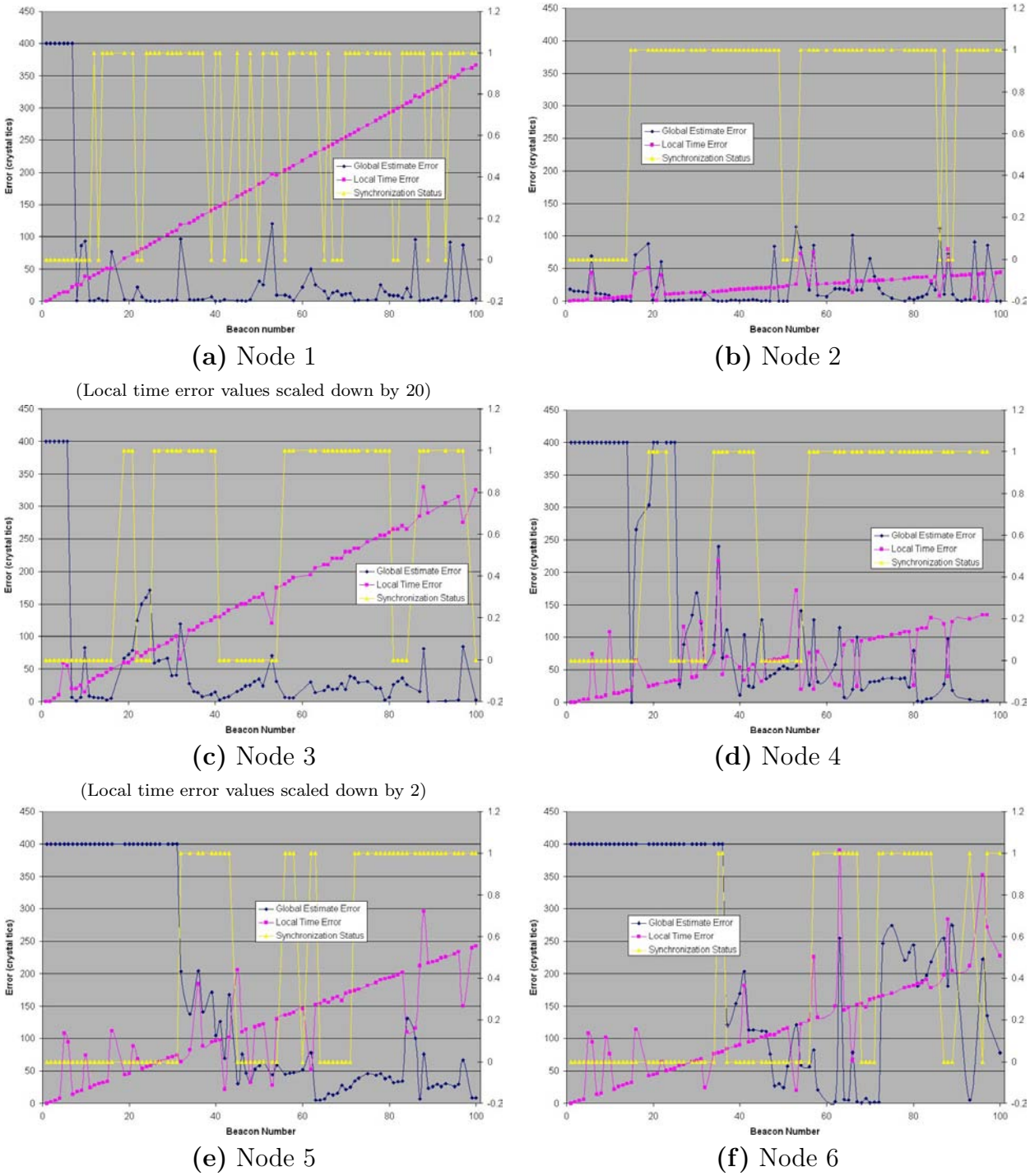
(d) Node 4

(Local time error values scaled down by 2)

(e) Node 5

(f) Node 6

Figure 6.7 FTSP error graphs for original implementation with beacon period of 10 s (1 crystal tic = 30.5 μs; global estimate error values greater than $\geq$400 were clipped at 400)



(a) Node 1

(b) Node 2

(Local time error values scaled down by 20)

(c) Node 3

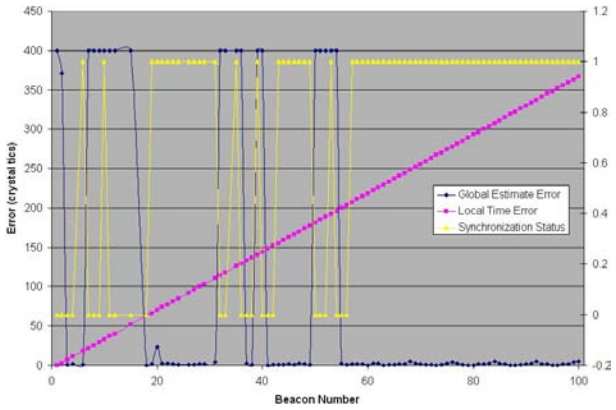
(d) Node 4

(Local time error values scaled down by 2)

(e) Node 5

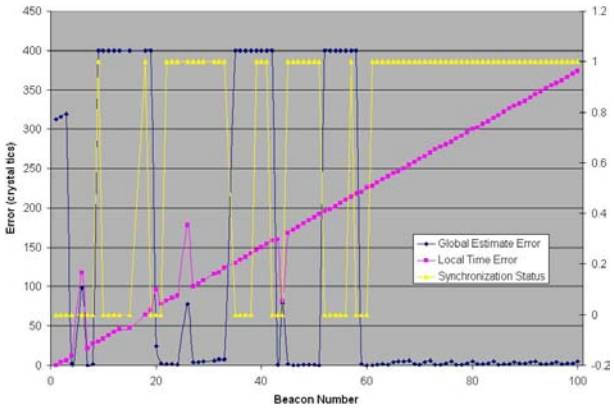
(f) Node 6

Figure 6.8 FTSP error graphs for modified implementation with beacon period of 10 s (1 crystal tic = 30.5 μs; global estimate error values greater than $\geq$400 were clipped at 400)

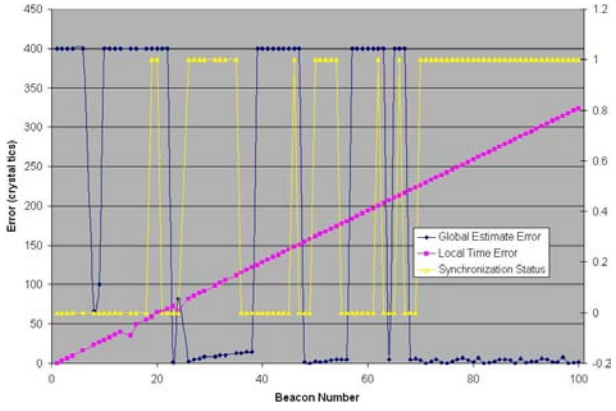


(a) Node 1

(Local time error values scaled down by 100)

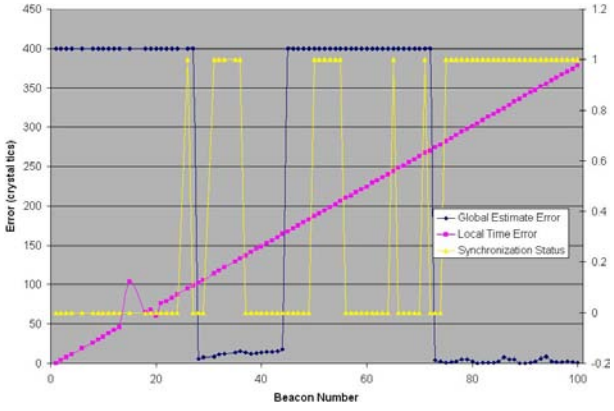


(b) Node 2



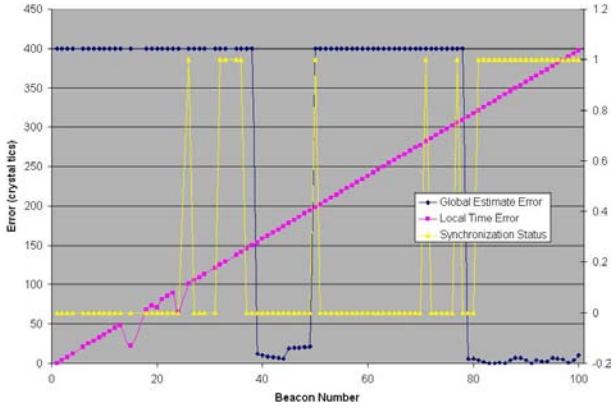
(c) Node 3

(Local time error values scaled down by 10)



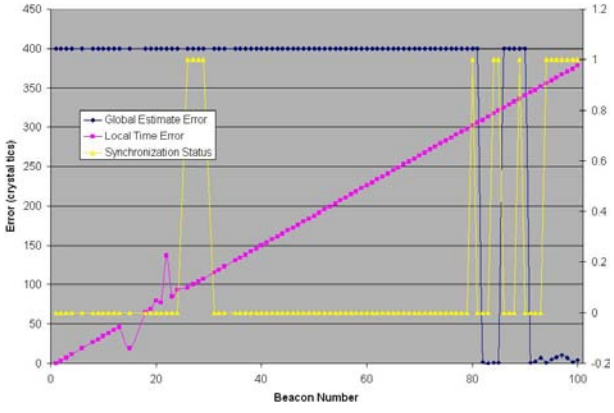
(d) Node 4

(Local time error values scaled down by 2)



(e) Node 5

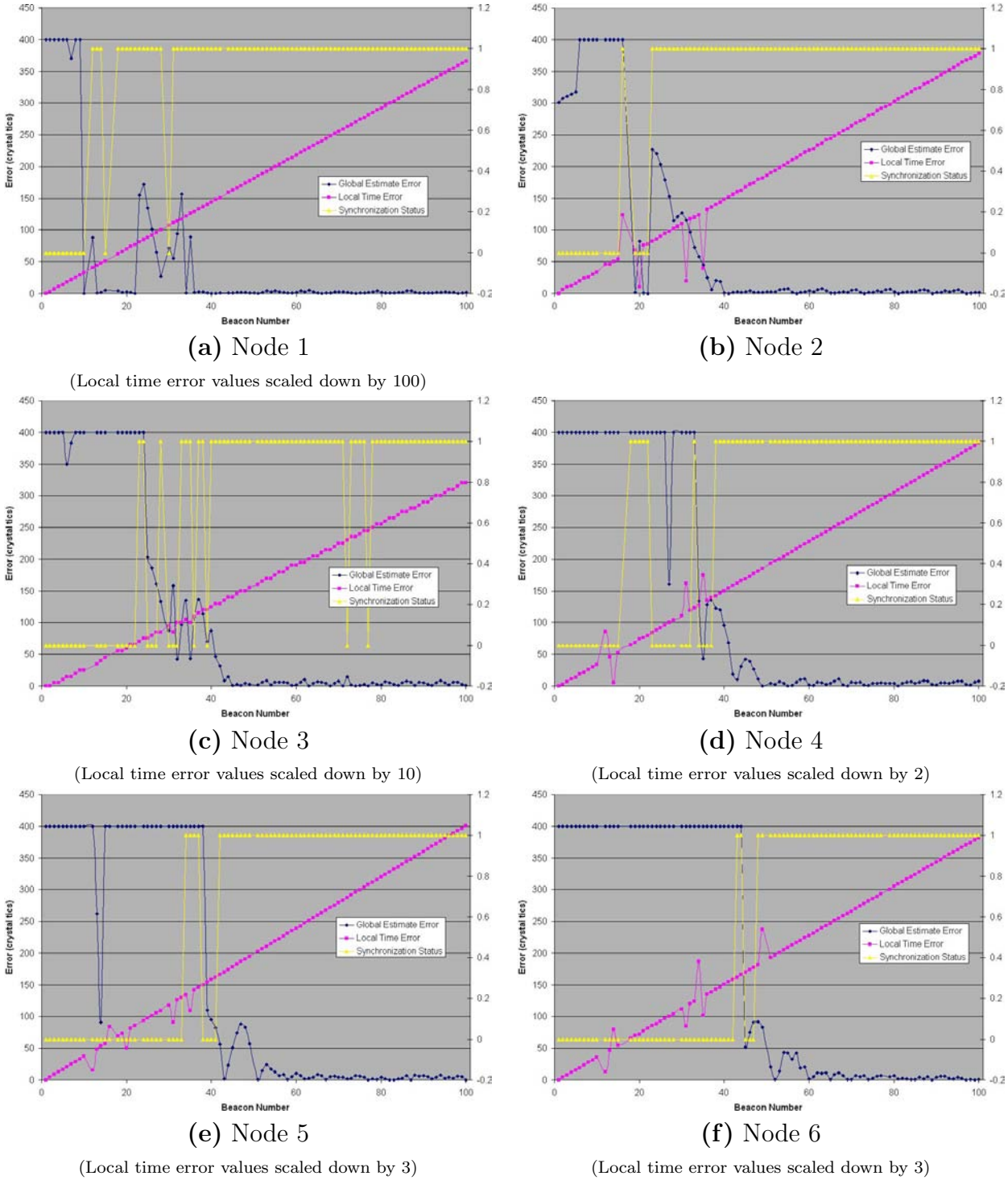
(Local time error values scaled down by 3)



(f) Node 6

(Local time error values scaled down by 3)

Figure 6.9 FTSP error graphs for original implementation with beacon period of 50 s (1 crystal tic = 30.5 μs; global estimate error values greater than $\geq$400 were clipped at 400)



(a) Node 1

(b) Node 2

(Local time error values scaled down by 100)

(c) Node 3

(d) Node 4

(Local time error values scaled down by 10)

(Local time error values scaled down by 2)

(e) Node 5

(f) Node 6

(Local time error values scaled down by 3)

(Local time error values scaled down by 3)

Figure 6.10 FTSP error graphs for modified implementation with beacon period of 50s (1 crystal tic = 30.5 μs; global estimate error values greater than $\geq$400 were clipped at 400)

Beacon Rate (s)	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
2 (original)	5	91	-	-	-	-
2 (modified)	10	31	58	59	-	-
10 (original)	49	> 95	-	-	-	-
10 (modified)	14	20	34	68	72	-
50 (original)	19	27	26	31	81	94
50 (modified)	18	34	41	42	49	51

Table 6.4 Least beacon id for stable synchronization
(units is number of crystal tics, 1 crystal tic = 30.5 μs; **org** = Original implementation, **mod** = Modified implementation)

achieve synchronization in the 100 beacon periods for our approach than the original implementation.

Also to note from the above figures is the fact that network wide synchronization is achieved at an earlier beacon id for larger beacon periods. Also the fluctuations in state decrease with increasing beacon period. One reason for this phenomenon could be the fact that the nodes use there respective internal clocks for firing timers. These timer values are not compensated for skew in the current work. As the beacon period increases, the timer fires at different nodes happen in a more scattered fashion. Now for FTSP to work correctly it needs to time-stamp the received packet as soon as it receives it. With scattered packets coming there are less interrupts and cpu processing thus packets are marked accurately. For smaller beacon rates, the packets arrive very close to each other thus resulting in a number of interrupts and hence inaccurate stamping at times which throws the regression in larger error regions. Also as we noted above (Section 6.5), receive rates are lesser than sending rates and hence closely transmitted packets can be dropped, again leading to increase in error rates.

The fact supporting this is the number of packets received by the logger (node listening for packet traffic and passing to computer). Table 6.5 give the number of packets received for different nodes at different rates. We see that number of packets received decreases as we decrease the beacon rate. This shows that packet collisions

Beacon rate	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
2 s	73	78	73	72	78	73
10 s	82	83	86	82	82	83
50 s	92	93	93	92	93	92

Table 6.5 Number of packets received at data logger for different nodes at different beacon periods

or packet dropping is happening at nodes. Thus for better results we should choose optimal rate which will be between 10-50 s.

Chapter 7
Conclusions and Future Work

7.1 Conclusion

In this work we present a mechanism for a Bridge Monitoring system. The proposed system has a number of advantages over the currently used wired systems. Our system is easier to deploy, lower in cost and autonomous in its operation. The system is designed such that it can be left on site and the data can be retrieved on demand. We propose a new approach to fetch the data from the remote site using the train itself which has not been used earlier in other sensor network projects. We show by evaluation that the use of MEMS sensors does not affect the data quality. We also use a new approach of Wake-on-WLAN to detect 802.11 signals with 802.15.4 compliant radios. This method is used to detect the approaching train which gets used as a vibration generation source (similar to mechanical shaker) for the current run and as a data transporter for data collected at an earlier round.

This work was part of the BriMon project. In separate work [28], routing and reliable data transport modules were designed. After integration of the time synchronization and event detection modules with the routing and transportation modules we intend to deploy the system on a bridge for further study. Such a deployment might bring issues in light which are not visible right now. As stressed earlier the proposed architecture is highly cost effective and saves cost in the tune of 96% when compared with existing systems. The architecture designed has high degree of automation making it very scalable. The on-demand data acquastion system will be a great boon to Indian Railways as they are having one of the largest rail network, in which 74% bridges older than 60 years. Our system gains even more importance in the light of the fact that railways is becoming more aware of this problem and work is underway to develop a comprehensive bridge monitoring systems [31].

Although our work focuses on monitoring of railway bridges, the design can be used in other places as well. For road bridges our design needs minimal changes. Also we can put the accelerometers (or other sensors also) in a network over a structure (a historic building or a damaged structure) and a data recovery vehicle can take the data from these remote sites while moving on the road. Environmental, agricultural or other related monitoring applications can also be made from our design. This is possible by our innovative use of Wake-on-WLAN feature. It offers remote wake-up of a larger bandwidth radio to allow substantial sensor data collected, to be transferred in small contact time. Overall we see this work utilized in future deployments and applications.

7.2 Future Work

Our work would have been complete with a deployment of a prototype. We plan to do this in the near future after integration of the different components. Deployment will also bring to front, issues such as rare events in the automate of the design which may not have been handled or overlooked. Thus rigorous testing before live use is needed. Also there are issues which need further investigation. Some of them are,

- Study of the response of ADXL MEMS accelerometers for frequencies less than 0.1 Hz.
- Use of more sensitive and better resolution accelerometers such as Silicon Design's 1221L MEMS accelerometer.
- Current design with time-stamping each data point has huge overhead (40%). Group based time-stamping techniques, with group size dependent on the permissible error and error rate of time-synchronization algorithm can be studied.
- Boot-up time of the Soekris is a bottle-neck parameter. Reducing the booting time can result in increased efficiency of the system.
- Data compression and edge computing can be introduced to save on transfer time and data needed to be transfered, thus increasing power savings.
- The current 12 bit ADC is not sufficient for certain class of structural data analysis methods. Separate data acquisition boards can be designed which offer more ADC resolution as well as filter the data, reducing the relevant data requiring transmission.

References

1. 8052 core based microcontrollers.
   http://www.atmel.com/atmel/acrobat/doc0313.pdf

2. A. Lucas Structural Health Monitoring Studies on a Girder Bridge
   http://cive.seas.wustl.edu/wusceel/reujat/2003/Lucas.pdf

3. Analog Device's low power MEMS accelerometer.
   http://www.analog.com/en/prod/0,2877,ADXL202,00.html

4. Analog Device's low power MEMS ±2g accelerometer .
   http://www.analog.com/UploadedFiles/Evaluation_Boards_Tools/
   535395787ADXL203EB_0.pdf

5. A. Anandarajah, K. Moore, A. Terzis and I-J. Wang. Sensor networks for landslide detection In *SenSys'05*, Nov 2005.

6. Article from Hindu newspaper.
   http://www.thehindubusinessline.com/2005/05/17/stories/
   2005051700840700.htm

7. A. Sheth, K. Tejaswi, P. Mehta, C. Parekh, R. Bansal, S. Merchant, T. Singh, U. B. Desai, C. A. Thekkath and K. Toyama. SenSlide: a sensor network based landslide prediction aystem In *SenSys'05*, Nov 2005.

8. BIM-433 Radio Transreceiver module.
   http://www.radiometrix.co.uk/products/bimsheet.htm

9. B. Raman, K. Chebrolu, N. Madabhushi, D. Y. Gokhale, P. K. Valiveti and D. Jain. Implications of Link Range and (In)Stability on Sensor Network Architecture To appear in *WiNTECH 2006*, A MOBICOM'06 Workshop, Sep 2006.

10. BriMos:Bridge Monitoring System.
    http://www.brimos.com

11. Chipcon's CC2420 radio transreceiver chip.
    http://focus.ti.com/docs/prod/folders/print/cc2420.html

12. Cost of Kinemetrics Seismic accelerometers.
    http://www-bsac.eecs.berkeley.edu/archive/users/warneke-brett/
    pubs/warneke-IMECE02.pdf

13. Data Loggers/Analyzers.
http://www.campbellsci.com.au/structural-seismic
http:
//www.home.agilent.com/cgi-bin/pub/agilent/Product/cp_Product.jsp?
NAV_ID=-536900525.0.00&LANGUAGE_CODE=eng&COUNTRY_CODE=US
http://www.amplicon.co.uk/dr-prod2.cfm/secid/1/subsecid/10037.htm
http://www.monarchinstrument.com/dcloggers.htm

14. D. Culler, D. Estrin and M. Srivastava. Overview of Sensor Networks In *IEEE Computer*, Special Issue in Sensor Networks, Aug 2004.

15. Discussions for domain knowledge with Dr. K.K. Bajpai from Structures Lab, IIT Kanpur

16. C. Guo and A. Fano. Cargo Container Security using Ad Hoc Sensor Networks In *IPSN'05*, Apr 2005

17. C. Park, Q. Xie and P. H. Chou. DuraNode: Wi-Fi-based Sensor Node for Real-Time Structural Safety Monitoring In *IPSN'05*, Apr 2005.

18. CS344:Sensor Network Systems.
http://www.stanford.edu/class/cs344a

19. Deluge: TinyOS Network Programming.
http://www.cs.berkeley.edu/~jwhui/research/deluge

20. D. L. Mills. Internet time synchronization: The Network Time Protocol In Z. Yang and T.A. Marsland, editors, Global States and Time in Distributed Systems. IEEE Computer Society Press, 1994.

21. E. Sazonov, K. D. Janoyan,and R. Jha. Wireless Intelligent Sensor Network for Autonomous Structural Health Monitoring Smart Structures/NDE 2004, San Diego, California 2004.

22. Galloping Gertie.
http://www.nwrain.net/~newtsuit/recoveries/narrows/gg.htm

23. GPS Module
http://www.futurlec.com/GPS.shtml

24. G. Simon , M. Marti , . Ldeczi , G. Balogh , B. Kusy , A. Ndas , G. Pap , J. Sallai and K. Frampton. Sensor network-based countersniper system In *SenSys'04*, Nov 2004.

25. G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay and W. Hong. A macroscope in the redwoods In *SenSys'05*, Nov 2005.

26. G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring Volcanic Eruptions with a Wireless Sensor Network In Proc. European Workshop on Sensor Networks (*EWSN'05*), January 2005.

27. G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz and J. Lees. Deploying a Wireless Sensor Network on an Active Volcano In *IEEE Internet Computing*, March-April, 2006.

28. H. Hemanth. BriMon: Design and Implementation of Railway Bridge Monitoring Application Mater's Thesis, Department of Computer Science and Engineering, IIT Kanpur, India, May 2006.

29. H. Yang and B. Sikdar. A Protocol for Tracking Mobile Targets using Sensor Networks In IEEE Workshop on Sensor Network Protocols and Applications, May 2003.

30. Indian Railways Corporation Safety Plan:2003-2013
`http://www.indianrailways.gov.in/railway/deptts/safety/corp-safetyplan-idx.htm`

31. Indian Railways White Paper
`http://www.indianrailways.gov.in/railway/whitepaper/wh_paper_idx.htm`

32. Intel's Stargate systems.
`http://www.xbow.com/Products/productsdetails.aspx?sid=85`

33. J. Elson and D. Estrin. Time Synchronization for Wireless Sensor Networks In Proceedings of the 15th International Parallel & Distributed Processing Symposium, p.186, Apr 2001

34. J. Elson, L. Girod and D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts In the proceedings of the fifth symposium on Operating System Desgn and Implementation (*OSDI 2002*), December 2002.

35. J. Polastre, J. Hill and D. Culler. Versatile low power media access for wireless sensor networks In *SenSys'04*, Nov 2004.

36. K. Chintalapudi, J. Paek, O. Gnawali, T. S. Fu, K. Dantu, J. Caffrey, R. Govindan, E. Johnson, S. Masri. Structural damage detection and localization using NETSHM In *IPSN'06*, Apr 2006.

37. K. Lin, J. Hsu, S. Zahedi, D. C. Lee, J. Friedman, A. Kansal, V. Raghunathan and M. B. Srivastava. Heliomote: Enabling Long-Lived Sensor Networks Through Solar Energy Harvesting In *Sensys'05* , Nov2005.

38. L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman and M. Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea In *SenSys'05*, Nov 2005.

39. L. Schwiebert, S. K. S. Gupta and J. Weinmann. Research challenges in wireless networks of biomedical sensors In *MobiCom'01*, Jul 2001.

40. M. B. Srivastava, R. R. Muntz and M. Potkonjak. Smart kindergarten: sensor-based wireless networks for smart developmental problem-solving enviroments In *MobiCom'01*, Jul 2001.

41. M. Choudhury. Three Beacon Sensor Network Localization through Self Propogation Master's Thesis, Dept. of CSE, IIT Kanpur
`http://www.cse.iitk.ac.in/users/braman/students/2005/mocho.html`

42. Mica2 nodes.
`http://www.xbow.com/Products/productsdetails.aspx?sid=72`

43. M. Krger and C. Grosse. Structural health monitoring with wireless sensor networks. In *Otto-Graf-Journal* 14 (2004), pp 77-90.

44. M. Marti, B. Kusy, G. Simon and . Ldeczi. The flooding time synchronization protocol In *SenSys'04*, Nov 2004.

45. N. Mishra, K. Chebrolu, B. Raman and A. Pathak. Wake-on-WLAN. In *WWW*, May 2006.

46. N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan and D. Estrin. A wireless sensor network For structural monitoring In *SenSys'04*, Nov 2004.

47. P. Levis, D. Gay, and D. Culler. Active Sensor Networks In *NSDI'05*, May 2005.

48. P. Zhang, C. M. Sadler, S. A. Lyon and M. Martonosi. Hardware design experiences in ZebraNet In *SenSys'04*, Nov 2004.
Additional information:
http://mailman.dtnrg.org/pipermail/dtn-interest/2004-February/001324.html

49. R. Adler, M. Flanigan, J. Huang, R. Kling, N. Kushalnagar, L. Nachman, C. Wan and M. Yarvis. Intel mote 2: an advanced platform for demanding sensor network applications In *SenSys'05*, Nov 2005.

50. Railway Minister's Budget speech 2005-2006
http://www.indianrailways.gov.in/railway/budget2005-2006-new.htm

51. R. Gass, J. Scott and C. Diot.
Measurements of In-Motion 802.11 Networking In *WMCSA'06*, Apr 2006.

52. R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson and D. Culler. An analysis of a large scale habitat monitoring application In *SenSys'04*, Nov 2004.

53. S. Ganeriwal, D. Ganesan, H. Shim, V. Tsiatsis and M. B. Srivastava. Estimating clock uncertainty for efficient duty-cycling in sensor networks In *SenSys'05*, Nov 2005.

54. S. Ganeriwal, R. Kumar, M. B. Srivastava. Timing-sync protocol for sensor networks In *SenSys'03*, Nov 2003.

55. Silicon Design Low Noise Capacitive accelerometers.
http://www.silicondesigns.com/1221.html

56. Single axis Forced Balance Accelerometers from Kinemetrics (obsolete). http://www.kinemetrics.com/download_Content.asp?newsid=136 http://www.pemed.com/lab/lab.htm

57. Structural Monitoring Systems.
http://www.keynes-controls.com/products.htm
http://www.aes-group.com/index.php?id=92
http://www.brimos.com

58. Structural Health Monitoring of Golden Gate Bridge (using MEMS based accelerometers).
http://www.cs.berkeley.edu/~binetude/ggb/
http://www.citris-uc.org/system/files?file=pakzad-ggb.pdf

59. Sunturn Ultra Alkaline Batteries
http:
//www.sunturnbattery.com/applications/ultra-alkaline-battery.php

60. Telos specifications.
http://www.tinyos.net/scoop/special/hardware\#telos

61. TinyOS help list email exchanges regarding observed data rates on Tmote and MicaZ platforms.
https://mail.millennium.berkeley.edu/pipermail/tinyos-help/2005-May/009448.html

62. TI's MSP430 Low power 16bit microcontrollers.
http://focus.ti.com/docs/prod/folders/print/msp430f1611.html

63. Tmote-sky nodes.
http://www.moteiv.com/products-tmotesky.php

64. The TinyOS 2.x Working Group TinyOS 2.0 In *SenSys'05*, Nov 2005.
http://www.tinyos.net/scoop/story/2006/2/10/93050/8222

65. TinyOS operating system for Sensor Networks.
www.tinyos.net

66. V. Mehta and M. EL Zarki. A Bluetooth Based Sensor Network for Civil Infrastructure Health Monitoring *Wireless Networks*, 10: 401-412, 2004

67. WiFi and ethernet enabled single board computers.
http://www.soekris.com

68. Wireless Data Logger: Amplicon.
http://www.amplicon.co.uk/dr-prod3.cfm/subsecid/10037/secid/1/groupId/11809.htm