



**University of Fort Hare**  
*Together in Excellence*

# **Development of a web-based interface for a wireless sensor network monitoring system**

A thesis submitted in fulfilment of the requirements of the degree of

**Master of Science in Computer Science  
at**

University of Fort Hare

by

**Sibukele Gumbo**

**Supervisor: Prof HN Muyingi**

December 2007

## **Acknowledgements**

I would like to extend my gratitude to my supervisors Prof HN Muyingi and Prof A Terzoli for the guidance and help they gave me for this work.

I would also like to thank my sponsors, Telkom Centre of Excellence and the National Research Foundation for making my dream of pursuing a Masters degree a reality by giving me financial assistance

I would also like to thank my family, Munya , Jim Chadwick, my classmates, Honours students, Odilo, Sibho, Mabanza, Nyaladzi, Peter, Mthimbani, Tinah , Ron , Mamello and everyone else that knows me....

## Declaration

I, the undersigned declare that the work contained in this dissertation is my own original work and has not previously in its entirety or in part been submitted at any educational institution for a similar or any other degree awarded.

Signature.....

Date.....

## **Publications**

Gumbo, S. Muyingi, H. 2006. Investigating wireless sensor network performance for remote monitoring of a long distance power transmission overhead line. SATNAC conference, Cape Town, South Africa

Gumbo, S., Muyingi, H. 2007. Development of a web-based interface for remote monitoring long-distance power transmission overhead lines. SATNAC conference, Sugar Beach Resort, Mauritius

## Acronyms

ACK	Acknowledgement
ADC	Analog to Digital Converter
AFIS	Advanced Fire Information Service
API	Application Programming Interface
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSIR	Council for Scientific and Industrial Research
CSMA/CA	Carrier Sensor Multiple Access with Collision Avoidance
CSS	Cascading Style Sheet
DARPA	Defense Advanced Research Project Agency
DBLogger	Database Logger
DRC	Democratic Republic of Congo
DSSS	Direct-Sequence Spread Spectrum
EJB	Enterprise Java Beans
FHSS	Frequency Hopping Spread Spectrum
GUI	Graphic User Interface
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IPC	Interprocess Communication
ISM	Industrial-Scientific-Medical
JDBC	Java Database Connectivity
J2EE	Java 2 Enterprise Edition
J2SE	Java 2 Platform Standard Edition
JSP	Java Server Page
JVM	Java Virtual Machine
LED	Light Emitting Diode
MAC	Media Access Control
MCU	Micro Controller Unit
MEMS	Micro-electro Mechanical Sensor

## Development of a web-based interface for a wireless sensor network monitoring system

MHz	Mega Hertz
MVC	Model-View-Controller
ORDBMS	Object-relational Database Management System
OSI	Open System Interconnect
PHY	Physical layer
PLC	Power Line Communication
RAM	Random Access Memory
RFID	Radio Frequency Identification
Rx	Receiver
SCADA	Supervisory Control and Data Acquisition Systems
SQL	Structured Query Language
TCP	Transmission Control Protocol
TinyDB	Tiny DataBase
TinyOS	Tiny Operating System
TinyViz	Tiny Visualizer
TOSSIM	Tiny Operating System Simulator
Tx	Transmitter
UDP	User Datagram Protocol
WESTCOR	Western Corridor
WIFI	Wireless Fidelity
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network

Development of a web-based interface for a wireless sensor network monitoring system

## **Abstract**

In the recent past, wireless sensor technology has undergone advancements in its autonomous data collecting aspects, and has become an area worth investigating in relation to structural monitoring applications. The system described in this thesis aims at acquiring, storing and displaying overhead transmission line related data collected from a wireless sensor network. Open source tools were used in its development and implementation. The inherent linearly aligned topology of transmission line monitoring devices is not without shortcomings; hence analysis of linear node placement, hardware and software components was carried out to determine the feasibility of the system. Their limited data processing capabilities has motivated the development of a post processing wireless sensor application in order to present any collected structural data in an understandable format.



# Table of Contents

<b>CHAPTER 1 RESEARCH INTRODUCTION .....</b>	<b>15</b>
1.1. Introduction .....	15
1.2. Research problem.....	16
1.3. Related work.....	17
1.4. Research objectives and methodology.....	18
1.5. Research deliverables .....	19
1.6. Organisation of dissertation.....	19
<b>CHAPTER 2 WIRELESS SENSOR NETWORKS.....</b>	<b>21</b>
2.1. Introduction .....	21
2.2. Comparison of protocol stacks.....	22
2.3. The wireless sensor network protocol stack.....	23
2.3.1. Physical layer .....	24
2.3.2. Media Access Control layer .....	25
2.3.3. Network layer.....	27
2.3.4. Transport layer.....	30
2.3.5. Middleware layer.....	30
2.3.6. Application layer.....	31
2.3.7. Management planes.....	32
2.4. Commercial wireless sensor prototypes.....	33
2.5. Advantages of using wireless sensor networks.....	35
2.6. Disadvantages of using wireless sensor networks.....	35
<b>CHAPTER 3 SYSTEM REQUIREMENTS .....</b>	<b>36</b>
3.1. Introduction .....	36
3.2. Functional requirements.....	38
3.3. Non functional requirements .....	39
3.4. Wireless sensor system components .....	40

## Development of a web-based interface for a wireless sensor network monitoring system

3.5.	Wireless sensor data acquisition system.....	40
3.6.	Wireless sensor network application.....	42

## **CHAPTER 4 DATA ACQUISITION SYSTEM..... 43**

4.1.	Introduction .....	43
4.2.	Hardware platform architecture for a Mica2 node .....	45
4.2.1.	Computing subsystem .....	45
4.2.2.	Sensing subsystem.....	46
4.2.3.	Power supply subsystem .....	46
4.2.4.	Communication subsystem.....	47
4.3.	Architecture of the base station node .....	48
4.4.	Comparison of wireless sensor operating systems.....	51
4.5.	TinyOS and the development environment.....	52
4.5.1.	Tiny Microthreading operating system .....	52
4.5.2.	SerialForwarder.....	52
4.5.3.	TinyOS Messages .....	53
4.5.4.	Multihop messages .....	56
4.5.5.	Wireless sensor network simulators .....	57
4.5.6.	TOSSIM.....	57
4.5.7.	Sensor network query processing .....	58
4.5.8.	TinyDB.....	59

## **CHAPTER 5 THE WIRELESS SENSOR NETWORK APPLICATION ..... 60**

5.1.	Introduction .....	60
5.2.	Presentation layer .....	62
5.2.1.	Netbeans Integrated Development Environment (IDE) .....	62
5.2.2.	Servlets and Java Server Pages .....	63
5.2.3.	Alert indicators.....	64
5.2.4.	JFreeChart.....	64
5.3.	Data access layer.....	65
5.3.1.	Apache Tomcat web server .....	65
5.4.	Data storage system .....	65
5.4.1.	PostgreSQL.....	66

<b>CHAPTER 6 SYSTEM DESIGN .....</b>	<b>68</b>
6.1. Introduction .....	68
6.2. Wireless sensor design architecture .....	70
6.3. Base station design architecture .....	71
6.4. Wireless sensor application design architecture.....	72
<b>CHAPTER 7 WIRELESS SENSOR APPLICATION IMPLEMENTATION.....</b>	<b>74</b>
7.1. Introduction .....	74
7.1.1. Developer Installations.....	75
7.2. Displaying PostgreSQL data.....	77
7.2.1. Java Server Pages and servlets .....	78
<b>CHAPTER 8 DATA ACQUISITION IMPLEMENTATION AND TESTING .....</b>	<b>89</b>
8.1. Introduction .....	89
8.2. Hardware and software installations .....	90
8.3. Hardware evaluation of Mica2 nodes using continuous querying.....	98
8.3.1. Transmission range of Mica2 nodes.....	98
8.3.2. Battery lifetime of Mica2 nodes .....	102
8.3.3. Scalability of Mica2 nodes.....	104
8.4. Evaluation of response time in wireless sensor networks .....	108
8.4.1. Response time of TinyDB in Mica2 nodes .....	108
8.5. Evaluation of event based querying .....	110
8.6. Experiment results discussions .....	111
<b>CHAPTER 9 CONCLUSIONS AND FUTURE WORK.....</b>	<b>113</b>
9.1. Introduction .....	113
9.2. Project Summary .....	114
9.3. Project Evaluation.....	115
<b>REFERENCES .....</b>	<b>116</b>

## List of Figures

Figure 2.1 Wireless technology protocol stacks .....	23
Figure 3.1 Data acquisition tools .....	41
Figure 3.2 Wireless sensor application tools .....	42
Figure 4.1 A Mica2 node.....	44
Figure 4.2 Hardware architecture for a sensor node .....	45
Figure 4.3 Base station node.....	49
Figure 4.4 Reference model of the architecture of the base station software.....	50
Figure 4.5 SerialForwarder program.....	53
Figure 4.6 TinyOS message components.....	54
Figure 4.7 Multihop message components.....	56
Figure 5.1 Reference model for the wireless sensor network application.....	61
Figure 5.2 The Netbeans IDE interface .....	62
Figure 5.3 PostgreSQL configuration file .....	66
Figure 5.4 Communications between the client host and the database .....	67
Figure 6.1 Overall wireless sensor network design architecture .....	70
Figure 6.2 Sensor node functional loop .....	71
Figure 6.3 Base station functional loop .....	72
Figure 6.4 Web interface functional loop.....	73
Figure 7.1 JFreeChart classes .....	75
Figure 7.2 PostgreSQL jdbc driver.....	77
Figure 7.3 Visual look of the web-based interface.....	81
Figure 7.4 Main.jsp .....	82
Figure 7.5 Tabular and graph display .....	88
Figure 8.1 TinyOS packages for sensor nodes.....	90
Figure 8.2 Sensor node attached to a programming board.....	91
Figure 8.3 Mica2 node with sensor board.....	93
Figure 8.4 TinyViz graphic user interface.....	94
Figure 8.5 SerialForwarder program.....	95
Figure 8.6 TinyDB graphic user interface.....	96
Figure 8.7 PgAdmin III interface .....	97
Figure 8.8 Transmission range experiment.....	99
Figure 8.9 Transmission ranges of Mica2 nodes (sunny weather conditions) .....	100
Figure 8.10 Transmission ranges of Mica2 nodes (rainy weather).....	101
Figure 8.11 Battery life experiment .....	102
Figure 8.12 Battery lifetime of wireless sensor nodes .....	103

## Development of a web-based interface for a wireless sensor network monitoring system

Figure 8.13 Scalability of Mica2 nodes experiment .....	104
Figure 8.14 Scalability results of Mica2 nodes at 80 metres .....	105
Figure 8.15 Scalability results of Mica2 nodes at 40 metres .....	106
Figure 8.16 Scalability results in TOSSIM at 100 feet (33 metres).....	107
Figure 8.17 Topology for TinyDB response time experiment. ....	108
Figure 8.18 Response time for TinyDB results in Mica2 nodes placed 40 metres apart.....	109
Figure 8.19 Response time for TinyDB results in TOSSIM nodes placed 10 feet apart .....	109
Figure 8.20 Event based querying of sensor nodes .....	110
Figure 8.21 Event based query results.....	111

## List of Tables

Table 2.1 Performance of current wireless technology. ....	24
Table 2.2 Raw data packet format.....	28
Table 2.3 Raw data packet field descriptions.....	28
Table 2.4 Characteristics of commercial wireless sensor prototypes .....	34
Table 4.1 Mica2 power consumption.....	47
Table 4.2 Comparison of frequencies and ranges of Mica sensor systems.....	48
Table 4.3 Comparison of wireless sensor network operating systems.....	51
Table 4.4 TinyOS message format .....	55
Table 4.5 TinyOS message field descriptions.....	55
Table 4.6 Multihop message format.....	56
Table 4.7 Multihop message component description .....	56
Table 4.8 Comparison of sensor network database management systems.....	58
Table 7.1 Main JSP pages implementations.....	78
Table 7.2 Main servlet implementations .....	79
Table 7.3 Comparison of ArrayLists to Arrays.....	83
Table 8.1 A PostgreSQL table .....	97

# Chapter 1

## Research Introduction

### 1.1. Introduction

In recent years, there has been increasing interest in the adoption of wireless sensor technologies in the development of applications for monitoring structural systems. These devices have not been widely deployed within the African continent, but their continued appraisal by researchers world wide has fuelled interest in investigating the possibility of integrating them into ongoing applications instead of traditional tethered methods (CSIR, 2007). Structural wireless sensor monitoring systems have gained much attention in research and public communities because of the remarkable progress made in their processor, memory and radio technology that are now capable of performing sensing, computing and wireless communication with minimal human involvement (Major, 2006).

## 1.2. Research problem

The significant driver of wireless sensor networks is vested in environmental monitoring, but an extension to other applications susceptible to these environmental conditions is possible. Power utility sectors in Southern Africa are motivating research into the investigation of the use of wireless sensors to monitor their overhead electrical transmission systems (CSIR, 2007). The continued outage of their exposed systems due to these environmental, as well as operational and man made events requires an active monitoring system to report any anomalies or failures so that efficient service delivery is satisfied (Eskom, 2007). To monitor power line systems, the wireless sensor nodes are typically placed in a straight line along a long and narrow area, with a control centre at the end where further data processing occurs, thereby creating a many-to-one communication model. A temporal database at the control centre provides storage for the data collected about the structure in one place using a common format (Das, 2004).

As with any new technology, wireless sensor networks have many issues that need to be addressed before their incorporation into applications. Documented challenges in these networks include the absence of post processing algorithms, sensor node constraints, media link constraints, variable data packet deliveries and security issues (Alkydiz, 2002). The lack of algorithms to process the data delivered at the control centre invariably leads to the display of overwhelming, continuous values which make structural assessment a difficult and time consuming task that can only be accomplished by highly-trained engineers. It became necessary to develop a post processing algorithm capable of organizing and displaying sensor data retrieved from the network for fast and efficient data access.

In order to execute the post processing application, a linear wireless sensor network to produce sensor data was deployed under natural and simulated environments. The results from wireless sensor nodes were not always delivered to the control centre due to the sensor node and media link constraints. In order to identify some of the constraints that affected



wireless sensor network packet delivery, some experiments were carried out in both environments.

### **1.3. Related work**

Successful implementations using sensor technologies to measure variables and operational conditions in high-voltage lines began in the seventies. The devices measured voltage, current and temperature parameters which could be transferred over a wireless link (Leskovar, 1977). In 1984, another similar wireless sensor data acquisition system was developed to measure and transfer quantities in electrical power conductors. This system was an improvement to the latter as it allowed management of the sensor nodes by a remote control centre (Fernandes, 1984).

Recently, field surveys and tests of the use of distributed sensors for fault detection at the substation level of power transmission lines were carried out. The tests that were performed represent concrete examples of the applicability of wireless communication in monitoring of industrial environments parameters, such as those in long distance power transmission lines (Nordman, 2004).

Current transmission line systems parameters in the region are monitored using power line communication (PLC), aerial optical fibre cable and satellite sensors (CSIR, 2007)(Nordman, 2004). The burden, complexity and expense of laying cable to instrument monitoring large structures are quite high. In addition, the power line-based telecommunication medium has limited reliability.

In the attempt to combat the escalation of fires causing transmission line faults, Eskom and the Council for Scientific and Industrial Research (CSIR) in South Africa developed a sensor satellite monitoring system called Advanced Fire Information Service (AFIS). This system contains post processing algorithms to effectively identify which regions have conditions that are favourable for the spread of veldt fires or those already affected by fires. The South

African Broadcasting Corporation (SABC) implemented this system and was the first public broadcaster to televise active fire maps as part of its weather reports (CSIR, 2007).

## **1.4. Research objectives and methodology**

This project involved the deployment of a post processing application to organize and manage data retrieved from a linearly aligned wireless sensor network. Since the number of wireless sensor nodes available for experiments was limited, a simulator was used to perform tests requiring a larger number of nodes. Simulating a wireless sensor network also helped in viewing debug messages and monitoring network traffic to identify any problems which may go unnoticed during a real world deployment of nodes (Lewis, 2003).

The implementation of the prototype involved the following steps:

- **Understanding the components of wireless sensor networks, wireless sensor devices and their functions.**

Research was undertaken in the area of wireless sensor networks to investigate wireless sensor components, commercially available wireless sensors nodes, simulators and the software available for these devices. Identifying the challenges and constraints in wireless sensor networks and their simulators was also necessary to equip and give guidance to the author on the expected performance of devices when practically deployed.

- **Defining a post processing paradigm to access sensor data.**

Access to a temporal database that stores sensor readings was achieved using a web-based interface to analyze, sort and visualize data. The typical components of the monitored system needed to be investigated in order to define and implement a suitable post processing algorithm that was suitable for wireless sensor networks.

- **Investigating the feasibility of a wireless sensor devices and simulators in the delivery of transmission line-related quantities.**

A physical deployment of wireless sensor nodes and a simulator was made. Experiments that determine the suitability of a linear sensor network in delivering typical transmission line quantities were carried out. A comparison between physical and virtual node performances was also made since the data they collect is recorded in the same temporal database.

- **Ascertaining the usefulness of the post processing prototype developed.**

The web-based interface that displays information was deployed in order to determine its ability to retrieve sensor data from the temporal database and display it.

## **1.5. Research deliverables**

The project has provided an opportunity for the author to explore wireless sensor networks and have an in-depth understanding of the properties of sensor devices. Experiments with linearly aligned wireless sensor networks were carried out and documented to provide the research community with an evaluation on their performances. A web-based prototype was developed and tested to organise and manage sensor data.

## **1.6. Organisation of dissertation**

The remainder of this dissertation is organised as follows:

Chapter 2 identifies the requirements of the proposed wireless sensor monitoring system and integrates technical solutions into each requirement.

Chapter 3 introduces the concepts of wireless sensor networks giving emphasise to their protocol stack, limitations and commercial sensor nodes currently available on the market.

## Development of a web-based interface for a wireless sensor network monitoring system

Chapter 4 describes the hardware and software components of Mica2 sensor nodes and the TOSSIM simulator, which had been selected as the data collecting devices in the wireless sensor prototype.

Chapter 5 describes the application developed to post process and display data acquired from the wireless sensor network on a web interface.

Chapter 6 illustrates the overall system design architecture for the complete prototype consisting of the data acquisition sensor nodes and the web-based wireless sensor application.

Chapter 7 describes the experiments carried out to implement and evaluate the data acquisition system.

Chapter 8 describes the major classes, methods and functions essential to implement the wireless sensor application.

Chapter 9 summarises the work undertaken in this project, including all successes and failures in the deployment of the wireless sensor network. Possible future work is then suggested.

## Chapter 2

### Wireless sensor networks

*This chapter provides an opportunity to give a description of wireless sensor networks, their protocol stack and commercial sensor prototypes that are currently available on the market. The performance parameters that will be used to evaluate the use of wireless sensor system are also given.*

#### 2.1. Introduction

A wireless sensor network consists of coordinated sensor nodes, also known as ‘motes’, that collectively observe environmental changes or operate as central control units, in which case they are referred to as sink nodes (Surve, 2006). Any sensed environmental changes are to some extent processed within the node then communicated to neighbours, toward the sink node, using a flexible wireless architecture (Pathan, 2006). David (Culler, 2004) provides a very apt description of these sensor devices, *“The new genre of embedded software is characterized by being agile, self organizing, critically resource constrained, and communication-centric on numerous small devices operating as a collective, rather than highly engineered to a particular standalone task on a device sized to suit”*.

## Development of a web-based interface for a wireless sensor network monitoring system

Initial wireless sensor network research was motivated by military surveillance systems, dating back to at least 1978 by the Defense Advanced Research Project Agency (DARPA). These nodes didn't include system software because of the scarce resources in the wireless sensor nodes and simplicity of the required applications (Callaway, 2004). Currently, more complex applications such as in habitat and ecological sensing, structural monitoring and smart spaces, emergency responses and remote surveillance require system software to ease the control of resources and increase the predictability of executions (Suramian, 2005).

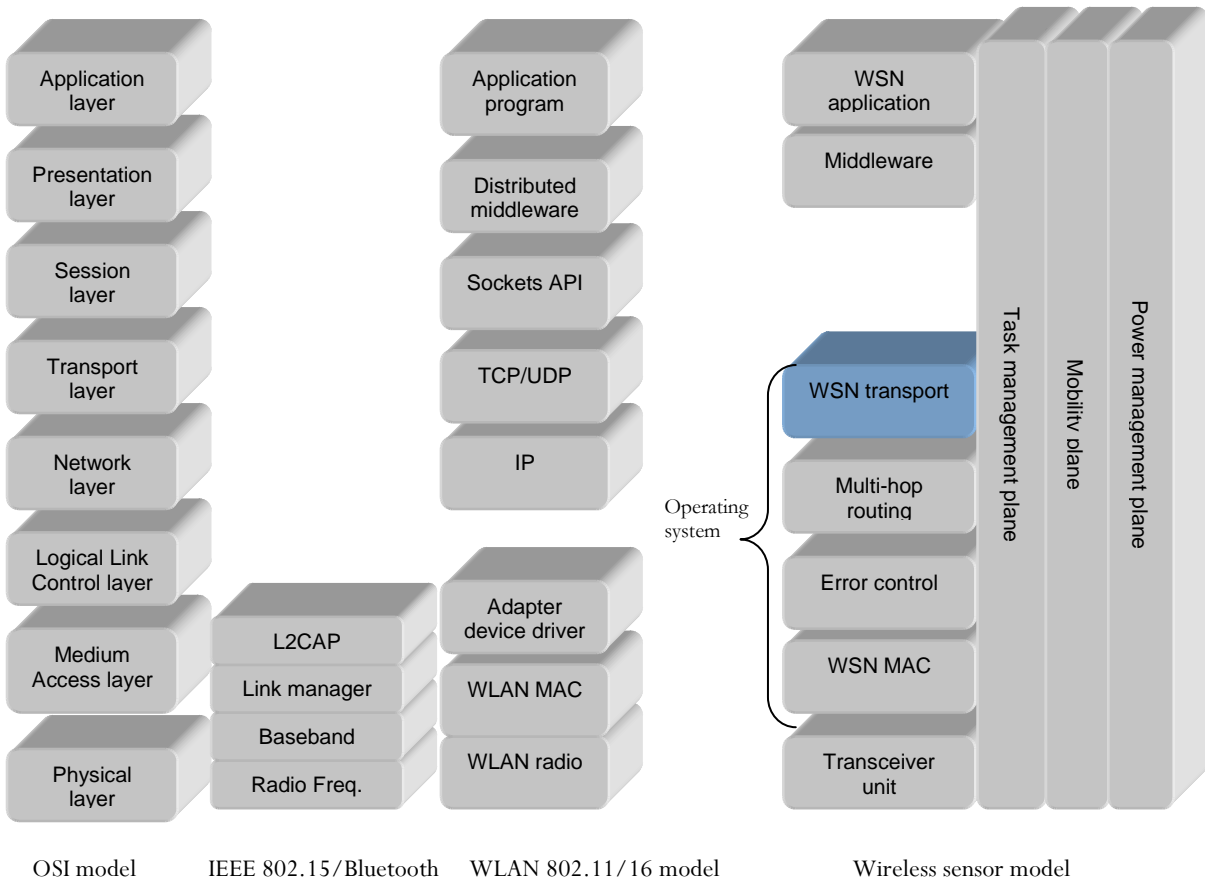
Wireless sensor networks currently available still differ from ad hoc networks in terms of:

- Limited power and processing capabilities in wireless sensor nodes.
- Broadcast communication and paradigm.
- Large number of devices required in most applications.
- Resource constraints that make them more susceptible to failures (Mainwaring, 2002).

## **2.2. Comparison of protocol stacks**

The wireless sensor network protocol stack can be compared to three widely used stacks i.e. the OSI model, Bluetooth and a Wireless Local Area Network model (WLAN) as shown in Figure 3.1 (Kuorilehto, 2005).

## Development of a web-based interface for a wireless sensor network monitoring system



**Figure 2.1** Wireless technology protocol stacks

### 2.3. The wireless sensor network protocol stack

The Institute of Electrical and Electronics Engineers (IEEE) has defined a standard for low-power, low-data-rate wireless sensor data transmission, mainly built over a Zigbee stack (Gutierrez, 2001). There is no unified protocol stack for wireless sensor networks, but one can be built from a collection of known protocol functions. The lower layers in this stack, i.e. from the wireless sensor network Media Access Control (MAC) protocol to the transport protocol, represent the operating system whose role is to hide any implementation details of the underlying wireless sensor hardware from the upper software layers (Alkydiz, 2002). The layers above the operating system, i.e. the middleware and the application layer, are

responsible for the operation of the wireless sensors (e.g. data collection, storage, or communication) and the execution of embedded data interrogation methods (Alkydiz, 2002).

### 2.3.1. Physical layer

The physical layer in wireless sensor networks is responsible for frequency selection, carrier frequency generation, signal detection, modulation and data encryption. It represents the actual hardware communication interface (Alkydiz, 2002). The physical layer may be implemented using Zigbee (most popular), Bluetooth and WiFi stacks, whose characteristics are shown in Table 2.1 (Gutierrez, 2001).

WIRELESS TECHNOLOGY	ZIGBEE	BLUETOOTH	WI-FI
Standard	802.15.4	802.15.1	802.11b
Memory requirements	4 – 32 KB	250 KB +	1MB +
Maximum battery life	Years	Days	Hours
Maximum data rate	250 Kb/s	1 Mb/s	11Mb/s
Maximum nodes per master	65000+	7	32
Maximum range	300m	100m	100m
Maximum frequency	2.4 GHz	2.4 GHz	2.4 GHz

**Table 2.1** Performance of current wireless technology.

### ZIGBEE

Zigbee was specifically designed to develop and solve problems in wireless sensor networks, as well as to allow the implementation of a wireless personal area network (WPAN). Its stack was founded over the IEEE 802.15.4 standard which was developed to “*provide a standard for ultra-low complexity, ultra-low cost, ultra-low power consumption, and low data rate wireless connectivity among inexpensive devices*”. Zigbee specifies only the physical (PHY) and media access control (MAC) layers of other models, defining the radio frequency and communication components to neighbouring devices (Alkydiz, 2002)(Garcia, 2006). It allows data rates of 20-250 kbps depending on the frequency band being used: 868 MHz at



## Development of a web-based interface for a wireless sensor network monitoring system

20 Kb/s, 915 MHz at 40 Kb/s, and 2.4 GHz at 250 Kb/s (Gutierrez, 2001). Zigbee devices employ a direct-sequence spread spectrum (DSSS) scheme because of its lower cost analog implementation as compared to frequency-hopping spread spectrum (FHSS) (Gutierrez, 2001).

## **BLUETOOTH**

The Bluetooth standard, IEEE 802.15, is an infrastructure-less, short-range radio frequency wireless system used in the development of ‘Pico’ networks (Bramwell, 2006). Bluetooth originated at Ericsson in 1994, together with its software and hardware definitions for short range (10 – 100 meters), low power (typically around 20dBm) and low cost radio links. Unlike other wireless standards, Bluetooth includes data link layer and application layer definitions in its implementations for product developers (Bramwell, 2006).

### **2.3.2. Media Access Control layer**

All nodes in a wireless sensor network are data sources and routers; since data traffic originates from and is routed through every node in the network (Alkydiz, 2002). The development of an application for remote monitoring transmission lines, as previously mentioned is linear; hence more routing occurs at the nodes that are close to the control centre. In order for the wireless sensor network to function correctly in this scenario, the media access control mechanism in these devices should allocate bandwidth according to the proximity of the node to the sink.

The absence of a unified protocol stack for wireless sensor networks means there is no specific design for the media access protocol, and is at the discretion of the wireless sensor node manufacturer (Anushruthi, 2005). Sensor nodes that use the Zigbee stack implement the carrier sense multiple access collision avoidance (CSMA/CA) mechanism which essentially takes a ‘listen-before-talk’ approach to detect if any radio traffic is in its vicinity (Gutierrez, 2001). Nodes may hear a ‘clear line’ simultaneously, and begin an

## Development of a web-based interface for a wireless sensor network monitoring system

unsynchronized transmission if a continuous query is issued, or attempt to report the same event if an event-based query is issued. If a collision occurs, some messages may be corrupted, resulting in sensor readings being lost at any level of the network from the node to the base station (Anushruthi, 2005) (Ye, 2003). The half duplex nature of these devices means that they cannot detect collisions (Madden, 2003). Many wireless sensor network implementations do not incorporate message retransmissions because of the constraints that exist in these networks (Alkydiz, 2002).

Interest mostly lies in the most recent measurement from each sensor in the network. As a result, any undelivered measurements are ignored. Sensor nodes transmit measurements with the state of the internal counter (timestamp) to their control centre; therefore programs built should include the Structured Query Language (SQL) query:

```
Select DISTINCT ON (nodeid) * from ....
```

to select the most recent data entry from each distinct sensor node.

For the successful deployment of a wireless sensor network, it is necessary to incorporate some mechanisms that limit the number of collisions or demand on the wireless link by the nodes. These include:

- Reduction of nodes in the network.
- Decrease in the temporal sampling rate on the nodes.
- The implementation of on-board data processing capabilities in the nodes that broadcast only summarized or event triggered data (Gutierrez, 2001).

As a result, it may be useful to investigate the tradeoffs between the frequency of sensor sampling and data broadcasting, together with the associated cost in terms of energy, memory usage, CPU utilization and information loss due to on-board pre-processing and wireless transmissions.

Challenges in the media access of sensor networks are as follows:

- The traffic originating from a node has to compete with the traffic being routed through that node.
- The probability of data corruption and contention is higher for the nodes which reside further from the control centre.
- Energy is invested in every packet that is routed through every node, therefore the longer the packet is routed, the more expensive it is to drop that packet.
- An undetected node might exist in the network, which might result in its unexpected contention for bandwidth of the node traffic and that being routed through it (Anushruthi, 2005).

### **2.3.3. Network layer**

The network layer in wireless sensor networks is responsible for route discovery and delivery of data packets supplied by the transport layer (Alkydiz, 2002) (Martincic et al, 2005). The network of sensors nodes transmit data packets directly to the base station through a single hop or through other intermediate nodes using a multihopping scheme by radio, infrared or optical media. Implementation of an efficient multihop system requires optimal routing to facilitate the discovery of the shortest route, reduced power consumptions and reliable packet transmissions (Akkaya and Younis, 2005).

The selection of the network layer routing protocol employed in wireless sensor devices is constrained by the limited nature of the sensor nodes and the topology relevant to the wireless sensor network application (Suramania, 2005). A typical raw data packet in the network layer is illustrated in Table 2.2 and Table 2.3 (Thom, 2005).

SYNC-BYTE	PACKET TYPE	PAYLOAD DATA	SYNC_BYTE
0	1	2...n-1	n

**Table 2.2 Raw data packet format**

BYTE NUMBER	FIELD	DESCRIPTION
0	Packet frame sync byte	Always 0x7E
1	Packet Type	<p>There are five known packet types:</p> <ul style="list-style-type: none"> <li>• P_PACKET_NO_ACK (0x42) – user packet with no ACK required</li> <li>• P_PACKET_ACK (0x41) – user packet. ACK required</li> <li>• P_ACK (0x40) – the ACK response to a P_PACKET_ACK packet</li> <li>• P_UNKNOWN (0xFF) – an unknown packet type</li> </ul>
2..n-1	Payload data	In most cases will be a TinyOS Message of varying length with a maximum of 36 bytes.
N	SYNC_BYTE	Always 0x7E

**Table 2.3 Raw data packet field descriptions**

## **ROUTING TECHNIQUES IN WIRELESS SENSOR NETWORKS**

A wireless sensor routing tree is rooted at the base station, and is used to disseminate queries or collect query results from the network (Maroti, 2004). It is initialized using a command issued by the control centre and is maintained by the exchange of requests and responses between the sink node and the sensor nodes using the following techniques:

- **Flooding**

A wireless sensor network employing flooding as its routing technique allows a sensor node which receives data packets to broadcast them to all its neighbours. These broadcasts continue until a packet arrives at its destination or the maximum number of hops in the packet has been reached. Minimum computation occurs when flooding is used and the topology maintenance of the discovered route to the control centre is unnecessary (Suramania, 2005) (Maroti, 2004).

**Drawbacks of flooding include:**

- Consumption of large amounts of energy when broadcasting packets without consideration for wireless sensor network energy constraints.
- Sending of similar packets towards the base station if two overlapping nodes sense in the same region (Suramania, 2005) (Maroti, 2004).

- **Gossiping**

When a wireless sensor network employs gossiping as its routing technique, each node maintains a list of neighbours and routing information about the connectivity of those neighbours to the rest of the network (Suramania, 2005) (Maroti, 2004). When a node receives a data packet, it randomly selects a neighbour from the list to which it forwards the packet, avoiding the problem of implosion. This takes place until the packet reaches its destination or the maximum hop count is reached.

**Drawbacks of gossiping include:**

- The possibility that the data packet does not reach its destination, given the random nature of neighbour selection (Suramianian, 2005).

### **2.3.4. Transport layer**

The transport layer in wireless sensor networks maintains the flow of data within a network. It also enables end users to access sensor system data through the Internet and other external devices. Data can be transmitted using Transmission Control Protocol (TCP), User Datagram Protocol (UDP) or any other custom transport layer protocol (Alkydiz, 2002). When TCP is used to transmit data, no messages are lost, and if any collisions occur in the vicinity of a broadcasting node, there is an attempt to retransmit the message. UDP, on the other hand does not guarantee the delivery of sensor data (Hill, 2003).

Applications built using wireless sensor devices require minimum transmissions in order to conserve as much energy as possible. In addition, only the most recent measurements in the network are important, hence, messages are sent without requests for acknowledgements so that any lost messages are ignored. The reduction in transmission of data packets requiring acknowledgements (P\_PACKET\_ACK (0×41) in Table 2.3) allows for faster transfer of data due to less contention for the wireless link (Bajcsy, 2005).

### **2.3.5. Middleware layer**

Middleware in a wireless sensor network is equivalent to the presentation layer in the OSI model. It sits between the operating system and the application layer in the wireless sensor stack, thereby gluing together the network hardware, operating systems, and applications (Alkydiz, 2002).

The functions of the middleware layer include:

- Support for the development, deployment and execution of sensor based applications. It achieves this by simplifying complex high level tasks which need to be communicated to nodes or aggregation of results from the network (Alkydiz, 2002) (Romer, 2002).
- Support for adaptive and efficient resource usage by controlling data processing in nodes in the attempt to prolong the up-time of a sensor network (Alkydiz, 2002) (Romer, 2002).

### **2.3.6. Application layer**

The application layer aims at creating programming capabilities for the efficient extraction, manipulation, transport and representation of information derived from sensor data (Alkydiz, 2002). Some of the functional components within this application layer are sensor detection, data collection, signal processing and data diffusion. Protocols that exist in the application layer for the successful deployment of sensor nodes include:

- **Sensor Management Protocol (SMP)**

The sensor management protocol enables the sensor network to be accessed through other networks such as the Internet allowing system administrators to interact with these sensor networks. SMP provides support for:

1. Querying the sensor network configuration and reconfiguring the network.
2. Introducing rules related to data aggregation from nodes,
3. Time synchronizing of the wireless sensor nodes,
4. Turning the nodes on and off,
5. Sensor authentication, key distribution and security in data communications. (Raghavendra, 2004).

- **Sensor query and data dissemination protocol (SQDDP)**

SQDDP provides a platform for user applications to issue queries and receive responses to them. These queries are generally issued to sensor networks using attribute-based or location-based naming. For instance, “*select the nodeid of all nodes with a temperature higher than 60 degrees*” is an attribute-based query and similarly, “*Select the temperatures read by the nodes in Region P*” is an example of location-based naming (Raghavendra, 2004).

### **2.3.7. Management planes**

The power, movement and task distribution among the nodes is monitored using power, mobility and task management planes. The management planes are needed for the sensor nodes to work together in a power efficient way, route data in mobile sensor networks and share resources between sensor nodes to prolong the up-time of the network (Alkydiz, 2002).

The power management plane monitors how the sensor node uses its power. If the power level in a sensor is low, the sensor node transmits a message to its neighbours that it cannot participate in message routing, reserving any remaining power for sensing. To avoid duplicating messages, a sensor node may also turn off its receiver after receiving a message from one of its neighbours.

The mobility management plane detects and registers the movement of sensor nodes so that a route to the control centre is always maintained. The sensor nodes always keep track of their neighbours to balance their power and task usage.

The task management plane balances and schedules the sensing tasks given to a specific region (Alkydiz, 2002) (Al-Obaisat, 2007).



## **2.4. Commercial wireless sensor prototypes**

Many companies are realizing the huge potential that wireless sensor devices possess and are spending a lot of time and effort in addressing issues and developing solutions for these devices. As a result, a number of commercial platforms shown in Table 2.4 have been manufactured by Crossbow Technologies, Ember, Microstrain, and Sensametrics (Lynch and Loh, 2006).

Development of a web-based interface for a wireless sensor network monitoring system

PLATFORM	UC BERKELEY CROSSBOW WEC(1999) [14]	UC BERKELEY CROSSBOW RENE(2000) [14]	UC BERKELEY CROSSBOW MICA(2002) [14]	UC BERKELEY CROSSBOW MICA2(2003) [14]	INTEL IMOTE KLING (2003)	MICROSTRAIN GALBREATH ET AL (2003) [16]	ROCKWELL AGRE ET AL (1999)
<b>DATA ACQUISITION SPECIFICATIONS</b>							
A/D Channels	8	8	8	8	No A/D	8	4
Sample rate	1 kHz	1 kHz	1 kHz	1 kHz	No A/D	1.7 kHz (one channel)	400hz
A/D resolution	10-bit	10-bit	10-bit	10-bit	No A/D	12-bit	20-bit
<b>EMBEDDED COMPUTING SPECIFICATIONS</b>							
processor	Atmel AT90LS8535	Atmel Atmega163L	Atmel Atmega103L	Atmel Atmega128L	Zeevi ARM7TDMI	Microchip PIC16F877	Intel StrongARM 1100
Bus size	8-bit	8-bit	8-bit	8-bit	32-bit	8-bit	32-bit
Clock speed	4 MHz	4 MHz	4 MHz	7.383 MHz	12 MHz		133 MHz
Program memory	8 kB	16 kB	128 kB	128 kB	64 kB		1 MB
Data memory	32kB	32kB	512kB	512kB	512kB	2MB	128kB
<b>WIRELESS CHANNEL SPECIFICATIONS</b>							
radio	TR1000	TR1000	TR1000	Chipcon CC1000	Wireless BT Zeevo	RF Monolithics DR3000-1	Conexant RDSSS9M
Frequency band	868/916 MHz	868/916 MHz	868/916 MHz	315, 433 or 868/916 MHz	2.4 GHz	916.5 MHz	916 MHz
Wireless standard	n/a	n/a	ISM	ISM	IEEE 802.15.1	n/a	n/a
Spread Spectrum	No	No	No	Yes (software)	Yes		Yes
Outdoor range				1000, 500 or 328 feet			
Enclosed range				40 meters		30m	100m
Data rate	10 kbps	10 kbps	40 kbps	38.4 kbps	600 kbps	75 kbps	100 kbps
<b>FINAL ASSEMBLED UNIT ATTRIBUTES</b>							
Dimensions	2.5 * 2.5 *1.3 cm			6*3*1cm			7.3 *7.3 *8.9cm
Power	575mAh	2850mAh	2850mAh	1000mAh			
Power source	Coin cell	Battery (3V)	Battery (3V)	Coin cell	Battery(3V)	Battery (3.6V)	Battery (two 9V)

**Table 2.4 Characteristics of commercial wireless sensor prototypes**

## **2.5. Advantages of using wireless sensor networks**

Wireless sensor networks are unique and promising in terms of research and economic potential because of their ability to be deployed in very large scales without complex preplanning, architectural engineering or physical barriers that wired systems faced in the past (Nordman, 2004). They are active in nature; hence they can react quickly if there are environmental changes or a refocus in their task because of their self-organising and self-maintaining capabilities (Suramanian, 2005). In-situ retasking allows the scientists to refocus their observations by issuing commands to the nodes via the wireless link (Mainwaring, 2002). Deployed sensor devices are capable of discovering neighbours and communication topologies and this removes the burden of statically configuring these devices before field deployment (Suramanian, 2005).

While a single sensor node may present limited capabilities, grouping together thousands of such devices offers radical new technological possibilities. Unlike existing, conventional wireless devices, these nodes do not need to communicate directly with the sink node but can be equally effective talking to local peers which then relay any data towards the sink node. This provides lots of advantages including compensating for node failure, easy scalability supporting dynamic architecture and less failure scenarios (Stavrou, 2006).

## **2.6. Disadvantages of using wireless sensor networks**

The physical constraints that exist in individual sensor nodes e.g. power and memory may impact the up-time or paralyze a network if node failures occur (Kay et al, 2004). The error-prone wireless channel which connects the sensor nodes is susceptible to packet losses, but for the system to scale, the fraction of delivered packets must be representative of the original pool (Cao, 2006) (Gumbo et al, 2007).

The absence of a standard architecture or protocol suite for the manufacturing of sensor devices allows vendors to develop individual architectures which make it difficult for cross-vendor compatibility of devices (Suramanian, 2005).

## Chapter 3

### System requirements

*An analysis of the system requirements helps understand the components and the nature of a project being conducted. It establishes the features and services that the wireless monitoring system should provide and the constraints under which it must operate.*

#### 3.1. Introduction

The characteristics of wireless sensor monitoring system mainly depend on:

- **The layout of the structure that is being monitored.**

The inherent lineal characteristic of power overhead transmission lines drives the overall topology of the wireless sensor network which is used to monitor it. The highly asymmetrical many-to-one scenario created by this topology is characterised by heavier traffic loads occurring in the nodes closer to the sink (Cheng, 2004).

## Development of a web-based interface for a wireless sensor network monitoring system

Wireless sensor nodes communicate with the control centre via their adjacent nodes, which relay the packets using a multi hopping scheme (Cheng, 2004). Multihopping is defined as the retransmission of data by intermediate wireless sensors so that data can arrive at the base station.

- **The environmental conditions that affect the operation of the structure.**

Transmission lines are subject to loads imposed by the environment such as wind, ice, snow, earthquakes and flooding as well as human hazards such as fires, accidents and acts of terrorism. The Southern African region, characterised mainly by savannah climate and vegetation, has its major transmission line outages caused by fires (Eskom, 2007).

Due to these loads, a failure in a transmission line may occur suddenly, resulting in the instantaneous instability, rupture or complete separation of the structure else they may occur progressively, resulting in loss of strength in the structure that eventually leads to damage after long periods of time (CSIR, 2007).

- **The variable parameters that need to be monitored in the structure.**

The parameters that may need to be monitored to improve diagnostic capabilities in transmission line systems include voltage, moisture, temperature rises, electric fields and magnetic fields (Nordman, 2004). Any changes in these parameters outside particular thresholds require early detection and reporting to minimize structural damages (Kim, 2005).

- **The battery lifetime requirement from the sensor nodes.**

The traffic load and the corresponding power consumption in this asymmetric network is location dependent hence the up-time of a network is dependent on the nodes closer to the base station with heavier traffic load (Cheng, 2004). The depletion in power nodes may paralyse an entire network. Therefore, the wireless sensor monitoring system should function sparingly in order to maximum battery life.

- **The frequency in which data should be collected.**

The timeliness and efficiency in delivery of a monitoring system is improved by limiting the demand on a communication channel (Seth, 2004). The interest disseminated into the network determines the frequency in which data is collected. It is important in sensor networks whereby the user sends their interest about a certain attribute to a sensor node, a subset of nodes or the whole network.

Data can be collected from the network using an aggregation of event-driven and continuous algorithms. Event-driven algorithms are executed when specific monitored conditions occur, such as parameters exceeding specific thresholds in transmission lines. The disadvantage of the exclusive use of event-driven algorithms is that the parameter values at the readings where no event occurs are unknown; therefore the incorporation of a low frequency continuous algorithm helps alleviate this problem by sending all sensor readings to the base station once in a while.

## **3.2. Functional requirements**

Functional requirements in a software engineering context describe the list of specific tasks or behaviours which a product must perform. The primary requirement in this project was creating a web-based tool that is capable of organizing and displaying collected sensor data.

Other functional requirements include:

- Arranging sensor nodes in a linear topology and localizing their positions to a fixed dataset,
- The ability to obtain readings from nodes in the wireless sensor network,
- Identifying overhead transmission line parameters that can be monitored by the available sensor nodes,
- Communication of measured readings to the network control centre for access using a web-based visualization tool,
- Verification using experiment to ascertain the effectiveness and usefulness of the prototype developed.

### 3.3. Non functional requirements

Non functional requirements describe the limitations or constraints in the implementation of a product. The non functional requirements that needed be investigated in this project include:

- **Accessibility.**

The distributed nature of transmission lines motivates the need to develop a web-based interface to deliver information to users. Such interfaces are accessible over the Internet, from any location without the need of installing specialized software to view data, and are compatible with the majority of web browsers available.

- **Scalability.**

The use of a web-based tool to visualize the activity of the wireless sensor networks was in the attempt to improve the scalability of the number of users who can access the system.

Typical long distance transmission lines would require a network of thousands of individual sensor nodes. It would also be an interesting challenge to see how a linear wireless sensor network scales while delivering a fraction of sensor readings that is representative of the original pool.

- **Reliable data propagation.**

A reliable wireless sensor monitoring system delivers readings even under abnormal conditions to the control centre, reducing damages and power supply disruptions. Thus, it was necessary to test the performance of the network under different environmental conditions, noting any propagation delays and overheads.

### **3.4. Wireless sensor system components**

The complete system for the management of power transmission lines consists of:

- The data acquisition tools i.e. the mote environment (further described in Chapter 4)
- The wireless sensor network application (further described in Chapter 5).

### **3.5. Wireless sensor data acquisition system**

The tools required for data acquisition for structural assessment are divided into hardware and software categories and are described Table 3.1.



## Development of a web-based interface for a wireless sensor network monitoring system

REQUIREMENT	VERSION	DESCRIPTION
<b>HARDWARE REQUIREMENTS</b>		
Wireless sensor nodes	Crossbow Mica2	Data collecting devices (Crossbow, 2007).
Sink node	Crossbow Mica2	Sensor network control centre (Crossbow, 2007).
Programming board	MIB1500	A programming board was used to upload and install software into the wireless sensor nodes before field deployment (Crossbow, 2007).
Antenna	916 MHz frequency	An antenna wireless communication between the wireless sensor nodes. The frequency utilized by an antenna is important as it determines the transmission range of the sensor node (Crossbow, 2007).
<b>SOFTWARE REQUIREMENTS</b>		
TinyOS	tinyos-2.0.2-2	The open source operating system designed for wireless sensor networks (TinyOS, 2007).
TOSSIM		A TinyOS, wireless sensor node simulator (TinyOS, 2007).
Cygwin	1.5.24-2	Linux emulation layer that provides Linux API functionality on the Windows platform. It gave the user a shell and UNIX tools which the TinyOS environment uses (Cygwin, 2007) (TinyOS, 2007).
nesc	nesc-1.2.8a-1	The programming language used to develop software on TinyOS (TinyOS, 2007).
TinyDB	tinydb-1.1.3	The query processing system for extracting information from the sensor network (Madden, 2003).
Java	jdk1.4.0_06	Java was used for communication between the sink node and a computer server (Sun Microsystems, 2007).

**Figure 3.1 Data acquisition tools**

### 3.6. Wireless sensor network application

For the development of the wireless sensor application which contains the post processing algorithms to process the data received from the network, the tools described in Table 3.2 are required.

SOFTWARE USED	VERSION	DESCRIPTION
Apache Tomcat	5.5	Servlet container for executing and testing codes (Apache Foundation, 2007).
Netbeans IDE	5.5	Integrated development environment (IDE) for writing and testing code (Netbeans, 2007).
Java 2 Platform Standard Edition (J2SE)	1.5 (JDBC 3)	Provided Java APIs, the Java Virtual Machine (JVM) and the components necessary to run Java programs. It contains Java Enterprise Edition (J2EE) for development of Enterprise JavaBeans, Java servlets, Java Server Pages and XML technology (Sun Microsystems, 2007).
JFreeChart	1.0.5	Java class library for generating charts (JFreeChart, 2007).
PostgreSQL	8.0	Open source, object-relational database management system (PostgreSQL, 2007).
pgAdmin III	1.8.0	Open source administration and development tool for PostgreSQL (pgAdmin, 2007).
Cygwin		Linux emulation layer were PostgreSQL database resides (Postgresql, 2007)(Cygwin, 2007).

**Figure 3.2 Wireless sensor application tools**

## Chapter 4

### Data acquisition system

*This chapter describes the Mica data acquisition system used to issue commands and collect data from the wireless sensor network. Its characteristics, such as hardware components, operating system and applications are also described to detail.*

#### 4.1. Introduction

A typical wireless data acquisition system consists of a computer, a base station and wireless sensor node platforms. For our work, Mica2 sensor nodes, shown in Figure 4.1, were chosen for conducting the experiments. The motivation behind choosing this type of sensor node is that it is based on an open source platform, which allows for the development of programs for it (Gumbo, 2007). The Mica2 platform (mote) was developed at the University of California (Berkeley) and commercialized by Crossbow with both its hardware and software design available to the public. Crossbow motes can be programmed conventionally over a serial cable, or re-programmed over a wireless link. Some of the application scenarios already deployed using Crossbow motes include environmental, health and structural monitoring (Crossbow, 2006) (Carter, 2006).

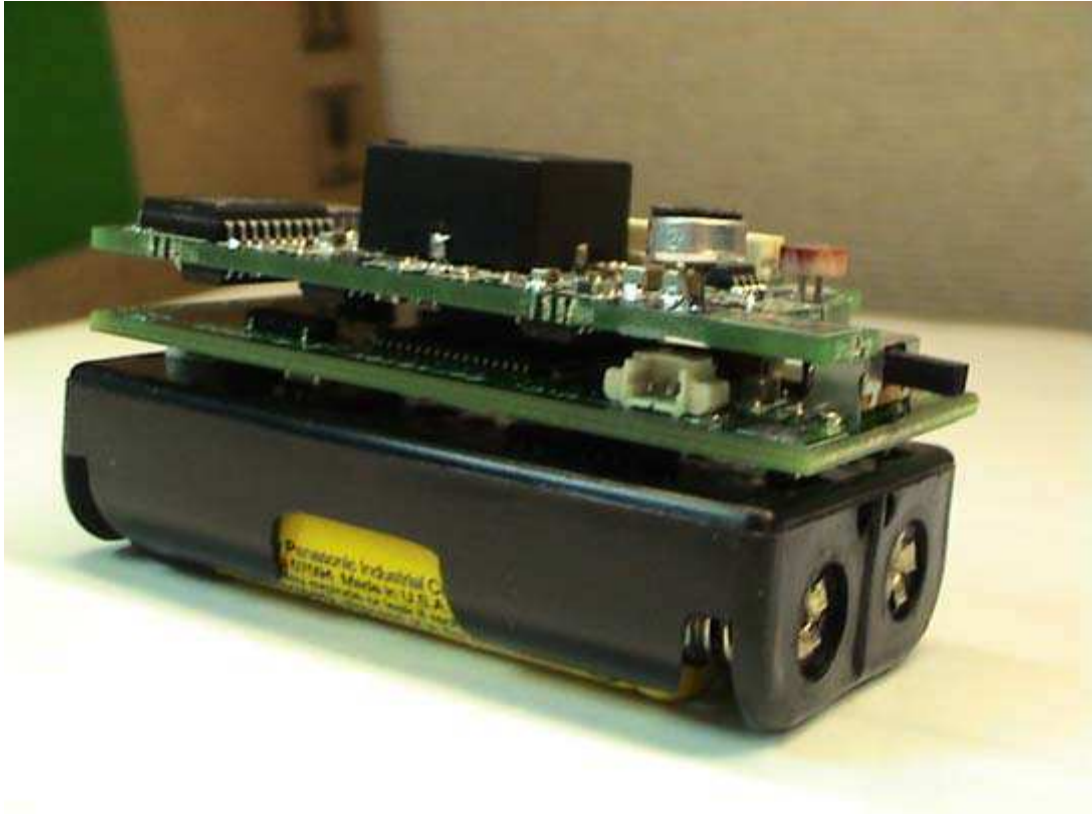


Figure 4.1 A Mica2 node

## 4.2. Hardware platform architecture for a Mica2 node

A typical wireless sensor node consists of four sub-systems as shown in Figure 4.2 (Alkydiz, 2002).

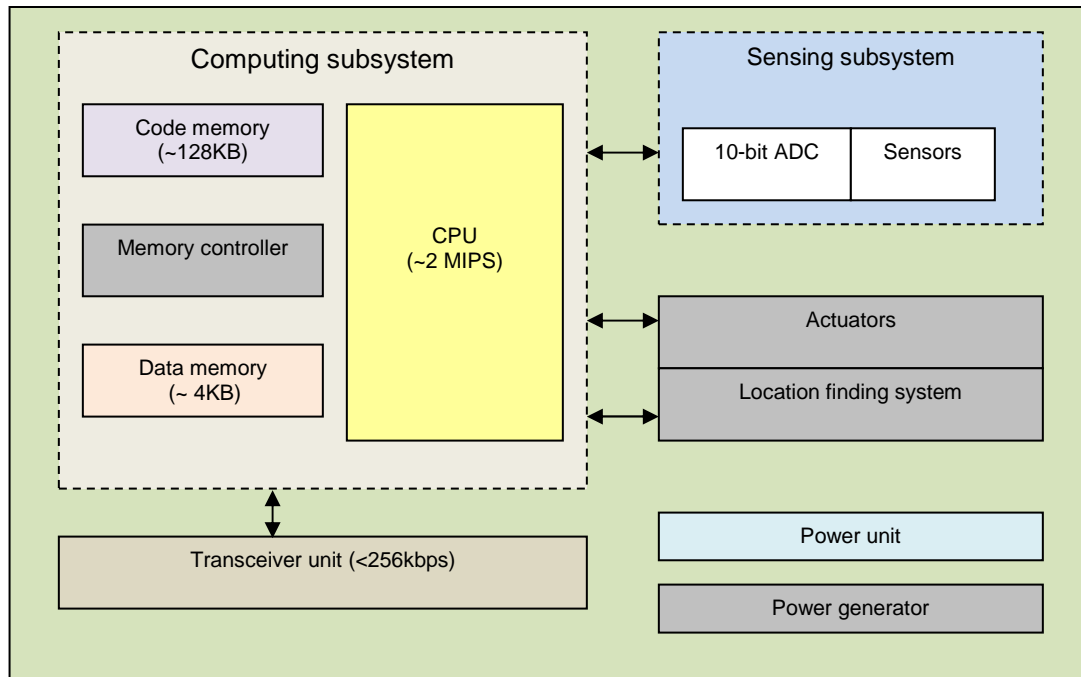


Figure 4.2 Hardware architecture for a sensor node

### 4.2.1. Computing subsystem

The computing subsystem comprises mainly of the micro controller unit (MCU) which is responsible for the control of sensors and execution of communication protocols (Bharathidasan and Ponduru, 2005). Its operation is strongly associated with memory. It provides the ability for a node to collaborate with others in order to fulfill assigned sensing tasks. The MCU operates under on idle and off operating modes, each with different power consumption levels shown in Table 4.1 (Alkydiz, 2002).

### **4.2.2. Sensing subsystem**

The sensing unit consists of sensors and analog to digital converters (ADC). A sensor is defined as “*a self-containing electronic device whose circuitry measures and senses changes in voltage, moisture, temperature values, electric field or magnetic field values*”. These are parameters that may need to be monitored in overhead power line systems (Nordman, 2004). The signal conditioning circuitry within sensors converts this physical parameter of the environment into an electrical analog signal. The ADC converts this electric signal into a digital one and inputs it to the computing system, which communicates the signal to neighbouring nodes via a directional antenna.

Sensors are deployed inside a phenomenon or very close to it to reveal its properties or events in its vicinity (Bramwell, 2006) (Kuuorilehto, 2005). Data extracted by the on-board sensors is propagated towards the control centre for further processing, since the sensor nodes deployed in the field cannot operate in complete isolation (Alkydiz, 2002).

### **4.2.3. Power supply subsystem**

A typical power subsystem consists of a 3 volt battery which supplies energy to the node. An alternative to the reliance on batteries, whose power may be short-lived for some applications, is the use of renewable energy sources such as solar or photovoltaic cells (Bharathidasan, 2005). Applications developed using wireless sensor devices should allow them to remain in sleep mode until the external condition warrants their re-awakening in order to lower power consumption (Hill, 2003). The power consumption of different node activities is shown in Table 4.1 (Crossbow, 2007).

Activity	Consumption	Activity	Consumption
Microcontroller Active	8.0 mA	LEDs (each)	2.2 mA
Microcontroller Idle	3.2 mA	Sensorboard	0.7 mA
Microcontroller sleeping	< 15 $\mu$ A	Radio receiving (Rx)	7.03 mA
Power-down from Active to Idle	103 $\mu$ A	Radio transmitting (power = 00)	3.72 $\mu$ A
Power-save from Idle to Off	110 $\mu$ A	Radio transmitting (power = 03)	5.37 $\mu$ A
Standby	216 $\mu$ A	Radio transmitting (power = 09)	7.05 $\mu$ A
Ext. Standby	223 $\mu$ A	Radio transmitting (power = 60)	11.57 $\mu$ A
Internal Oscillator	0.93 mA	Radio transmitting (power = FF)	21.48 mA

Table 4.1 Mica2 power consumption

#### 4.2.4. Communication subsystem

The antenna element is one of the key components in any wireless sensor system because it is the interface between the radio frequency channel and the system's hardware (Meguerdichan, 2001) (Hill, 2003). Antennas are susceptible to a variety of propagation impairments such as interference, reflections, scattering and shadowing which influence the wireless sensor application performance and cost (Hafez, 2005).

A fundamental choice in the design and development of wireless sensor applications is the selection of an operating frequency that complies with government and wireless standards. Current wireless sensor system frequencies are 315MHz, 433 MHz, 868 MHz (Europe), 915 MHz (North America), and the 2.45-GHz Industrial-Scientific-Medical (ISM) band. A

comparison of some of these frequencies and their subsequent performances are shown in Table 4.2 (Crossbow, 2007).

PARAMETER	MPR410CA*	MPR400CA*	TPR2400CA*
Frequency	433 MHz	868/916 MHz	2400 to 2483.5 MHz
RF transmit power	-20 to +10 dBm	-20 to +5 dBm	-24 to 0 dBm
Receiver sensitivity	-101 dBm	-98 dBm	-94 dBm
Current draw	25 mA @ max power	27 mA @ max power	23 mA @ max power
Data rate	38.4 kbaud	38.4 kbaud	250 kb/s
Battery	3 V	3 V	3 V
Range	1000 ft. line of sight	500 ft. line of sight	328 ft. line of sight

**Table 4.2 Comparison of frequencies and ranges of Mica sensor systems**

The basic concern in the use of these frequencies is the possibility of interference from other prevalent wireless technologies such as WLAN 802.11b/g, WiMAX and Bluetooth. As a result, spread-spectrum technology or frequency-agile techniques need to be implemented on transceivers to mitigate interference from these other wireless technologies. Mica2 nodes that do not incorporate the Zigbee stack prefer frequency-hopping spread spectrum (FHSS) schemes to direct-sequence spread spectrum (DSSS), because the former requires less circuitry to implement and is less energy consuming (Koubaa, 2005).

### 4.3. Architecture of the base station node

A base station or sink node, shown in Figure 4.3, is a personal computer node which links a sensor network to another existing infrastructure where extracted sensed data undergoes post processing.

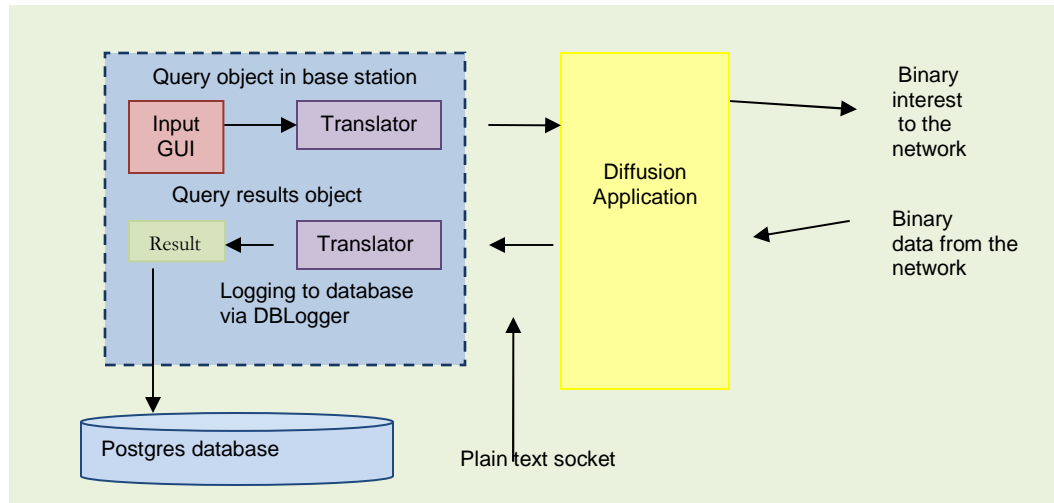




**Figure 4.3 Base station node**

Users communicate with sensor nodes via their serial port on the programming board, allowing them to inject queries or receive results from a radio based sensor network (Akkaya and Younis, 2005) (Crossbow, 2007). Results received from the sensor network can either be displayed on a base station software GUI or written to a PostgreSQL database via the DBLogger implementation layer within the base station as illustrated in Figure 4.4 (Gumbo, 2007).

## Development of a web-based interface for a wireless sensor network monitoring system



**Figure 4.4 Reference model of the architecture of the base station software**

The limitations of using the base station GUI are:

- The sensor nodes are presented as continuous values to the user because this software is incapable of performing any post processing on this sensor data.
- The display of the GUI is limited to the base station computer or any other computer nearby that has this software installed (Gumbo, 2007).

The idea of integrating enhanced processing logic with continuous dynamics in the control of complex systems using base stations leads to the evolution of wireless supervisory control systems. Wireless supervisory control and data acquisition systems (WSCADA) are high-level control schemes that manage large and/or heterogeneous systems such as electric network grids or nuclear power plants. In general, the supervisor implements a decision-making scheme based on the condition of the required application and involves principles such as data acquisition, data analysis, data display, feedback control and data storage in implementation (FreeOnlinePatent, 2004)(Das, 2004). WSCADA eliminates the difficulty of programming nodes individually by providing software packages that allow simultaneous node configurations to occur.

#### 4.4. Comparison of wireless sensor operating systems

A single node is controlled by its operating system or virtual machine (VM). Various operating systems are compatible with wireless sensor nodes, and some of these are shown in Table 4.3 (Kuuorilehto, 2005). TinyOS was developed especially for Mica prototypes, therefore it was naturally chosen for use in this project.

PROPOSAL	TEST ENVIRONMENT	SIMULATION AND TESTING TOOLS	PROTOTYPE PLATFORM	RESULT ACCURARY
TinyOS	Prototype	TOSSIM	Motes	Accurate
BerthaOS	Prototype	None	Pushpin	None
EYES OS	None	None	None	None
MOS	Prototype	PC emulator XMOS	Nymph	Moderate
BTnodes	Prototype	None	Micro-size BT nodes	Moderate

**Table 4.3 Comparison of wireless sensor network operating systems**

## **4.5. TinyOS and the development environment**

### **4.5.1. Tiny Microthreading operating system**

TinyOS is a mature operating system that was developed at the University of California (Berkeley) and has become the de-facto standard for wireless sensor networks (Hill, 2003). It was written in the nesC programming language, with a component-based architecture ideal for constrained devices with low powered processors and memory in the order of a few kilobytes (currently 256 bytes RAM and 4kB ROM) (TinyOS, 2007). TinyOS permits the assembling of application-specific systems using its component library which includes network protocols, sensor drivers, data acquisition tools and distributed services .A complete TinyOS archive consists of:

- The TinyOS core and runtime routines,
- A nesC compiler,
- TOSSIM, a discrete event simulator that captures TinyOS behaviour,
- TinyViz, a GUI for TOSSIM (TinyOS, 2007).

TinyOS supports autonomous routing of packets based on ad hoc, peer-to-peer connections between sensor nodes. In such networks, devices do not maintain network topology information. Identification of other nodes occurs only when routing of data is required. This allows shifts in network topologies in the case of movement in devices, power shortages, or shifting waves due to interference (Baghdasaryan, 2005).

### **4.5.2. SerialForwarder**

SerialForwarder is part of the TinyOS core programs based at the control centre which provides tools for communication with sensor networks. It launches a single process to query the sensor network and any packets received from the network are stored in the temporal database. It is able to read and relay data packets sent between the serial port on a

## Development of a web-based interface for a wireless sensor network monitoring system

base station and the wireless network, as shown in Figure 4.5. A SerialForwarder program listens and forwards TinyOS messages on a given TCP port (9001 is the default) (TinyOS, 2003).

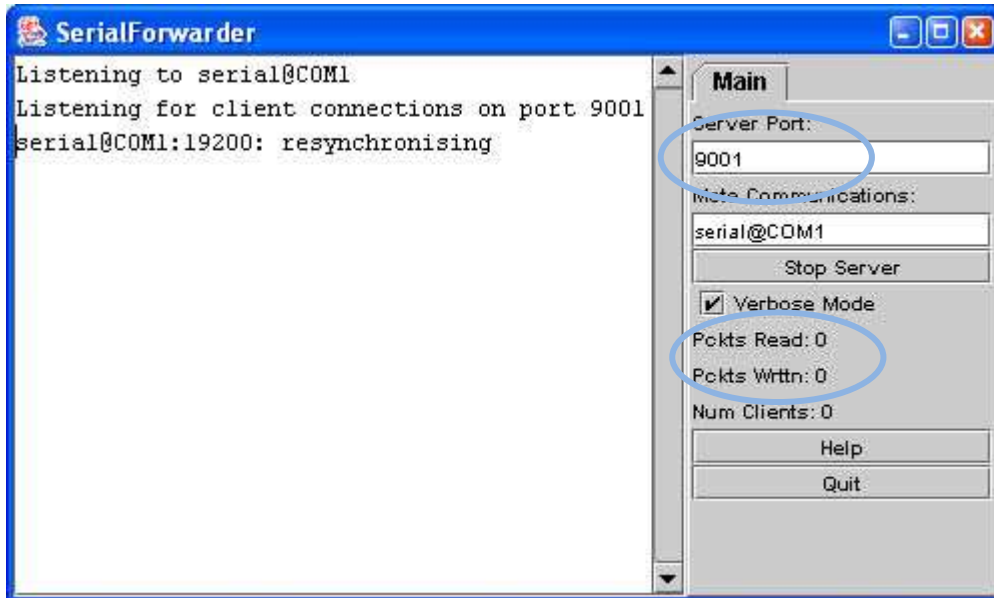


Figure 4.5 SerialForwarder program

### 4.5.3. TinyOS Messages

TinyOS typically employs a maximum of 36-byte long radio messages (Thom, 2006). Seven bytes are reserved by the operating system to store the destination address, message type, group id, data length and the cyclic redundancy check (CRC), and the rest are used for application-specific data, as illustrated in Figure 4.6, Table 4.4 and Table 4.5 (Thom, 2005)(Carter, 2006).

```
typedef struct TinyOS_Message {
    // The following fields are transmitted/received on the radio.
    // The units for the fields are in bits.
    uint16_t address;           // destination address
    uint8_t message type;      // Active Message handler ID
    uint8_t group id;          // Receive group ID
    uint8_t data length;       // Length of the data section
    int8_t payload data[TOS_HEADER_DATA_LENGTH]; // data packet up to 29 bytes
    uint16_t cyclic redundancy check; // CRC checksum

    // The following fields are not actually transmitted or received
    // on the radio! They are used for internal accounting only.
    // The reason they are in this structure is that the AM interface
    // requires them to be part of the TOS_Msg that is passed to
    // send/receive operations.
    uint16_t strength;
    uint8_t acknowledgement;
    uint16_t time;
    uint8_t sendSecurityMode;
    uint8_t receiveSecurityMode;
} TOS_Msg;
```

**Figure 4.6** TinyOS message components

ADDRESS		MESSAGE TYPE	GROUP ID	DATA LENGTH	DATA	CRC	
0	1	2	3	4	5...n-2	n-1	n

**Table 4.4 TinyOS message format**

BYTE NUMBER	FIELD	DESCRIPTION
0 -1	Message address	One of three possible values: <ul style="list-style-type: none"> <li>• Broadcast address (0×FFFF) – message to all nodes</li> <li>• UART Address (0×007e) – message from a node to the gateway serial port. All incoming nodes will have this address</li> <li>• Node address<sup>1</sup> – the unique ID of a node assigned when it is programmed to receive messages.</li> </ul>
2	Message Type	Active message (AM) unique identifier for the type of message it is. Typically each application will have its own message type. Examples include: <ul style="list-style-type: none"> <li>• AMTYPE_XUART = 0×00</li> <li>• AMTYPE_MHOP_DEBUG = 0×03</li> <li>• AMTYPE_SURGE_MSG = 0×11</li> <li>• AMTYPE_XSENSOR = 0×32</li> <li>• AMTYPE_XMULTIHOP = 0×33</li> <li>• AMTYPE_MHOP_MSG = 0×FA</li> <li>• AMTYPE_CNTTOLEDSANDBRFM = 0×04 (Shown in Fig 4.7)</li> </ul>
3	Group ID	Unique identifier for a group of motes participating in the network. The default value is 125 (0 × 7d). Only the motes with the same group id will talk to each other.
4	Data Length	The length in bytes of the data payload. This does not include the CRC of the frame SYNC bytes.
5..n-2	Payload Data	The actual message content. The data resides at byte 5 through byte 5 plus the length of the data. The data will be specified to the message type.
n-1,n	CRC	The two byte code that ensures integrity of the message. The CRC includes the Packet Type plus the entire unescaped TinyOS message.

**Table 4.5 TinyOS message field descriptions**

<sup>1</sup> A mote is assigned an address when it is programmed with an application through the command :  
make [platform] install.<addr> where *addr* is the address to the node.

### 4.5.4. Multihop messages

A linearly aligned wireless sensor network utilizes multihop messaging for communication between the nodes and the base station. The layout, format and field descriptions of a MultiHop message are illustrated in Figure 4.7, Table 4.6 and Table 4.7 (Thom, 2005).

```
typedef struct MultihopMsg {
// The units for the fields are in bits.
uint16_t sourceaddress;
uint16_t originaddress;
int16_t sequencenumber;
uint8_t hopcount;
uint8_t payload data[(TinyOSHeader_DATA_LENGTH - 7)];
}
```

**Figure 4.7 Multihop message components**

SOURCE ADDRESS		ORIGIN ADDRESS		SEQUENCE NUMBER		HOP COUNT	APPLICATION DATA
0	1	2	3	6	5	6	7...n

**Table 4.6 Multihop message format**

BYTE NUMBER	FIELD	DESCRIPTION
0 – 1	Source Address	The address of the forwarding node
2 - 3	Origin address	The address of the node that originated the packet
4 – 5	Sequence number	Used for determining route and calculating missed packets
6	Hop count	Used for calculating a route. Number of nodes transversed
7 ...n	Application data	The specific application data

**Table 4.7 Multihop message component description**



### **4.5.5. Wireless sensor network simulators**

Before deploying wireless sensor applications and systems in the real world, it is necessary to first test their feasibility in a controlled, simulated environment. Simulation is a standard tool for the evaluation of distributed and communication systems like sensor networks (Landsiedel, 2005). An ideal sensor simulator should provide sensor node hardware components (such as radio stacks and sensors), include the ability for packet level network abstraction and should be able to model typical sensor network behaviour. Current simulation efforts focus on the protocol and algorithmic issues affecting networks, but neglect time dependent issues (such as amount of time for sensor data to arrive at the base station) which are necessary to correctly interpret sensor behaviour (Landsiedel, 2005).

### **4.5.6. TOSSIM**

TinyOS simulator (TOSSIM) is a distributed simulation tool for wireless sensor networks. It simulates directly from TinyOS code, and captures the behaviour and interactions of thousands of nodes. The problem with simulators is that assumptions and simplifications of packet losses have to be made in their execution; this inevitably leads to the gap between reality and the simulated virtual world (Beutel, 2005). TOSSIM solves this problem by providing two radio stacks, a simple and a lossy model in the attempt to simulate events. The simple model assumed that all radio transmissions are perfect, with no loss or corruption of data due to interference while the lossy model allows error to occur in the wireless link (Lewis, 2003).

TinyViz is the TOSSIM visualization tool which allows the control and analysis of the virtual wireless sensor network behaviour (Lewis, 2003). Its interactive graphic user interface (GUI) allows developers to view debug messages and monitor network traffic giving them the ability to identify TinyOS network stack problems which go unnoticed in real world deployments of nodes (Crossbow, 2003).

### 4.5.7. Sensor network query processing

TinyOS sensors incorporate query processing technologies, described in Table 4.8 to extract information from wireless sensor networks (Kuuorilehto, 2005). These query processing techniques eliminate the need to write low-level code for data-driven applications. (Madden, 2003).

PROPOSAL	TEST ENVIRONMENT	SIMULATION AND TESTING TOOLS	PROTOTYPE PLATFORM	RESULT ACCURACY	PUBLISHED RESULTS
<b>Middleware architectures</b>					
MiLAN	None	None	None	None	None
SINA	Simulations	GloMoSim	None	Poor	SINA networking overhead, application performance
TinyDB	Simulations, prototype	Custom environment	TinyOS mote	Accurate	Query routing performance in simulations, sample accuracy and sampling frequency in prototypes
Cougar	None	None	None	None	None
LIME	JVM	None	PC	Poor	Approximations about Java code size
MARE	JVM	None	PDA	Poor	Service discovery performance
RCSM	Prototype	None	PDA with custom hardware	RCSM poor, RKS accurate	RCSM memory consumption, RKS size, communication, energy consumption

**Table 4.8 Comparison of sensor network database management systems**

#### **4.5.8. TinyDB**

Tiny Database (TinyDB) is a query processing technology for extracting information from a network of sensor devices. It is an open source application which has been successfully tested on TinyOS motes, thus it becomes the natural choice in the development of the project (Madden, 2003). TinyDB provides the facility to develop and deploy standalone data-driven applications from a sink to the network of sensor nodes using Structured Query Language (SQL) and a Java Application Programming Interface (API) (Mayer et al, 2003) (Gumbo, 2007). TinyDB uses a hybrid approach to broadcast SQL queries into a wireless sensor network. This means, any nodes that do not receive a query when it is initially flooded will eventually become aware of it because nodes periodically exchange all known TinyDB queries in the network (Madden, 2005).

## Chapter 5

### The wireless sensor network application

*The system described in this chapter aims at acquiring, storing and displaying data collected from a wireless sensor network. Open source tools have been used in the development and implementation of this system.*

#### 5.1. Introduction

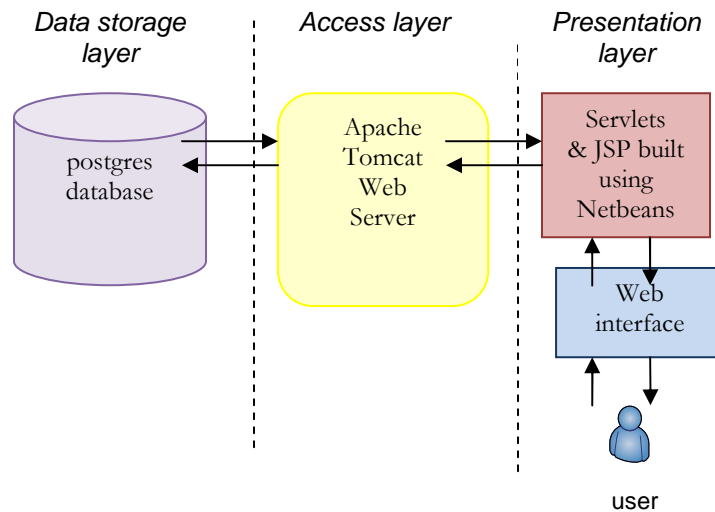
Sensor data includes raw data produced continuously by the sensor networks and data on the design of the monitored structure that is collected once and archived. The continuous, time series values collected by the sensors in the network can be overwhelming, making structural assessment a difficult and time consuming task that can be accomplished only by highly-trained engineers. A wireless sensor network requires an application which provides advanced post processing to improve interpretation of sensor data (Gumbo, 2007).

The sensor network, whose readings are stored in the centralized temporal database, is treated as a distributed network cluster. Each reading in this database is saved with the

## Development of a web-based interface for a wireless sensor network monitoring system

identifier of the node from which it originated. This is helpful when executing the post processing algorithm on the sensor readings in the database, because each query issued will receive sensor data results that are accompanied by these node identifiers.

The algorithm to improve the interpretation of sensor data is embedded in a wireless sensor application that consists of presentation, data access and storage systems as shown in Figure 5.1 (Gumbo, 2007).



**Figure 5.1 Reference model for the wireless sensor network application**

This wireless sensor network application is a multi-tier Java 2 Enterprise Edition (J2EE) architecture; hence it can run on any operating system that supports the Java runtime environment. Open source platforms have been used in its development because they require no special licensing to use for research or commercial purposes, they adhere to standards, and are easy to deploy on any system (Gumbo, 2007).

## 5.2. Presentation layer

The presentation layer consists of a static webpage and dynamic data requested by the user from the database. It provides a visual interface built using servlets, Java Server Pages and JFreeChart to present data from the sensor network and enhance data comprehension.

### 5.2.1. Netbeans Integrated Development Environment (IDE)

The Netbeans IDE was created by Sun Microsystems specifically to perform tasks on Java applications. Its interface is shown in Figure 5.2. It has an embedded Tomcat server, which allows the deployment of an application. It requires at least 256 MB RAM and processor speed greater than 700MHz when creating JSP and servlet technology based web applications.

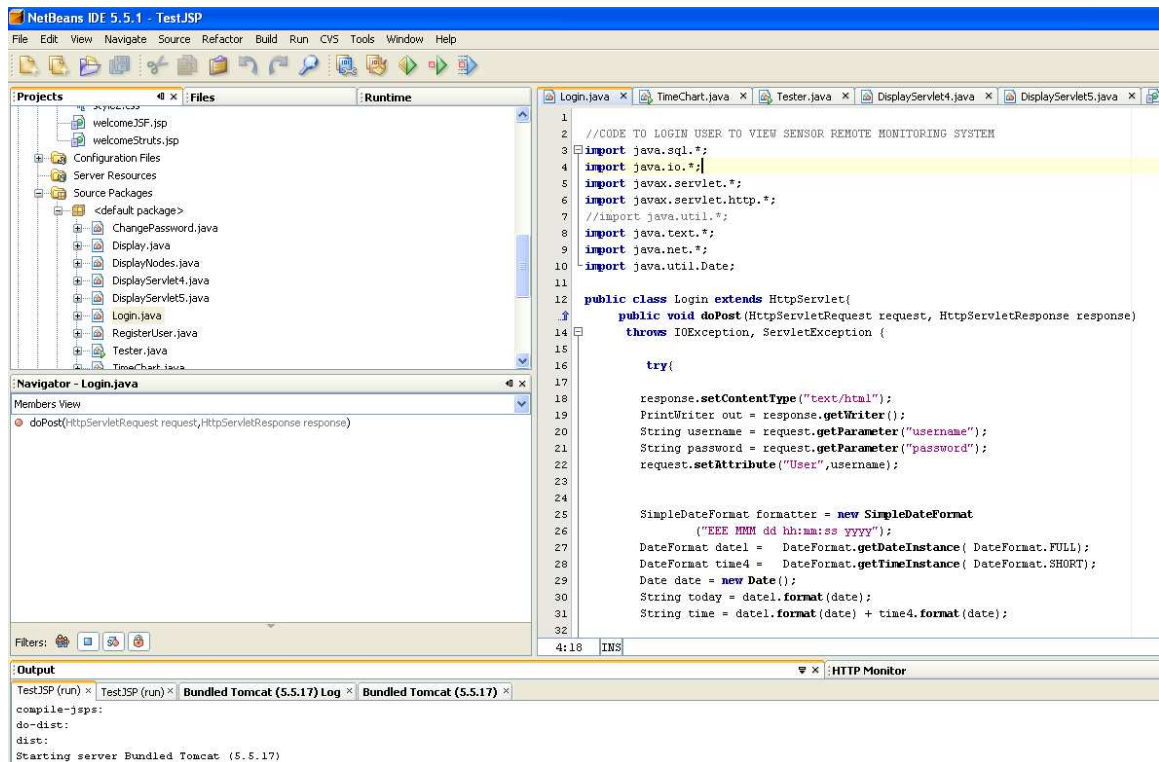


Figure 5.2 The Netbeans IDE interface

## Development of a web-based interface for a wireless sensor network monitoring system

The advantages of using the Netbeans IDE include:

- **Modular support.** Netbeans platforms are able to handle versioning and dependency management; hence applications developed using previous versions of Netbeans are automatically handled by more recent versions of Netbeans.
- **Flexible User Interfaces.** Menus, toolbars and keyboard shortcuts can be added, deleted and changed through an XML file called layer.xml within the application.
- **It is an open source API.**

The disadvantages of using the Netbeans IDE include:

- **API changes.** When Netbeans API versions change, the previous version become deprecated, or disappears, sometimes with no obvious replacement. (Netbeans, 2007)

### 5.2.2. Servlets and Java Server Pages

A portion of the presentation layer facility was developed using Java Server Pages (JSP) and servlets, which are built and tested using Netbeans IDE (Netbeans, 2007). The use of JSPs makes the system more flexible and easily customizable to suit the specific needs of the user while keeping the interface intact (Sun Microsystems, 2007).

Access to the webpage and its information is controlled by a password authentication system because of the sensitivity of information collected by the wireless sensor network. The web-based interface was chosen as a tool to deliver information because web browsers are familiar to computer users and no additional software must be installed in the user's computer, since web browsers are a standard application in most operating systems.

### 5.2.3. Alert indicators

Simplicity is required in this interface for efficient retrieval and visualization of data that is relevant to assessing structural states. Thus, a set of simple alert icons were incorporated alongside the data retrieved from the database into the webpage to depict the status of the sensors in the field. The structural alert indicators use a deterministic approach to present data. Three degrees of alerts were implemented into the interface – normal (green), warning (yellow) and critical (red). The criteria for activating an alert should be established by the engineer in accordance with the desired structural performance. In order to achieve this, various threshold limits were set for all the sensor readings, as indicators of the degree of damage to the structures being monitored. This allows information to be read quickly by simply looking at an icon and recognizing a colour. This tool was computed by analysis classes in Java that perform calculations on the recorded data sets using the threshold values to determine the status of the sensors (Gumbo, 2007).

### 5.2.4. JFreeChart

Charts, figures and graphs can enable large datasets such as sensor system data to be understood and analyzed quickly. These convey a significant overview and comparative information about data that is more difficult to understand from textual data. Time history graphs, which provide a tool to visualize changes in the power line structure parameters over time, can easily be integrated into web interfaces. These can be built and deployed using JFreeChart, an open source Java class library, that provides the ability to dynamically generate different types of graphs, plots or images which include pie, bar, line and Gantt charts from application code (JFreeChart, 2007) (Gilbert, 2004). JFreeChart is capable of running in a servlet environment which can be easily plugged into the existing interface framework.

The JFreeChart feature set includes:

- **Developer support.** The widely supported developer community has resulted in a consistent, well documented API to support a wide range of chart types.



## Development of a web-based interface for a wireless sensor network monitoring system

- **Flexible design.** JFreeChart is easily extended to support most applications, with many output types, including swing components, image files and vector graphic file formats (JFreeChart, 2007).

### 5.3. Data access layer

The data access layer provides a container that controls access to data in the repository and manages the retrieval of files over the network.

#### 5.3.1. Apache Tomcat web server

Apache Tomcat is a standalone, developmental web server that is used to support the implementation of servlet and Java Server Pages technologies. It offers a diverse set of open source Java solutions as part of the Apache Software Foundation that encourages a collaborative, consensus-based development process under an open source license (Apache Software Foundation, 2007). Its feature set includes:

- Ability to be integrated with native Windows and UNIX wrappers;
- Enhanced security manager support;
- Scalability and reliability enhancements;
- Performance optimizations and reduces garbage collection;
- A refactored application deployer, with an optional standalone deployer “*allowing validation and compilation of a web application before putting it in production*”;
- A vast amount of installation, configuration and deployment documentation (Apache Foundation, 2007).

### 5.4. Data storage system

The storage layer maintains the actual data files and the underlying database that provides live data to the user interface. The database provides a method of organizing data collected from the sensor network into one place, using a tabular format. This database should have a

double “interface” because it will interact with the sensor nodes to receive data and the wireless sensor network application extracts data from it for display on the web interface. The properties of the wireless link connections between the sensor nodes and the effects of interference and packet loss on the quality of the collected data can be evaluated by the amount of data collected in the database. Any undelivered measurements can be ignored since the main interest lies only with the most recent measurements.

### 5.4.1. PostgreSQL

PostgreSQL is an open source, object-relational database management system, developed at the University of California (Berkeley). It supports the logging of TinyDB queries, hence is able to support a wireless sensor network data acquisition system (PostgreSQL, 2007)(Madden, 2003). PostgreSQL is usually deployed in a Unix environment, therefore, in order to run it on a Windows platform, prior installation of WIN32, Cygwin or MinGW (linux emulators) is required (PostgreSQL, 2007).

The PostgreSQL configurations used in the project are shown in Figure 5.3.

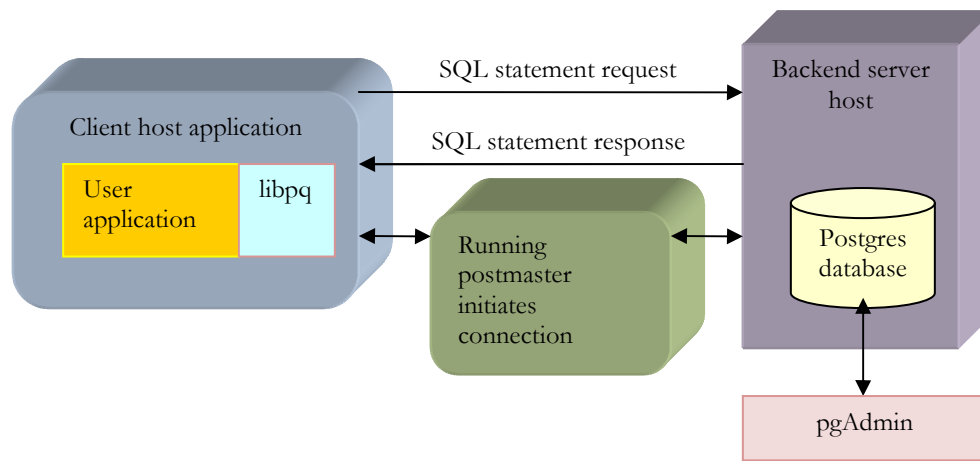
```
#-----  
# CONNECTIONS AND AUTHENTICATION  
#-----  
# - Connection Settings -  
  
listen_addresses = '*'      # what IP interface(s) to listen on;  
                           # defaults to localhost, '*' = any  
port = 5432                 # port number for connections  
max_connections = 100  
    # note: increasing max_connections costs about 500 bytes of shared  
    # memory per connection slot, in addition to costs from shared_buffers  
    # and max_locks_per_transaction.  
#superuser_reserved_connections = 2  
#unix_socket_directory = "  
#unix_socket_group = "  
#unix_socket_permissions = 0777 # octal  
#rendezvous_name = "        # defaults to the computer name
```

Figure 5.3 PostgreSQL configuration file

## Development of a web-based interface for a wireless sensor network monitoring system

A PostgreSQL session is successful if the following processes (illustrated in Figure 5.4) cooperate:

- A postmaster,
- Server processes,
- Client application,
- Libpq processes,
- PgAdmin. (Gumbo, 2007)



**Figure 5.4 Communications between the client host and the database**

In order for a client process to access a database, it needs to connect to a running “postmaster”, which is a multi-user database server, which allows many connections to the database. PostgreSQL is implemented using a ‘process per user’ client-server model, where a client is connected to a server process (“postgres”) via postmaster. A C-based application programmer's interface, libpq, is required to allow the client programs to pass queries to the PostgreSQL backend server and to receive the results of these queries. Many clients are based on the C-language library libpq, but several independent implementations of the protocol exist, such as the Java JDBC driver (PostgreSQL, 2007). PgAdmin is an open source administration and development platform that can be used for PostgreSQL (pgAdmin , 2007).

## Chapter 6

### System design

*This chapter discusses the design of the various aspects of the prototype, i.e. the data acquisition system and the wireless sensor application. The system design presents the function of each system in a manner that can be easily transformed into computer programs.*

#### 6.1. Introduction

The complete transmission line monitoring system can be broken into the wireless sensor node environment, the user application, the database and the SerialForwarder. It is designed using a Model-View-Controller (MVC) architecture which consists of:

- the **model** which is a domain-specific representation of data classes on which the application operates i.e. the mote environment;
- the **view** which renders the **model** into a form suitable for interaction. Views consist of a set of instruction which give visual results and are typically implemented as interfaces;

## Development of a web-based interface for a wireless sensor network monitoring system

- the **controller** which handles and processes user events and instructs the **model** and or the **view** to change accordingly (Cavaness, 2004).

The processes in an MVC architecture operate as follows:

- The user interacts with the **view** or interface in some way (e.g. by click a button);
- The **controller** handles the input event via registered handlers or call backs;
- The **controller** then accesses the **model**, possibly instructing it to perform a certain operation on behalf of the views;
- Results are returned to the **controller** by the model;
- The controller passes these results to the **view** and instructs it to generate content appropriately.

The overall design layout in Figure 6.1 shows the communication between the wireless sensor node environment (data acquisition system) and the user application occur via the SerialForwarder.

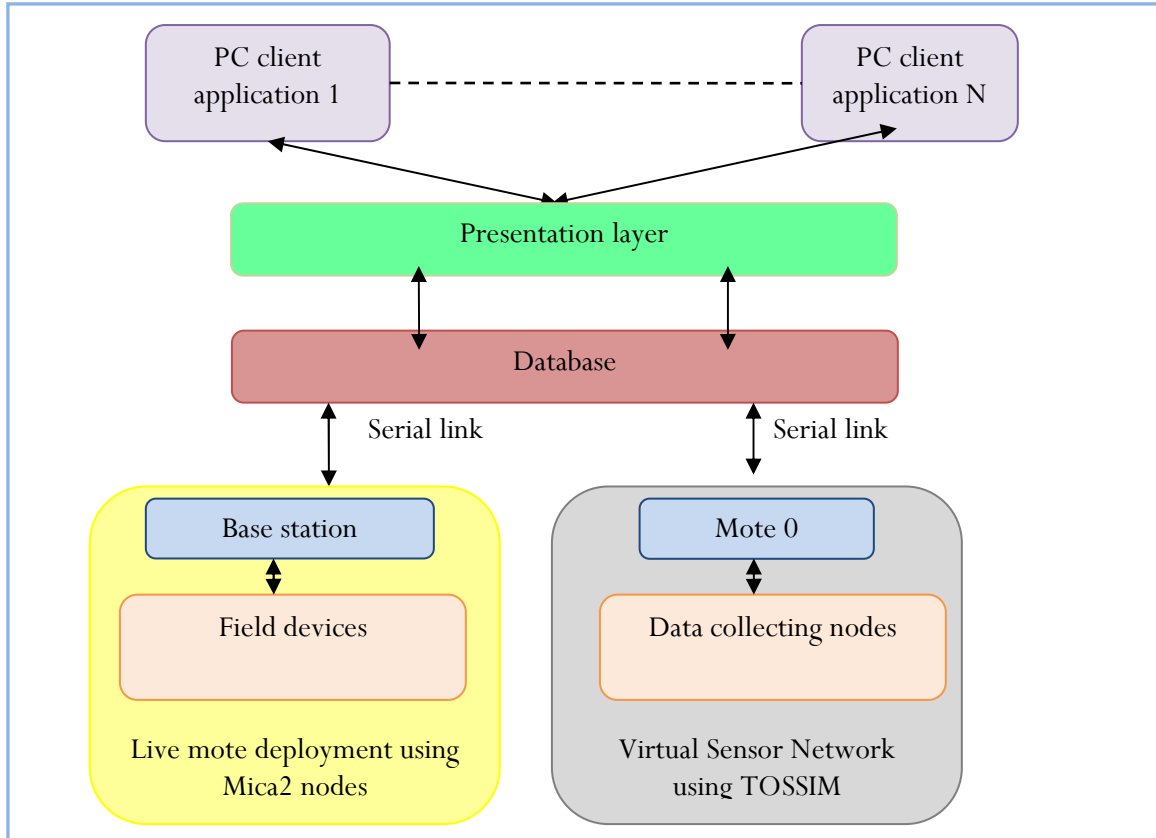


Figure 6.1 Overall wireless sensor network design architecture

## 6.2. Wireless sensor design architecture

It is essential that the power consumption in sensor nodes be as minimal as possible, considering their remote deployment. This can be accomplished by minimizing the amount of computations and radio traffic originating from the node (Baghdasaryan, 2005). Sensor nodes go through the processes shown in Figure 6.2.

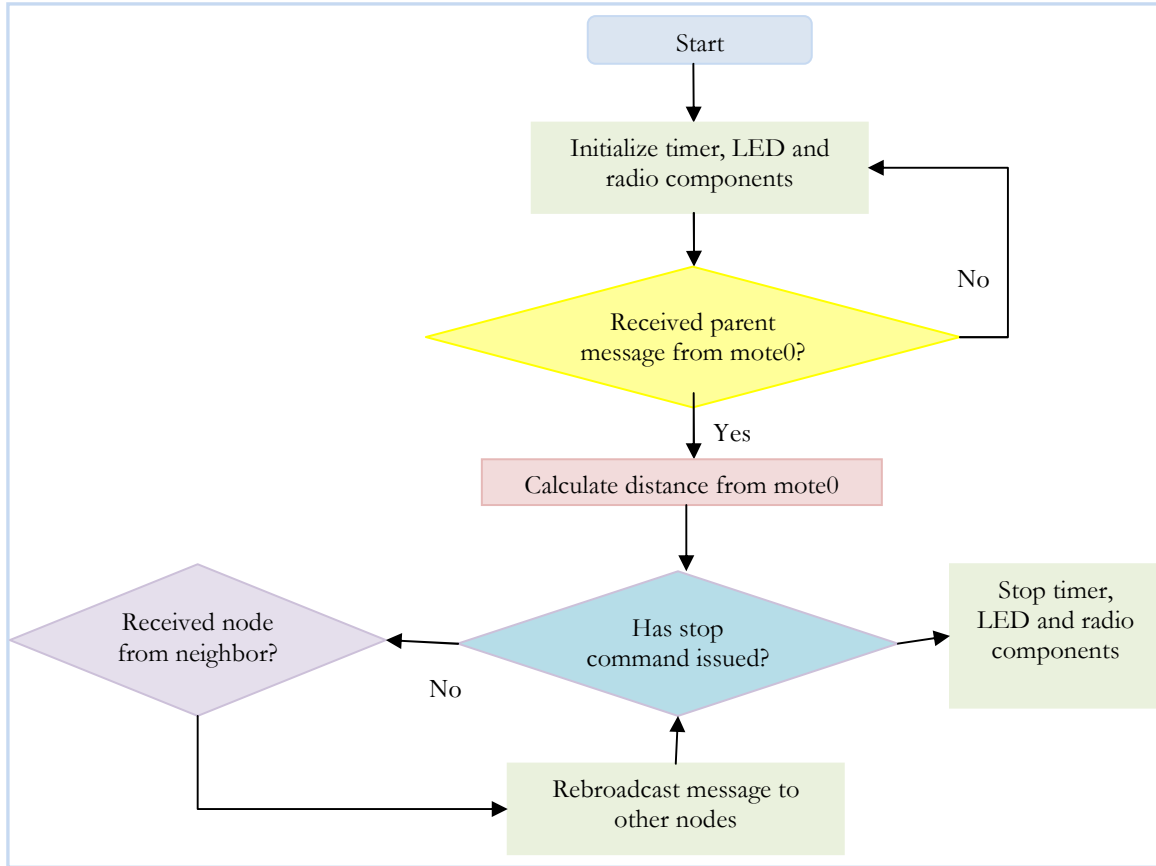


Figure 6.2 Sensor node functional loop

### 6.3. Base station design architecture

Any communication between a personal computer application and a simulated mote environment takes place via the base station, or equivalent Mote 0 in a simulated environment. Energy conservation is an important criterion in a sensor network, but it is not critical in base station nodes, as these are powered by a permanent source (Baghdasaryan, 2005).

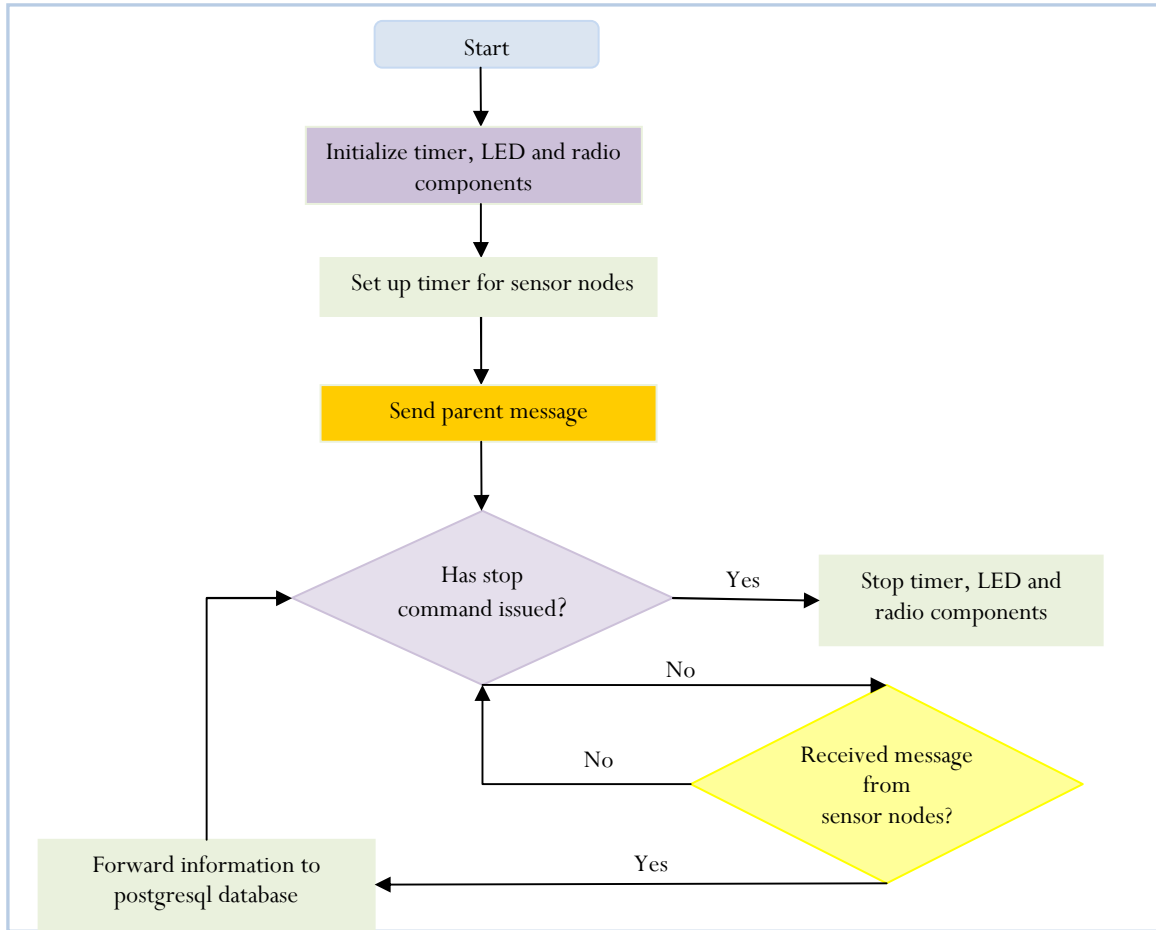


Figure 6.3 Base station functional loop

## 6.4. Wireless sensor application design architecture

The main purpose of the interface was to improve the accessibility and visual appeal in presentation of sensor data.



Development of a web-based interface for a wireless sensor network monitoring system

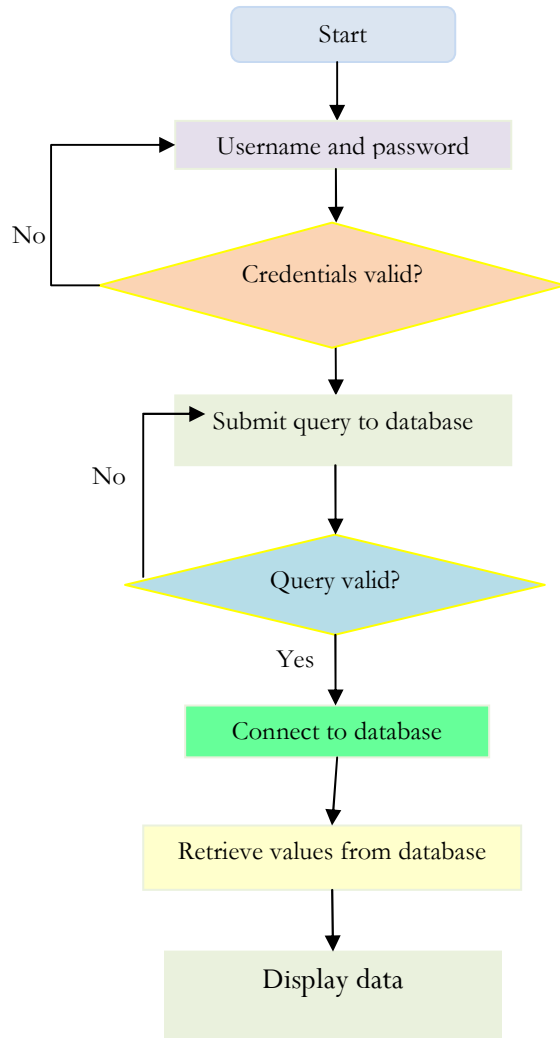


Figure 6.4 Web interface functional loop

## Chapter 7

### Wireless sensor application implementation

*The major Java classes used to build the wireless sensor application are described in this chapter. Java technologies have been chosen for their deployment as they are open source, they adhere to community standards and are easy to deploy on any system platform.*

#### 7.1. Introduction

The user interface enables the interaction between the user and the system. As previously mentioned, this was the main focus of the project, but due to their extensive implementation descriptions, Netbeans IDE, Tomcat 5.5, and associated classpaths were briefly described in this chapter. This interface is implemented in Java Server Pages and Servlets due to the fact that all libraries that interact with the mote environment and the interfaces for sending and receiving packets to the motes are implemented using Java.

## 7.1.1. Developer Installations

In order to successfully deploy the wireless sensor application, a series of software installations and configurations were required. The sequence of steps that were taken is described below:

### Step 1: Installation of the Netbeans IDE and Tomcat server

The Netbeans IDE provided a solid integrated development environment for editing the Java classes, the JSP web pages. The bundled Netbeans IDE contained a bundled Tomcat server, and the download distribution was unpacked into a single folder where the program was to be installed.

### Step 2: Addition of JFreeChart classes to the Netbeans IDE

In order to use the JFreeChart library, its classes were added to Netbeans as shown in Figure 7.1 below:

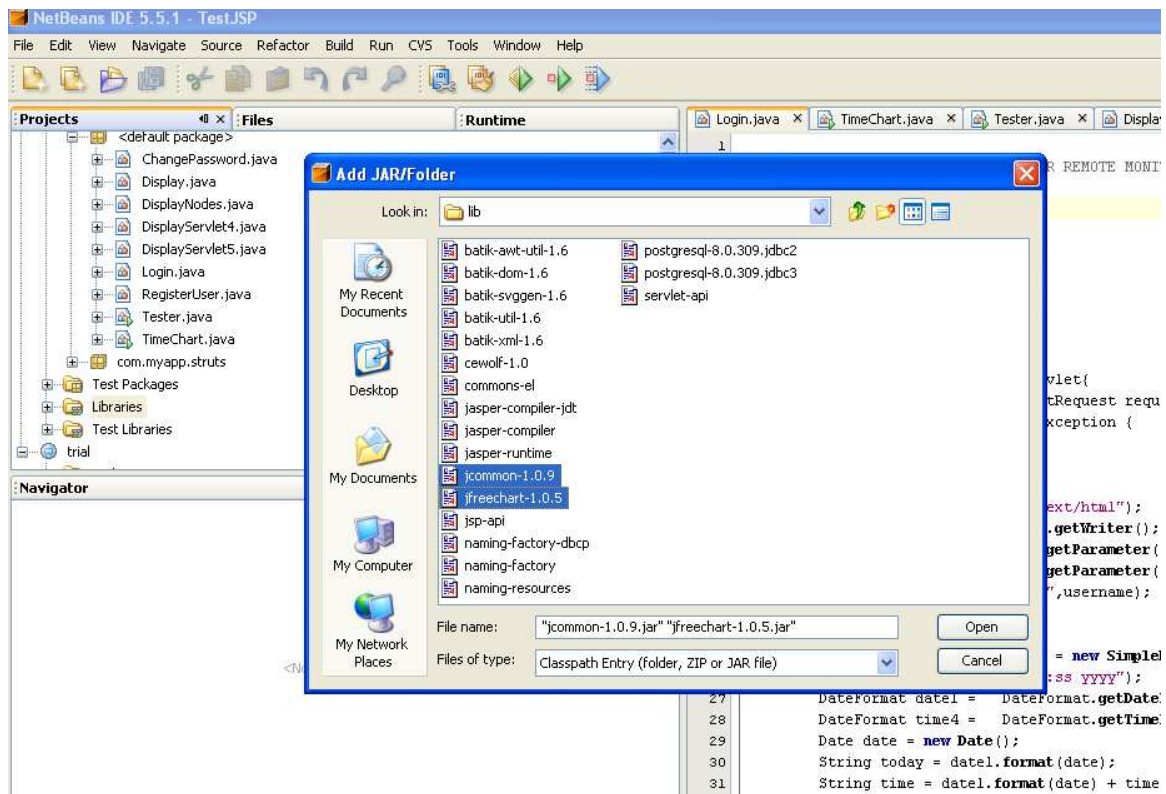


Figure 7.1 JFreeChart classes

Development of a web-based interface for a wireless sensor network monitoring system

The installation of PostgreSQL on Cygwin was achieved using the following instructions:

**Step 3: Unpacking the postgresql-8.0.7-1.tar.gz archive into the cygwin library using the instruction:**

```
tar -zxvf postgresql-8.0.7-1.tar.gz
```

**Step 4: Renaming the directory to something less complicated using the instructions:**

```
mv postgresql-8.0.18.tar.gz pgsql
```

**Step 5: Configuring, compiling and installing postgres using the instructions:**

```
cd pgsql  
./configure  
make  
make install
```

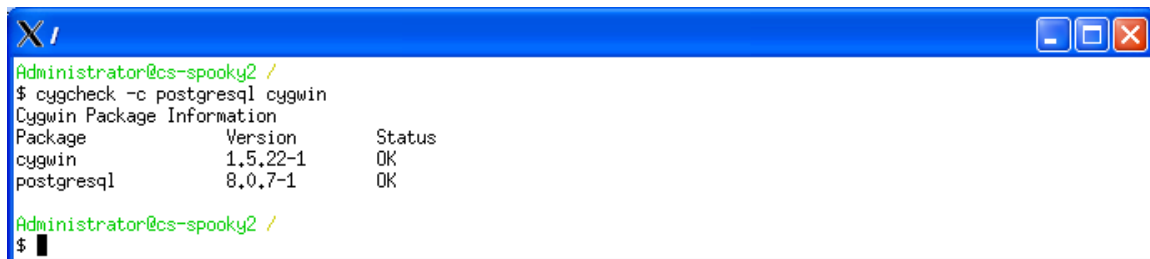
**Step 6: Setting the environment variables to tell Windows of the new inhibitor:**

```
PGHOME: "C:\cygwin\usr\local\pgsql"  
PGDATA: "C:\%PGHOME%\data"  
PGLIB: "%PGHOME%\lib"  
PGHOST: "localhost"
```

'Path' is modified by adding the following instruction:

```
%PGHOME%\bin; %PGHOME%\lib;c:\cygwin\bin
```

**Step 7: Verifying that PostgreSQL is properly installed:**



```
Administrator@cs-spooky2 /  
$ cygcheck -c postgresql cygwin  
Cygwin Package Information  
Package      Version      Status  
cygwin       1.5.22-1    OK  
postgresql   8.0.7-1     OK  
Administrator@cs-spooky2 /  
$
```

### Step 8: Addition of PostgreSQL JDBC driver to the Netbeans IDE:

The PostgreSQL JDBC driver is added to the Netbeans IDE as illustrated in Figure 7.2 below:

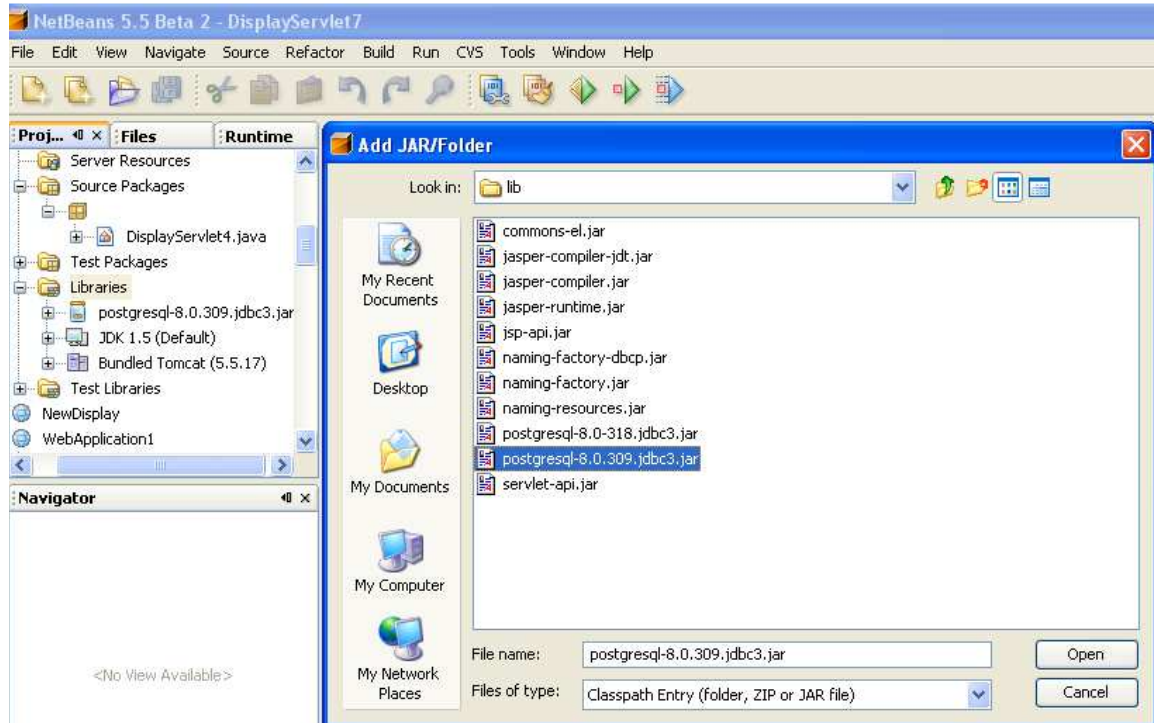


Figure 7.2 PostgreSQL jdbc driver

## 7.2. Displaying PostgreSQL data

A visual access paradigm is a preferred mechanism to read data from a PostgreSQL database. Simplicity was required in this interface for efficient retrieval and visualization of data relevant to assessing structural states. Thus, a set of simple alert icons were incorporated alongside the data retrieved from the database into the webpage to depict the status of the sensors on the field.

### 7.2.1. Java Server Pages and servlets

The individual pages are implemented using JSP pages and Cascading Style Sheets (CSS), as they allow the use of regular web page design programs and allow non-programmers to easily edit the look of the user interface. The servlets handle the requests from the user, and based on the needs, locate and return information to the JSP. The main servlet and JSP implementations are listed and described in Table 7.1 and Table 7.2.

FILE NAME	DESCRIPTION
Index.jsp	This is the login page for the wireless sensor application which takes the user inputs to authenticate him. It is illustrated in Figure 7.3.
Main.jsp	This is the main page of the application that is loaded after user authentication. It displays the username, the time he last logged in and links to query the sensor database. It is illustrated in Figure 7.4.
Administrator.jsp	Allows the administrator to set the threshold values that will be used in the application. It also allows the creation of new user accounts
DisplayValues.jsp	Displays the values retrieved from the database with the aid of a table and history graphs.
RegisterUser.jsp	This is an administrator page to allow new users to access the visualization tool.

**Table 7.1 Main JSP pages implementations**

FILE NAME	DESCRIPTION
Login.java	This servlet takes the input from the login page and verifies the credentials of the user. If passed, then the user is directed to secure main page else sent back to login page.
TimeChart.java	This is the servlet that processes the values from the database and generates a TimeSeries chart from the values.
Main.java	This servlet process the user requests and retrieves values for these requests from the PostgreSQL database. Any values attained are sent to the DisplayValues.jsp page.
Threshold.java	This servlet processes the administrator request to change the thresholds set to differentiate between critical, ok, and warning database values
RegisterUser	This servlet processed the request by the administrators to add new users, since a scalable interface was required
ChangePassword	Allows the users to change their login passwords

**Table 7.2 Main servlet implementations**

## Development of a web-based interface for a wireless sensor network monitoring system

In order to build and display the web interface servlets the following main imports were required:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.sql;
import java.util.*;
```

The connection to the database to obtain sensor readings was achieved using a PostgreSQL driver as follows:

```
Class.forName("org.postgresql.Driver");
con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/task",
"tele", "tiny");
```

The values from the database had to be separated into different levels of performance based on structural analysis. In order to achieve this, various threshold limits, set by an administrator, were used to separate the degree of damage to the structures being monitored. For this initial deployment, only the temperature value was used to determine whether the system was feasible as shown below:

```
operation = request.getParameter("select");
if(operation.equals("show all sensor nodes")){
    rs = stmt.executeQuery ("Select * from q1");
}
else if(operation.equals("latest sensor values")) {
    rs = stmt.executeQuery ("Select DISTINCT ON (nodeid)* from q1");
}
else if (operation.equals("critical sensor values")){
    rs = stmt.executeQuery ("Select DISTINCT ON (nodeid)* from q1 where temp >
+criticalThreshold+");
}
else if (operation.equals("show warning sensors")) {
    rs = stmt.executeQuery("Select DISTINCT ON (nodeid)* from q1 where temp >
+warningThreshold+ " );
}
```



Development of a web-based interface for a wireless sensor network monitoring system



Figure 7.3 Visual look of the web-based interface

# Development of a web-based interface for a wireless sensor network monitoring system

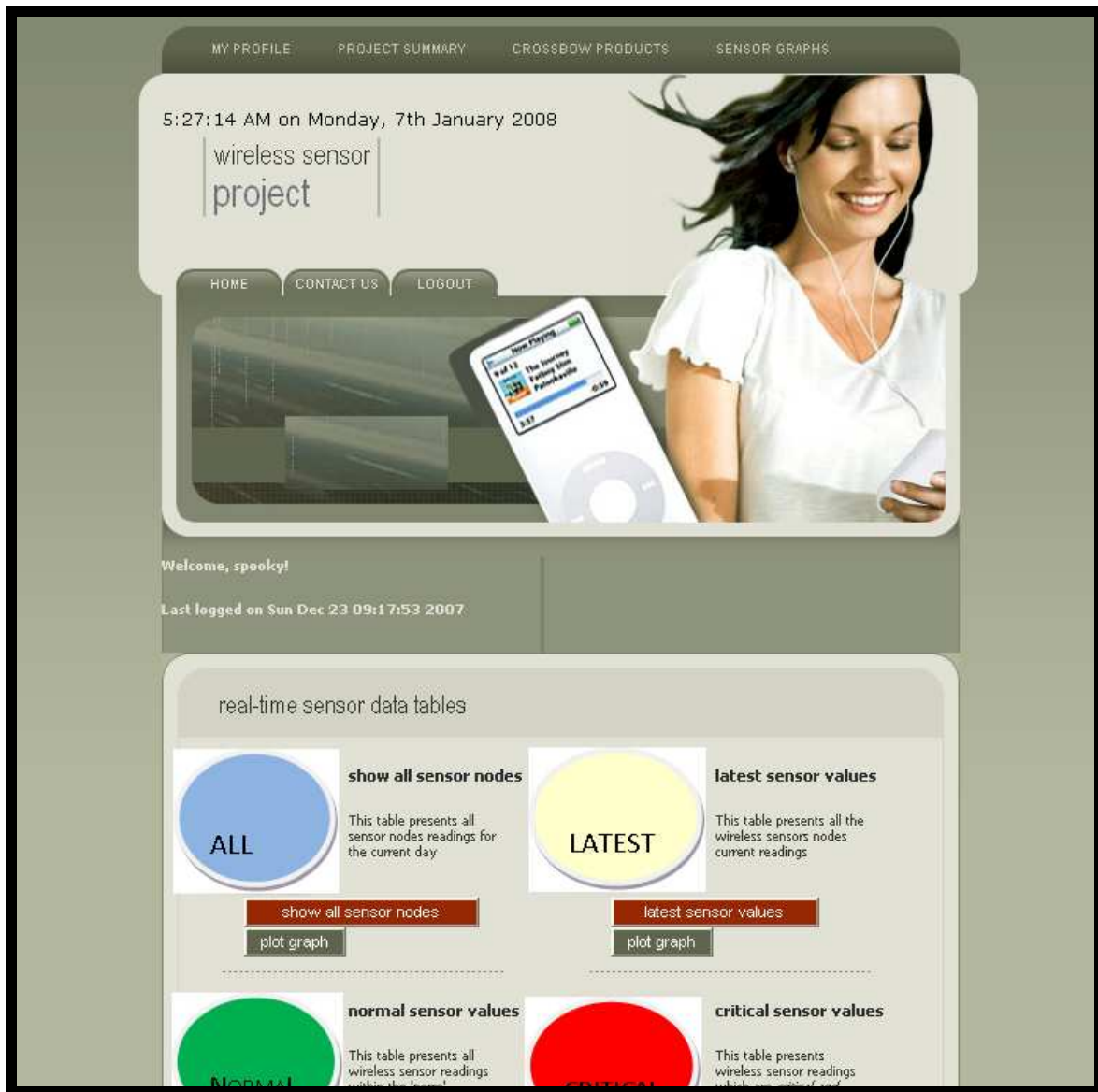


Figure 7.4 Main.jsp

## 7.2.1. ARRAYLISTS

In order to handle the large amounts of sensor data, ArrayLists were incorporated to aid in displaying the data in a tabular format on the web interface. ArrayList are a resizable-array implementation of the List interface, similar to Vectors, except that they are synchronized. Code written using ArrayLists is easier to handle and ‘cleaner’ compared to the corresponding array as shown in Table 7.3.

Adding an item to the end of a list	
ArrayList	Array
<pre>theItems.add(new Integer(intToAdd))</pre>	<pre>If(number of Items == MAX) // deal with this out-of-bounds error else     for(int i = numberOfItems;i&lt;pos;i--)         theItems [i] = theItems[i-1];     theItems[pos] = intToAdd;     numberOfItems ++; }</pre>
Print a list	
ArrayList	Array
<pre>for (int i = 0;i&lt;theItems.size();i++){     System.out.print(theItems.get(i) + ""); }</pre>	<pre>for (int i = 0; i &lt; numberOfItems; i++){     System.out.print(theItems[i] + ""); }</pre>

**Table 7.3 Comparison of ArrayLists to Arrays**

## Development of a web-based interface for a wireless sensor network monitoring system

```
//getColumnNames from the database and set ArrayLists
```

```
ArrayList a = new ArrayList();
for(int k = 1; k <= numCols; k++){
    a.add(rsmd.getColumnname(k));
}
request.setAttribute("Columnname",a);

ArrayList b = new ArrayList();
ArrayList dt = new ArrayList();
while(rs.next()) {
    for(int k = 1;k <=numCols ;k++) {
        b.add(rs.getObject(k).toString());
    }
}
request.setAttribute("Columndata",b);

ServletContext s =this.getServletContext();
RequestDispatcher dr = s.getRequestDispatcher("/DisplayValues.jsp");
dr.include(request,response);
```

```
//gets ArrayList previously set in the Servlet
```

```
<% String name = (String)session.getAttribute("User");
    String time = (String)session.getAttribute("time");
    String heading = (String)session.getAttribute("heading");
    ArrayList geta = (ArrayList)request.getAttribute("Columnname");
    ArrayList getb = (ArrayList)request.getAttribute("Columndata");
%>
```

```
//displays the results in a tabular format on the web interface
```

```
<% for(int k = 0; k < geta.size(); k++){
out.print("<th class = \"MYTABLE\" align = \"center\">" + geta.get(k) + "\t" +
</th>" );
    }
out.print("<th class = \"MYTABLE\" align = \"center\"> status</th>" );
out.print("<th 4class = \"MYTABLE\" align = \"center\"> additional info </th>");

for (int s = 0;s<getb.size();s+=geta.size()){
    out.println("<TR CLASS = \"MYTABLE\">");
    for (int p = 0;p <geta.size();p++){
        Object y = (Object)getb.get(s + p);
        out.print("<TD CLASS = \"MYTABLE\" >" +y+ "</td>");
    }
}
%>
```

### 7.2.3 GENERATING PLOTS

The plots used in the interface are generated using the JFreeChart library. JFreeChart was chosen because of its ability to generate different types of plots using Java classes (JFreeChart, 2007). The graphs used for visualization of data are generated from the PostgreSQL database. The JFreeChart library is capable of running in a servlet environment or can be displayed as a picture and in order to achieve this; the following classes have to be imported:

```
import org.jfree.chart.*;      // generates the chart
import org.jfree.data.xy.*;    // draws the axes
import org.free.data.jdbc.*;   // connects to the PostgreSQL database
import java.io.*;              // output the generated graph to a file
import org.jfree.data.time.*   // plots a time series graph
import java.sql;
```

The connection to the database is achieved using a PostgreSQL driver as follows:

```
Class.forName("org.postgresql.Driver");
con = DriverManager.getConnection("jdbc:postgresql://localhost:5432/task",
"tele", "tiny");
```

Any data collected from a sensor network is displayed with a time stamp as shown in Figure, thus the TimeSeries data structure to represent the data on the interface was used.

## Development of a web-based interface for a wireless sensor network monitoring system

In order to create a TimeSeries graph for each sensor node in the network the following code was used:

```
//creates a TimeSeries graph for each node  
  
for(int k = 1; k<=countNodes; k++){  
    times[k] = new TimeSeries("node"+k+"", Minute.class);} 
```

```
// gets values from the TimeStamps and adds them to the Minute class  
  
TimeSeries [] times = new TimeSeries [10000];  
while(rs.next()){  
    Timestamp z = rs.getTimestamp(2);  
    Calendar cal = Calendar.getInstance();  
    cal.setTime(z);  
    year = cal.get(Calendar.YEAR);  
    month = cal.get(Calendar.MONTH);  
    day = cal.get(Calendar.DATE);  
    hour = cal.get(Calendar.HOUR_OF_DAY);  
    min= cal.get(Calendar.MINUTE);  
    sec = cal.get(Calendar.SECOND);  
    times[w].add(new Minute(min, hour, day, month, year), t);  
    }  
  
datasets = new TimeSeriesCollection();  
for(int j = 1;j<= counting;j++){  
    datasets.addSeries(times[j]);  
    } //end for
```

```
private JFreeChart createChart(final XYDataset dataset) {
    final JFreeChart chart = ChartFactory.createTimeSeriesChart("Sensor nodes
graph", "time", "temperature", dataset, true, true, false);
    chart.setBackgroundPaint(Color.white);
    final XYPlot plot = chart.getXYPlot();
    plot.setBackgroundPaint(Color.lightGray);
    plot.setDomainGridlinePaint(Color.white);
    plot.setRangeGridlinePaint(Color.white);
    final XYLineAndShapeRenderer renderer = new XYLineAndShapeRenderer();
    renderer.setSeriesLinesVisible(0, false);
    renderer.setSeriesShapesVisible(1, false);
    plot.setRenderer(renderer);
    final NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
    rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());    return
chart;    }

public static void main(final String[] args) {
    final TimeChart demo = new TimeChart("Sensor nodes performance");
}
```

The plots generated were displayed as Joint Photographic Experts Group (JPEG) images. Image files were used to display the plot because it requires less overhead on the web browser. In order to achieve this, the JFreeChart was saved as a JPEG using:

```
ChartUtilities.saveChartAsJPEG(new File ("images/chart5.jpg"), chart, 500,300);
```

## Development of a web-based interface for a wireless sensor network monitoring system

If a query is sent to the database to display information, the retrieved data is shown in a table and graph as shown below in Figure 7.5.

Node id	Temperature (F)	Timestamp	Additional info.	Status
1	948	3:52:26 PM	Point 1	CRITICAL
2	26	3:52:29 PM	Point 2	NORMAL
3	110	3:52:30 PM	Point 3	NORMAL

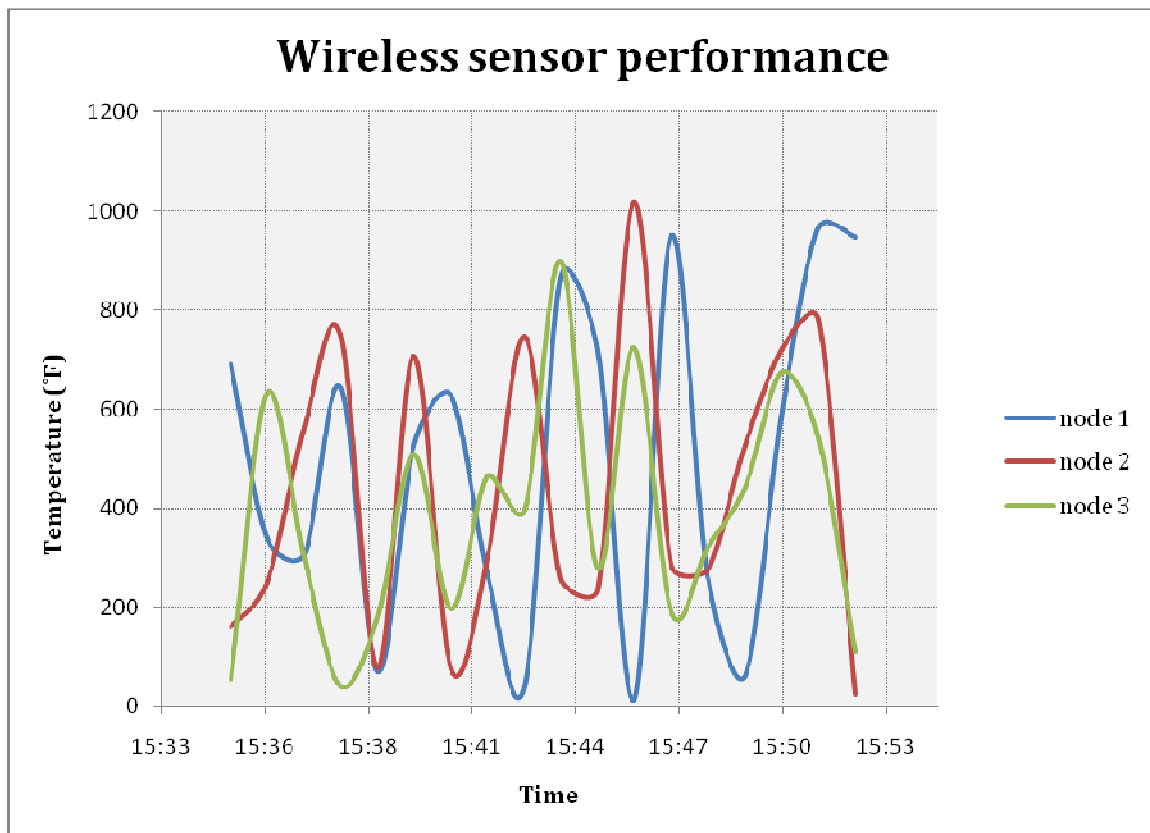


Figure 7.5 Tabular and graph display



## Chapter 8

### Data acquisition implementation and testing

*In order to determine the feasibility of linearly aligned sensor network, the performance of both the hardware and software components of the wireless sensor system need to be evaluated. This chapter presents experiments performed that characterise the hardware used in this project as well to evaluate the suitability of TinyDB, the application software to query the nodes.*

#### 8.1. Introduction

The hardware portion of the data acquisition system for the project consists of 6 433 MHz Mica2 sensor nodes, 5 sensor boards, a programming board, a TOSSIM simulator, a Mica2 base station and a portable personal computer. The primary focus for the implementation and system testing of these devices was to evaluate the properties of the wireless link and the delays between a linearly aligned network of sensor nodes and the base station, which had influence of the amount of data captured at the base station.

## 8.2. Hardware and software installations

In order to successfully deploy sensor nodes, a series of hardware and software installations and configurations were required on the network devices. The sequence of steps that were taken are described below:

### Step 1: Installation of the TinyOS operating system and packages onto the computer

The initial setup of the data acquisition system involves the installation of the TinyOS and its associated packages, shown in Figure 8.1, on a Cygwin platform.

```
avr-binutils-2.13.2.1-1w
avr-gcc-3.3tinyos-1w
avr-insight-pre6.0cvs.tinyos-1w
tinyos-tools-2.0.2-2
task-tinydb-1.1.3
tinyos-contrib-1.1.3
mspgcc-win32tinyos
avr-libc-20030512cvs-1w
avarice-2.0.20030825cvs-1w
nesc-1.2.8a-1
tinyos-2.0.2-2
```

Figure 8.1 TinyOS packages for sensor nodes

### Step 2: Installation of TinyOS into wireless sensor nodes

TinyOS was installed into a wireless sensor node via a programming board which attaches to a computer through a serial interface. The attachment of a node to the programming board is illustrated in Figure 8.2.



**Figure 8.2** Sensor node attached to a programming board

A root node, whose id is typically mote0, was programmed in cygwin using the following commands:

```
cd /opt/tinyos-2.x/apps/TOSBase
export MIB510="/dev/com1"
make clean
make mica2 install
```

All other sensor nodes required non-zero, unique node identification numbers during installation. TinyOS was uploaded into these sensor nodes using the following commands in the TOSBase folder:

```
cd /opt/tinyos-2.x/apps/TOSBase
export MIB510="/dev/com1"
make clean
make mica2 install.<unique mote number>
```

### Step 3: Installation TinyDB components into sensor nodes

To enable querying of nodes from a base station, a unique copy of TinyDB software had to be installed into all sensor nodes using the following commands:

```
cd /opt/tinyos-2.x/apps/TinyDBApp
```

- To compile and install a custom version of TinyDB in the root node the following command was used :

```
make mica2 install.0 mib510,/dev/ttyS0. //via the serial port
```

- To compile and install TinyDB in the other sensor nodes the following command was used :

```
make mica2 reinstall.<motenumber> mib510,/dev/ttyS0. //via the serial port
```

### Step 4: Setting up the base station node

When the nodes had been successfully installed with TinyOS and TinyDB components, the root node was re-attached to the programming node, and became the base station for the wireless sensor devices.

### Step 5: Setting up wireless sensor nodes

To measure environmental parameters, the wireless sensor nodes were connected to the sensor board as illustrated in Figure 8.3.

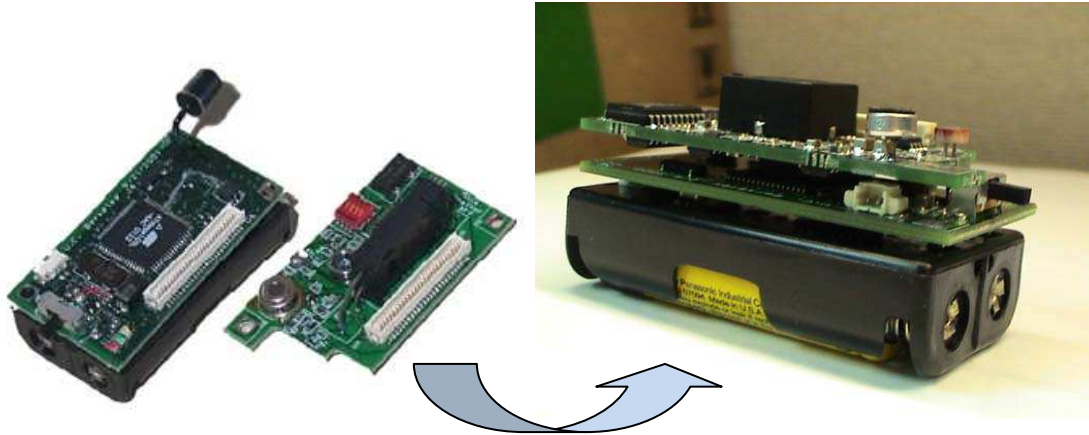


Figure 8.3 Mica2 node with sensor board

### Step 6: Running the TOSSIM simulator

TOSSIM, described in Section 4.6.1, compiles directly from TinyOS and was required to scale the data collecting devices to any number required in the experiments. TinyViz, illustrated in Figure 8.4, provides a graphical user interface layer to the TOSSIM simulations and allows the user to visually inspect debug messages, radio messages, radio connectivity, and mote placement. The following commands invoke TinyViz:

```
cd /opt/tinyos-2.x/apps/TinyDBApp
make -f MakePC
export DBG=usr1
tinyviz ./build/pc.main.exe <numnodes>
```

## Development of a web-based interface for a wireless sensor network monitoring system

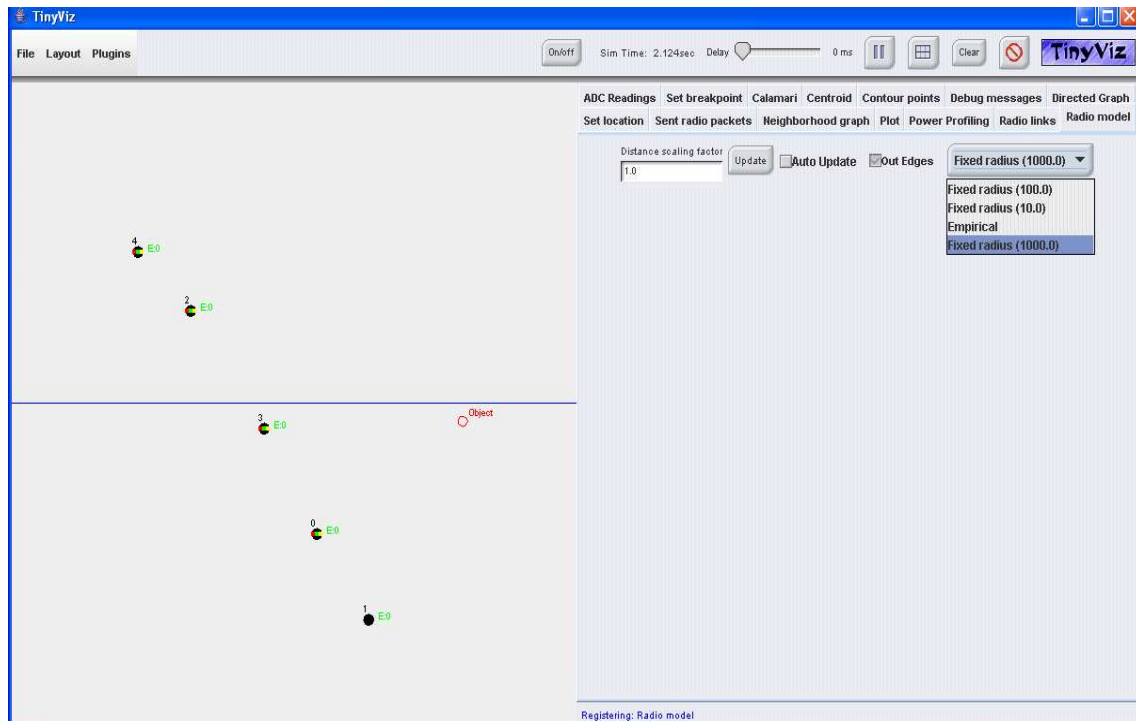


Figure 8.4 TinyViz graphic user interface

### Step 7: Invoking the SerialForwarder program.

In order for the sensor readings to be available at the base station, SerialForwarder (described in Section 4.5.2) needed to be invoked to open a connection port using the command:

```
cd /opt/tinyos-2.x/tools/java
java net/tinyos.sf.SerialForwarder -comm serial@COM1:57600 // in notes
java net/tinyos.sf.SerialForwarder -comm tossim-serial // in simulator
```

An illustration of the SerialForwarder program is shown in Figure 8.5.

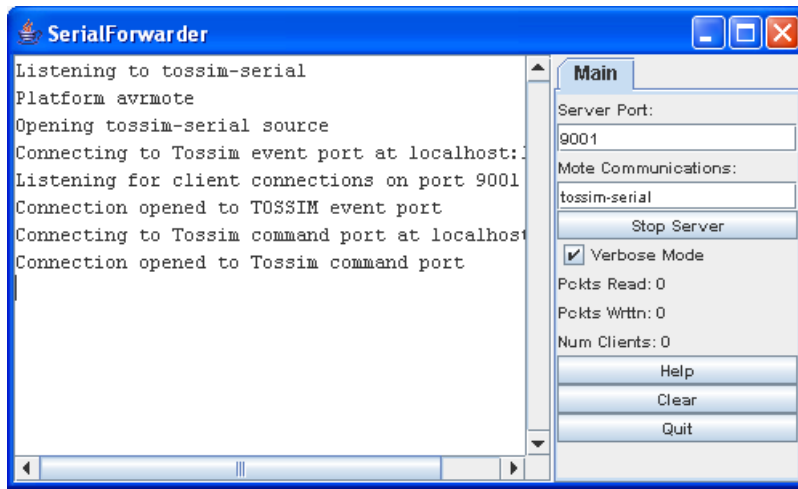


Figure 8.5 SerialForwarder program

### Step 8: Setting up TinyDB.

TinyDB is the overlay application that is used to query the sensor nodes in the network. To open the TinyDB GUI on the computer (shown in Figure 8.6), the following command had to be issued:

```
cd /opt/tinyos-2.x/tools/java  
java net.tinyos.tinydb.TinyDBMain
```

By checking the 'Log to Database' option in the TinyDB GUI, values from the sensor network were logged to a PostgreSQL database.

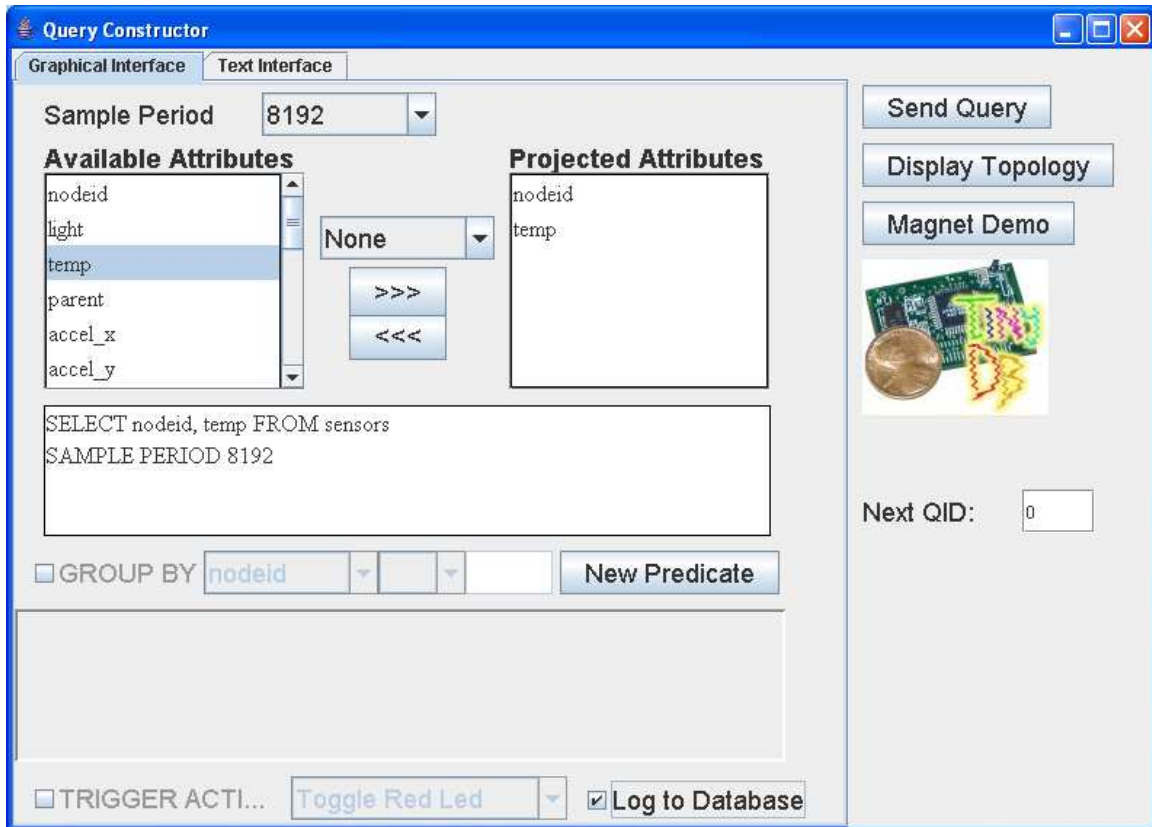


Figure 8.6 TinyDB graphic user interface

### Step 9: Installation and verification of a PostgreSQL connection

When TinyDB receives results from the wireless sensor network, the data is displayed within the TinyDB GUI as a spreadsheet. This is not a suitable method if data queries occur over an extended period of time or if there is need that the data be accessed via another application; hence it was desirable that data be stored in a PostgreSQL database.

The connection to the PostgreSQL database, where all sensor node readings were stored, had to be verified via pgAdmin III, whose interfaces are shown in Figure 8.7 and Table 8.1.



Development of a web-based interface for a wireless sensor network monitoring system

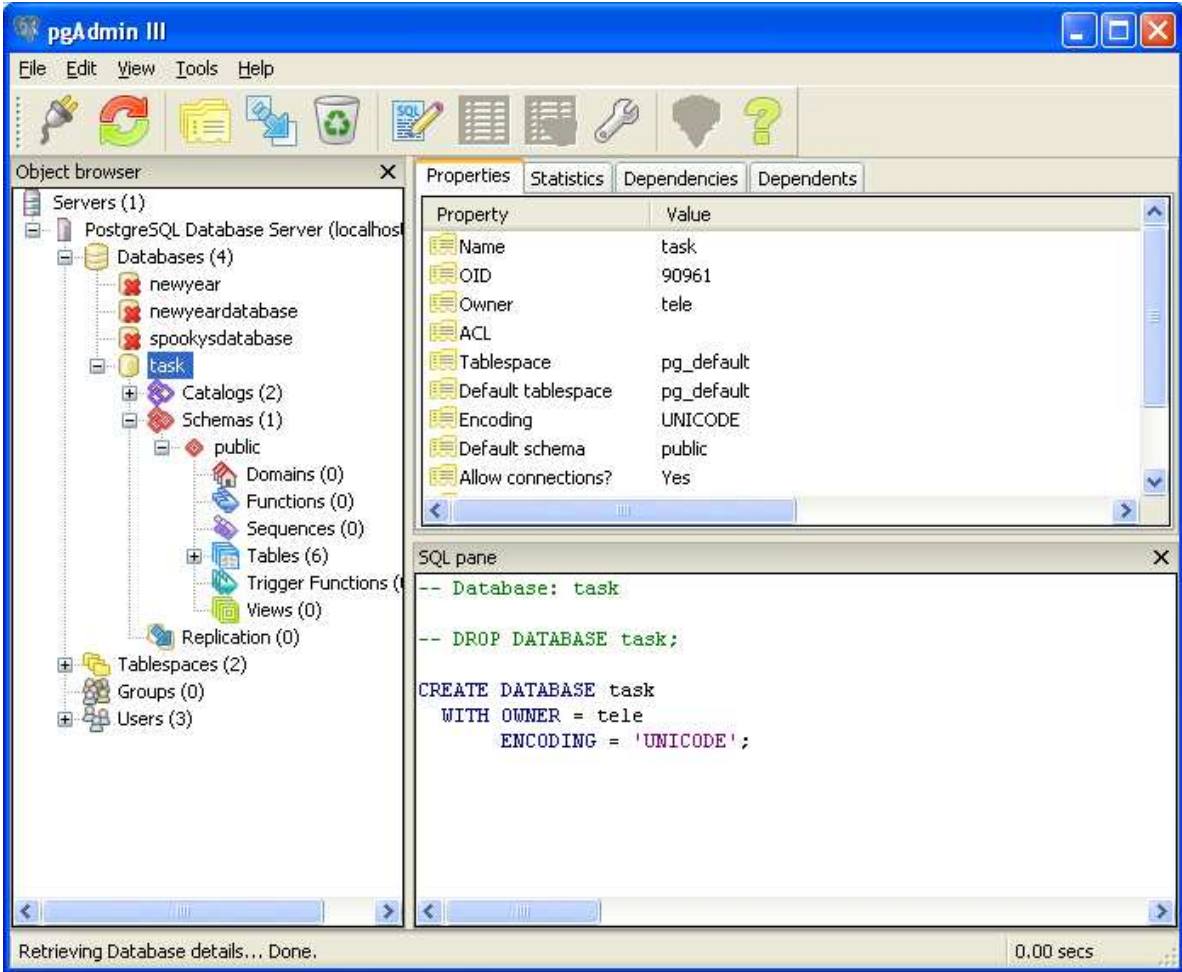


Figure 8.7 PgAdmin III interface

Edit Data - PostgreSQL Database Server (localhost:5432) - task - q1

	oid	result_time timestamp without time zone	epoch integer	nodeid integer	temp integer
1	90970	2007-11-18 15:35:00.85225	1		
2	90971	2007-11-18 15:35:01.00525	1		
3	90972	2007-11-18 15:35:01.18725	1		
4	90973	2007-11-18 15:35:01.26325	1		
5	90974	2007-11-18 15:35:01.41125	2	3	826
6	90975	2007-11-18 15:35:01.64625	2	4	709
7	90976	2007-11-18 15:35:01.81725	2	1	690
8	90977	2007-11-18 15:35:01.98425	2	2	25
9	90978	2007-11-18 15:35:02.55625	3	4	244
10	90979	2007-11-18 15:35:02.94725	3	1	339
11	90980	2007-11-18 15:35:03.10325	3	3	692

Table 8.1 A PostgreSQL table

Temperature values stored in the database are in Fahrenheits. A conversion from Fahrenheits to Celsius is achieved using the formula:

$$T_c = (T_f - 32) * 5/9$$

where  $T_c$  is the temperature on the Celcius scale and  $T_f$  is the temperature on the Fahrenheit scale.

### **8.3. Hardware evaluation of Mica2 nodes using continuous querying**

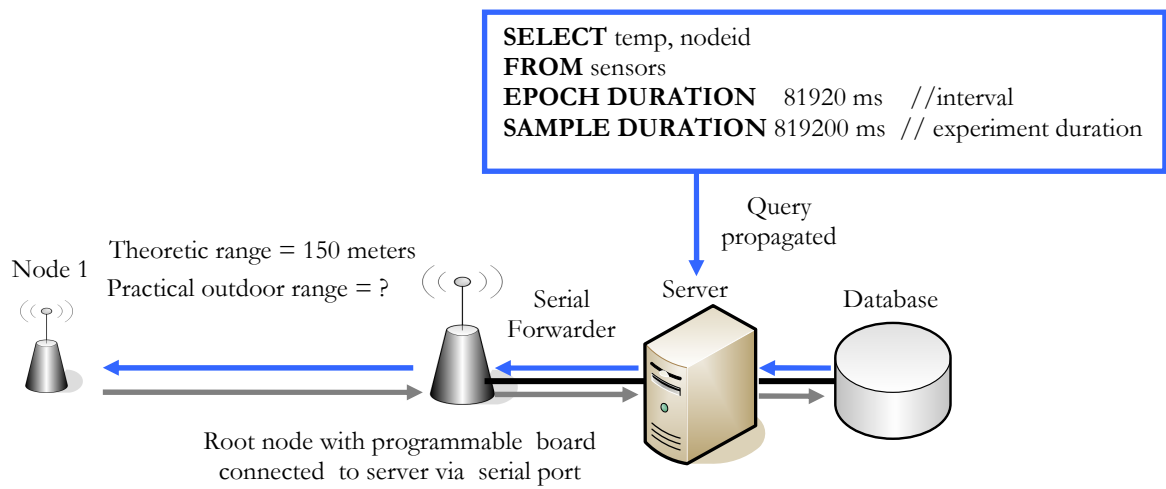
The performance of wireless sensor network devices was evaluated using experiments to test the transmission ranges and battery life of the nodes. Continuous queries were used because they present the worst possible scenario in deployed wireless sensor nodes. The epoch duration used for sampling data was randomly selected from a list available in the TinyDB querying tool.

#### **8.3.1. Transmission range of Mica2 nodes**

The theoretic outdoor transmission range of 433MHz Mica2 nodes is 500 feet (160 meters) (Akyldiz, 2002). The robustness of a network built using the Mica2 nodes can be verified by practically measuring their outdoor transmission range, based on the percentage packet loss from a single sensor node to the base station.

The experimental procedure involved the outdoor placement of nodes in a linear network as illustrated in Figure 8.8. Tests were run in both sunny and rainy weather conditions. The received data packets were plotted in Figure 8.9 and Figure 8.10. A query was sent from the base station, and by increasing the distance between communicating nodes by 10 meters at the time, the percentage of readings sent back by the sensor node and logged into the database illustrated the reliability of the link at various ranges. Using a single sensor node and base station meant any undelivered packets had not collided with data packets from other nodes, but had been lost because of the unreliable wireless link.

# Development of a web-based interface for a wireless sensor network monitoring system



**Figure 8.8** Transmission range experiment

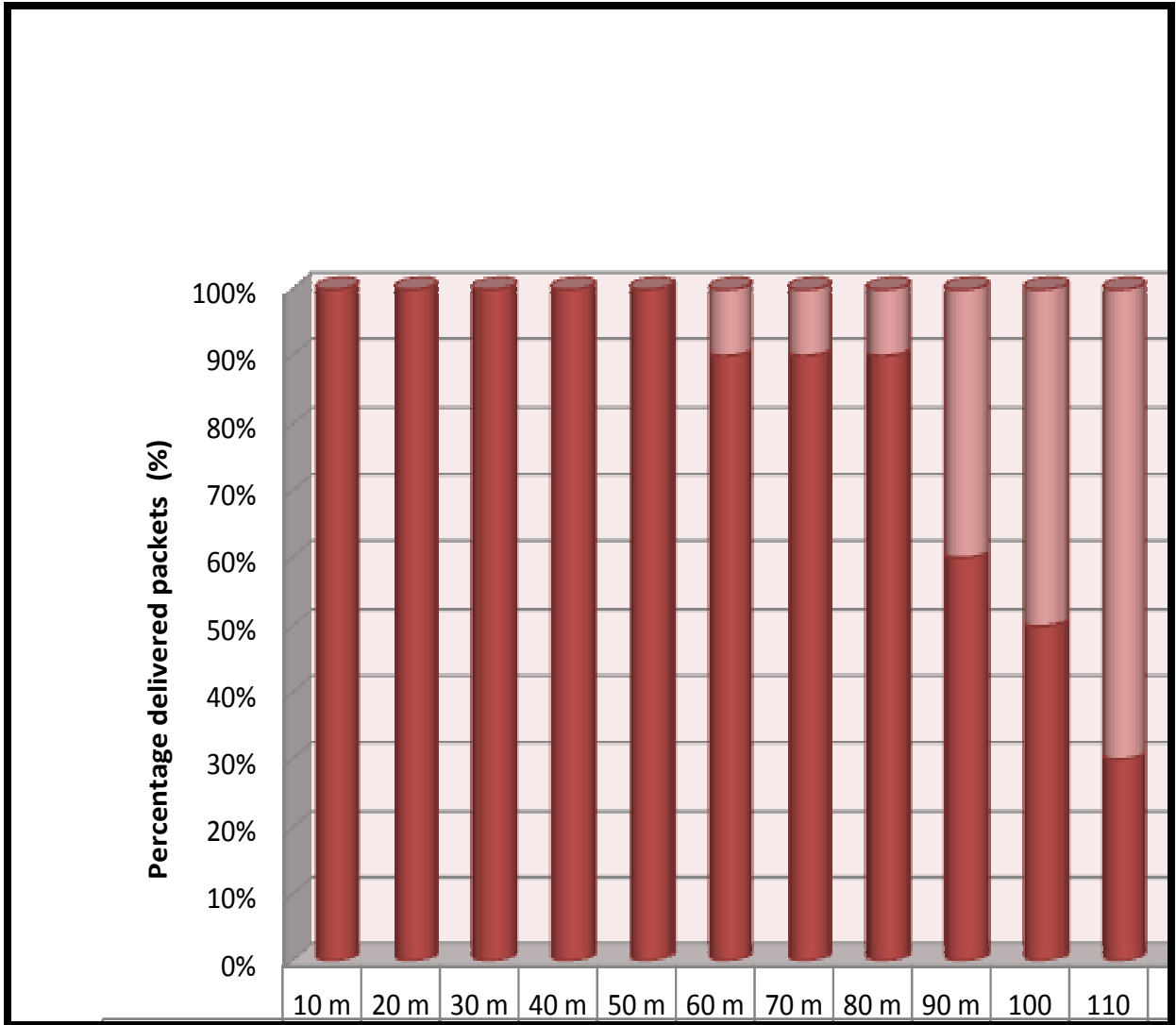
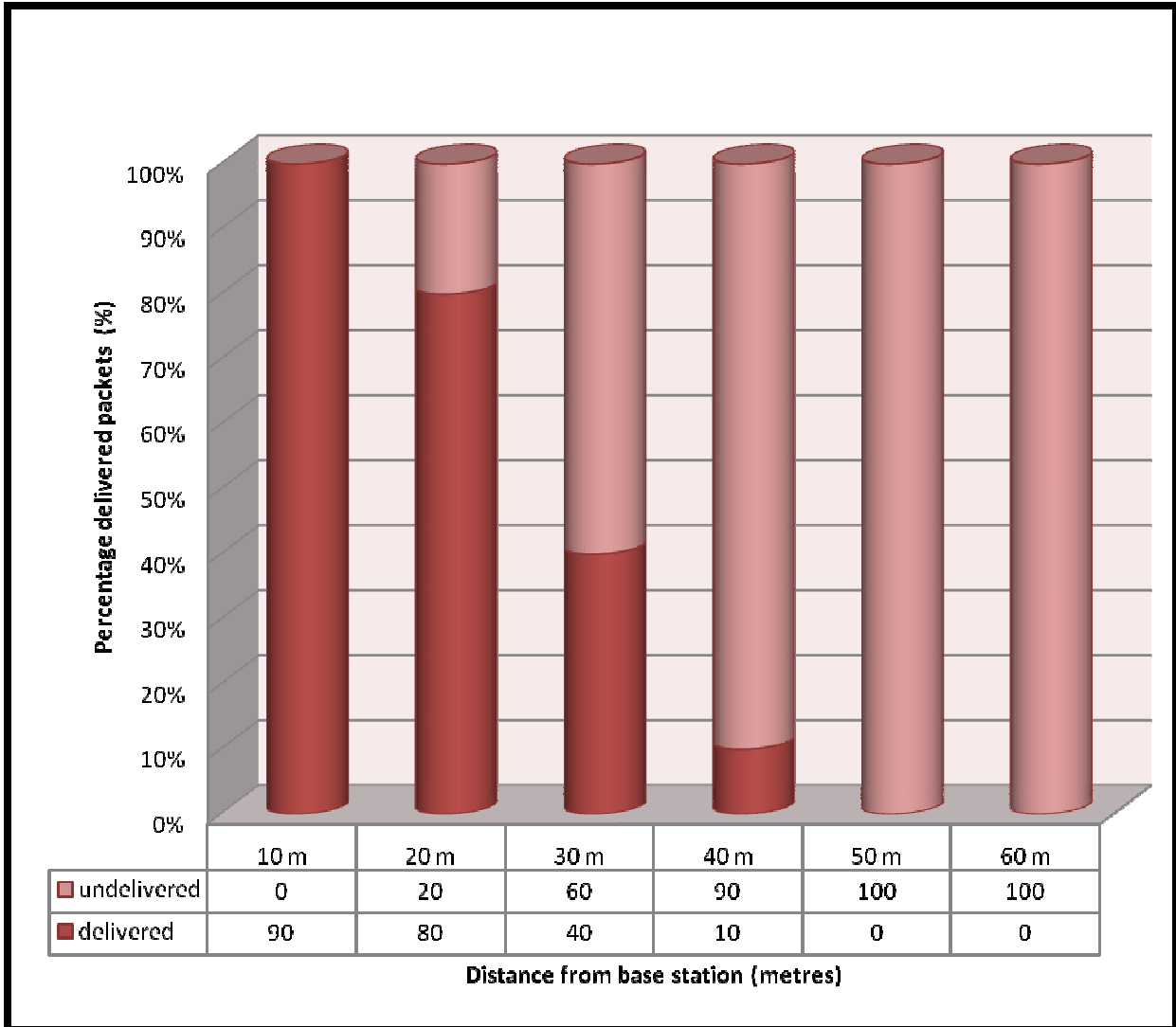


Figure 8.9 Transmission ranges of Mica2 nodes (sunny weather conditions)



**Figure 8.10 Transmission ranges of Mica2 nodes (rainy weather)**

Figure 8.9 illustrates that at a range of 80 meters (250 feet), the fraction of delivered readings from the sensor node was still representative of the original pool of packets, as only one reading does not arrive at the base station. However, when a similar experiment was carried out under rainy weather (Figure 8.10), a considerable amount of the data packets were dropped during transmission, resulting in transmission distances limited to an average distance of about 20 metres. It was deduced that the presence of moisture particles causes poor transmission of packets. The remainder of the experiments carried out (in sunny weather conditions) consisted of a linear placement of nodes at an approximate distance of 80 meters (250 feet), unless stated otherwise.

### 8.3.2. Battery lifetime of Mica2 nodes

Mica2 sensor nodes are powered using two alkaline batteries of 1.5 volts each. Power is a limited resource in wireless sensor applications, hence in real world deployment of these devices there is a need to incorporate power management mechanisms to extend the up-time of the sensor nodes (Baghdasaryan, 2005). The voltage reading of the battery in a sensor node can be queried using TinyDB. An experiment to determine the up-time of a network was performed using 3 linearly aligned sensor nodes and a base station illustrated in Figure 8.11.

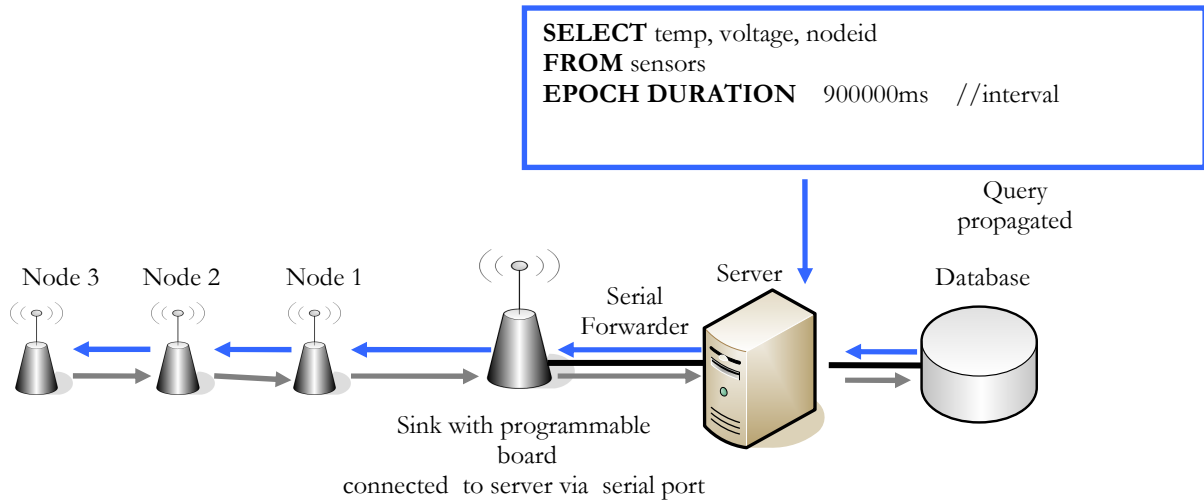


Figure 8.11 Battery life experiment

The sensor nodes were left running until no reading arrived in the PostgreSQL database, because the nodes had very low or no voltage left, and the results were plotted in Figure 8.12.

## Development of a web-based interface for a wireless sensor network monitoring system

No readings were logged into the database after approximately 2.25 days, with almost 150 readings logged for each node. The last reading from the nodes occurred when node 1 had a voltage of approximately 2 volts. It was concluded that at this voltage, packet losses are quite high, therefore a solution is required to extend the up-time of the nodes. Power harvesting techniques should be considered, for example, if the nodes are deployed in a real world application along a high voltage transmission line.

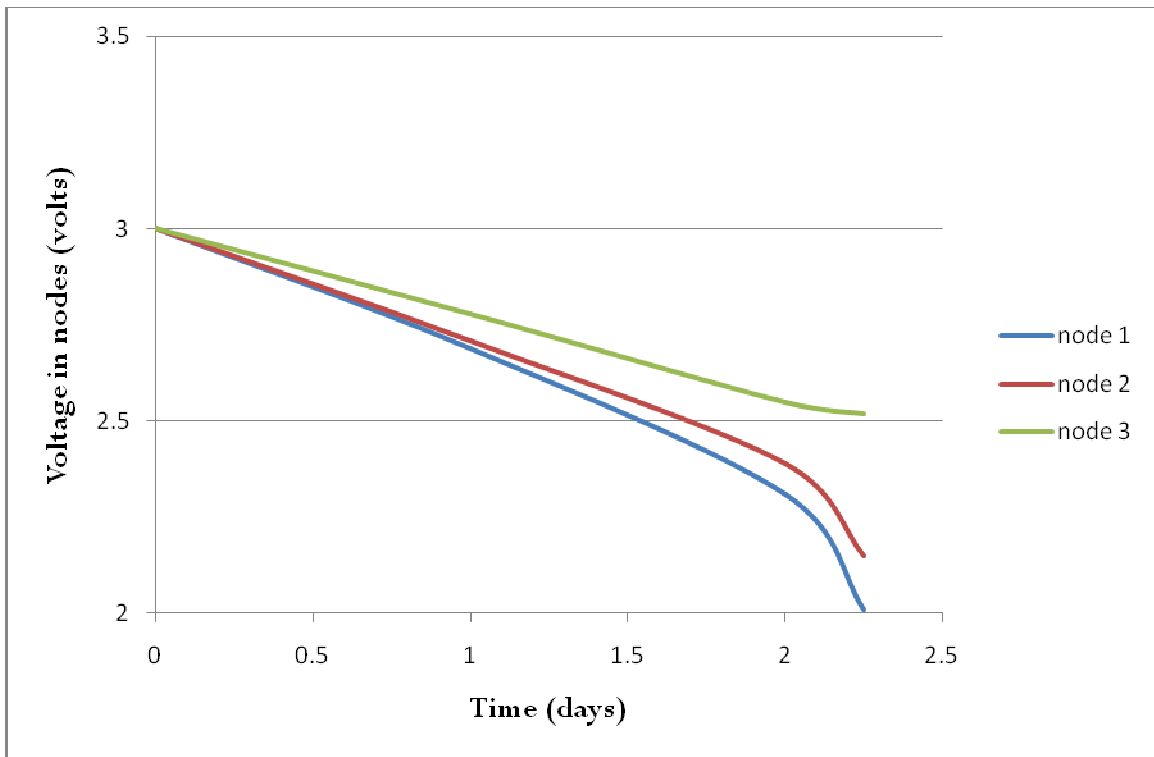


Figure 8.12 Battery lifetime of wireless sensor nodes

### 8.3.3. Scalability of Mica2 nodes

The topology of a wireless sensor network is an important factor that determines its scalability. An experiment was carried out to determine how the number of wireless sensor nodes a linearly aligned topology could handle. A linearly aligned network, shown in Figure 8.13, is the most challenging, as communication is limited to two neighbouring nodes.

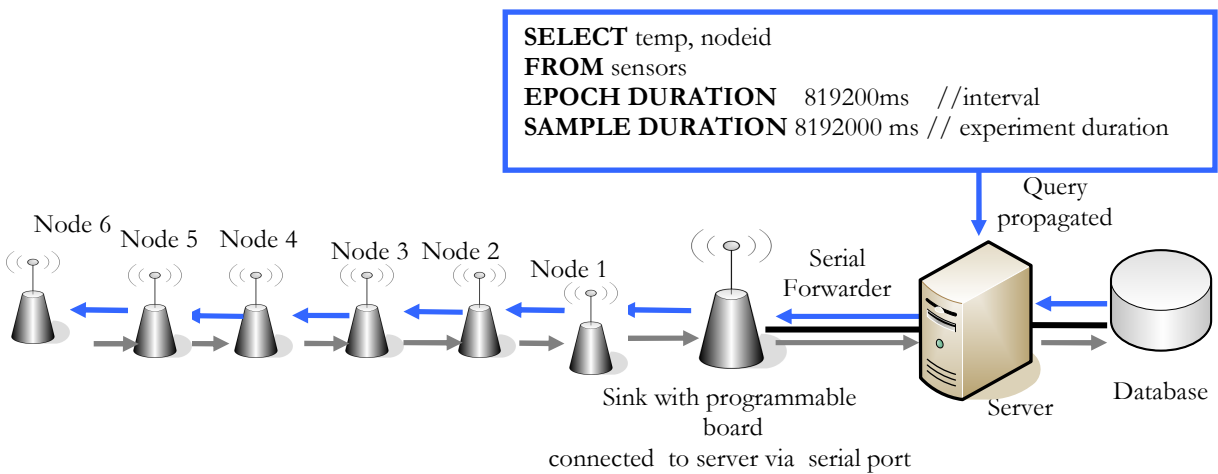
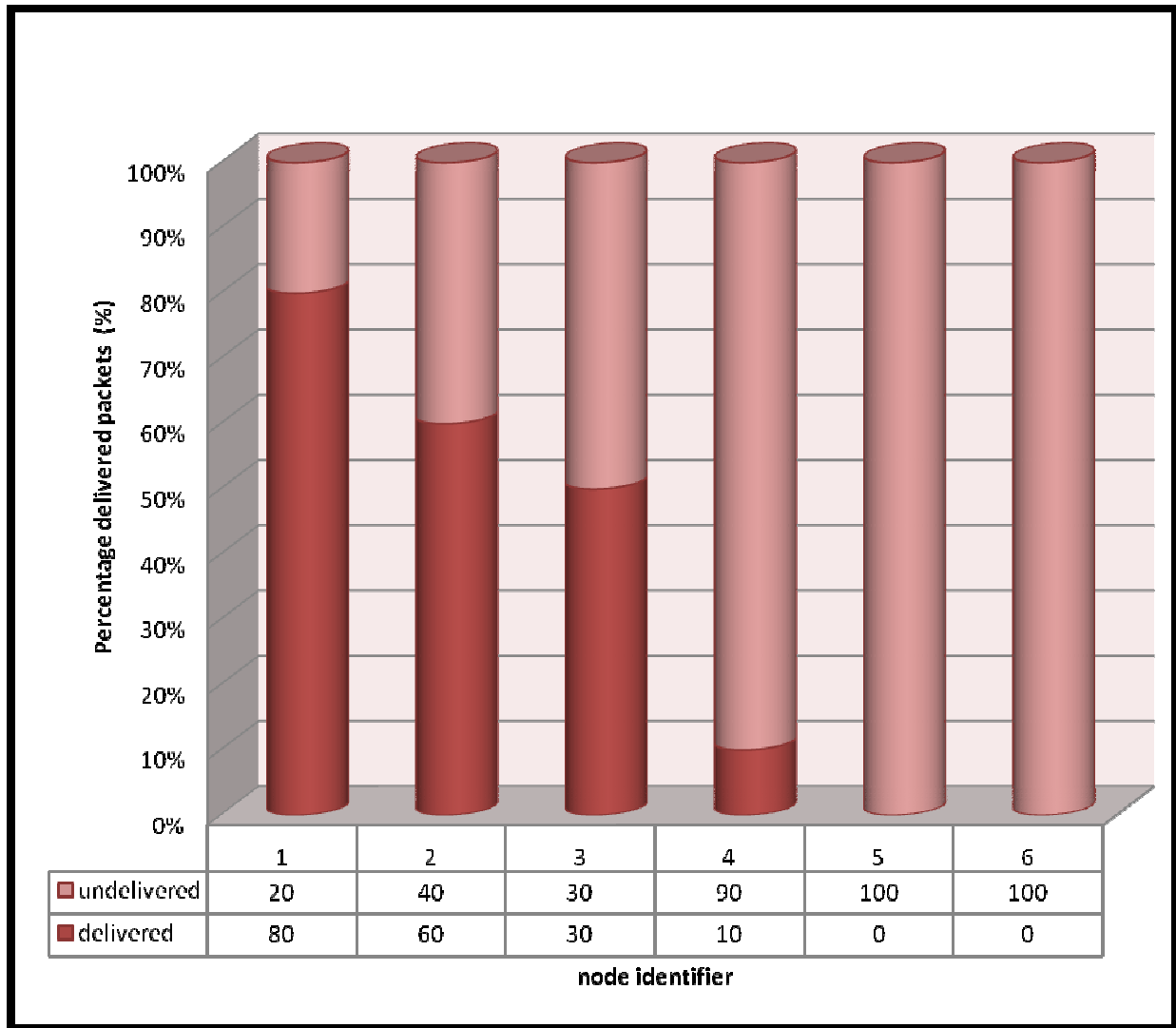


Figure 8.13 Scalability of Mica2 nodes experiment





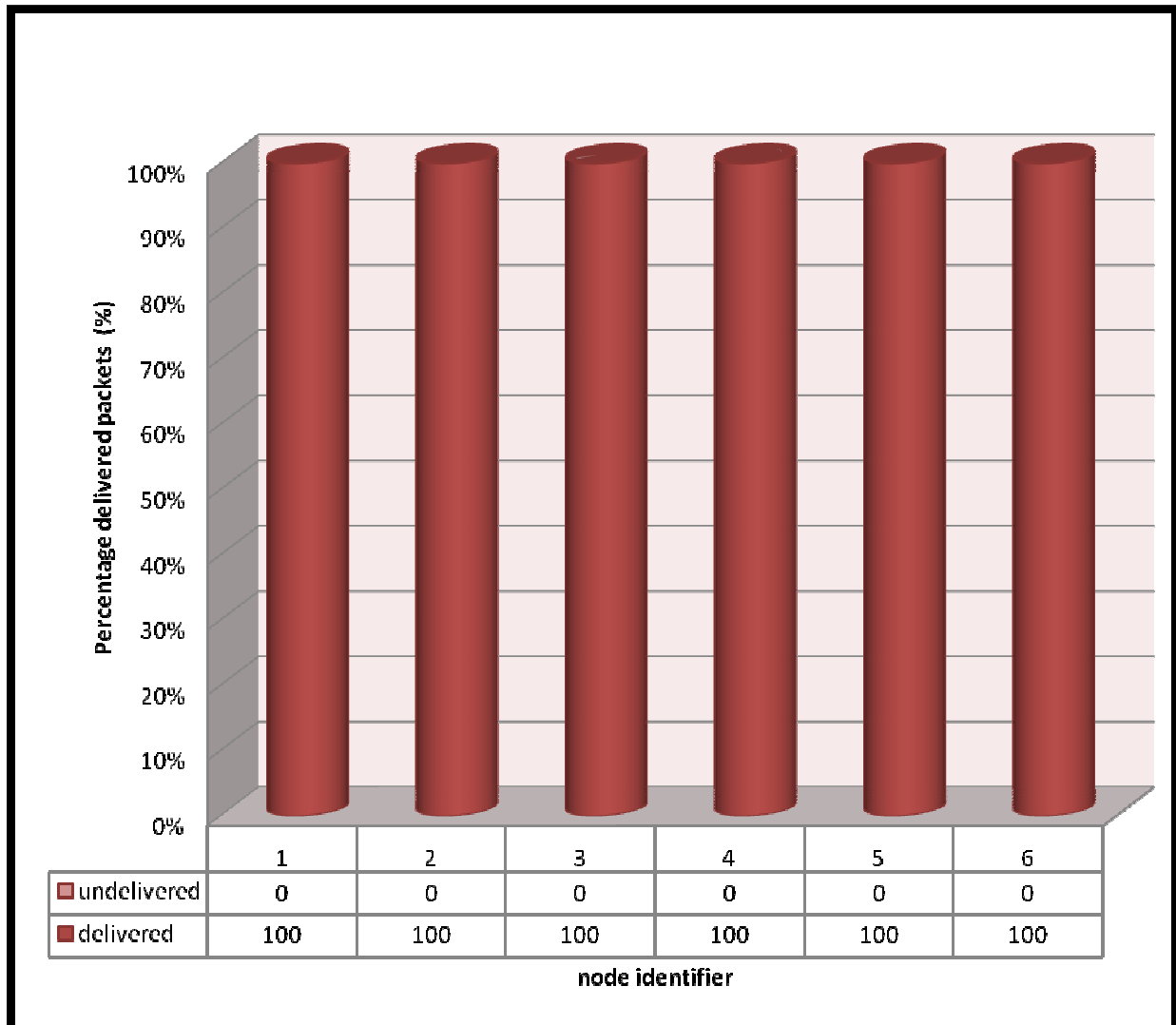
**Figure 8.14 Scalability results of Mica2 nodes at 80 metres**

When nodes are placed at the predetermined 80 meters range apart, disappointing results, shown in Figure 8.14, were obtained, as packet loss was very high. The wireless link dropped most of the packets and very few results were sent back to the base station.

The experiment was repeated with the nodes with placed 40 meters apart, which allowed a node to be able to communicate to two nodes instead of one since the predetermined

## Development of a web-based interface for a wireless sensor network monitoring system

transmission range of the nodes was 80 metres. The results from the experiment were plotted in Figure 8.15.



**Figure 8.15 Scalability results of Mica2 nodes at 40 metres**

The results from the experiment showed that at this distance of 40 metres, no packets were dropped.

The number of wireless sensor nodes available was limited, so TOSSIM, the simulation environment, was used to continue tests on the scalability of wireless sensor networks. The radio ranges of the simulated devices can be set to 10, 100 or 1000 feet as previously shown

## Development of a web-based interface for a wireless sensor network monitoring system

in Figure 8.4. In order to imitate the lossy nature of wireless links, the lossy radio model was used in the simulator.

An experiment was carried out with nodes laid out as shown in Figure 8.13 at distances of 100 feet (33 metres), (10 feet was too small and 1000 feet would result no values being delivered as this distance is too large.) The wireless sensor network did not scale well as shown in Figure 8.16.

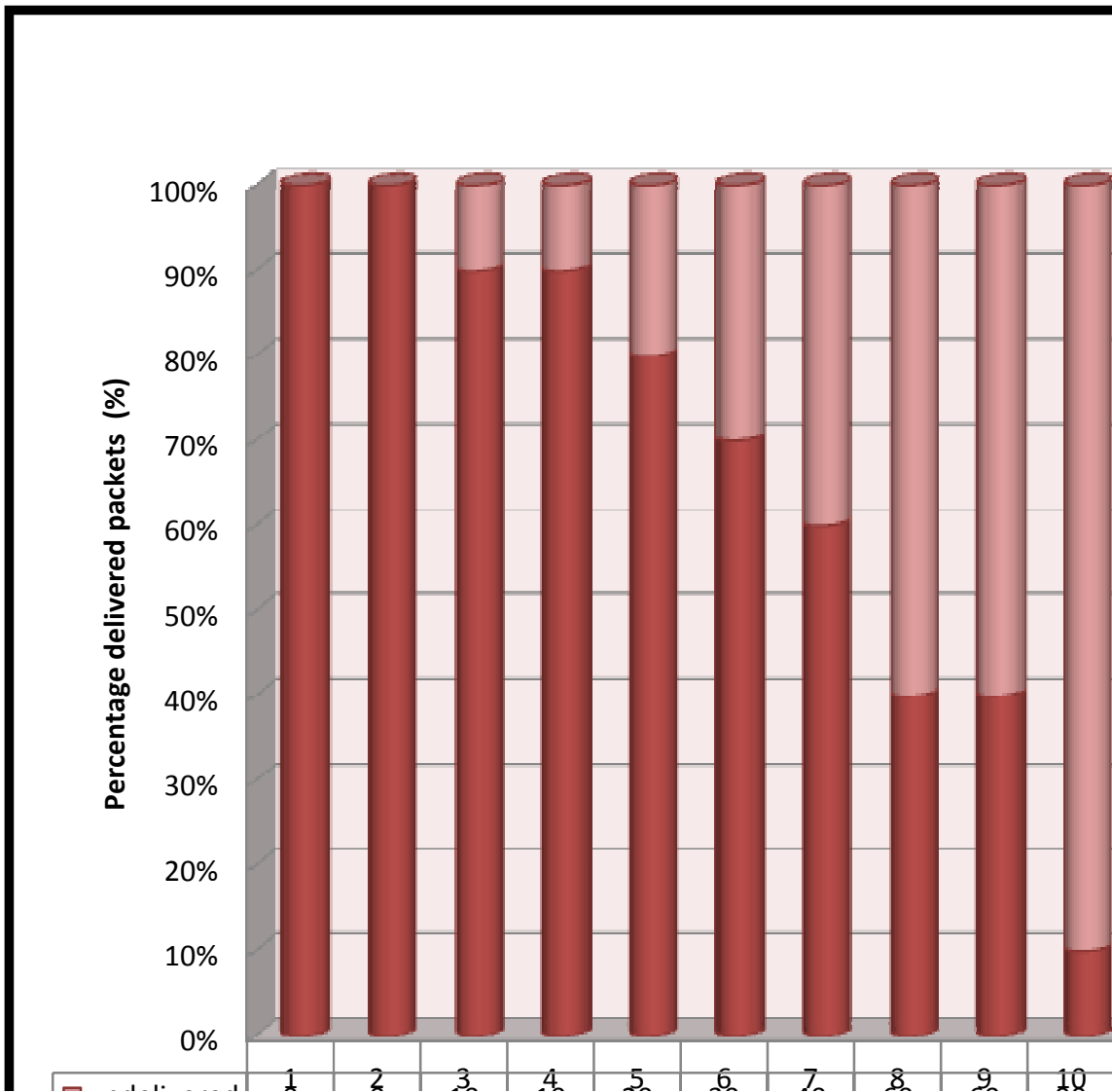


Figure 8.16 Scalability results in TOSSIM at 100 feet (33 metres)

## 8.4. Evaluation of response time in wireless sensor networks

### 8.4.1. Response time of TinyDB in Mica2 nodes

When a wireless sensor network is queried, some time elapses before any results are obtained from the database. An experiment using a single query to all sensor nodes (as shown in Figure 8.17) was used to determine the average time in which responses started to appear in a database from the Mica2 nodes and the TOSSIM simulator, keeping in mind that logged values are time stamped for each sensor node. The time stamp was compared to the time in which the query was issued, and the differences were plotted in Figure 8.18 and Figure 8.19. The Mica2 wireless sensor nodes were placed 40 metres apart to ensure that they all receive the request and all results are received at the base station. The simulator nodes were placed 10 feet apart so that none of the data packets sent from the nodes to the base station are lost.

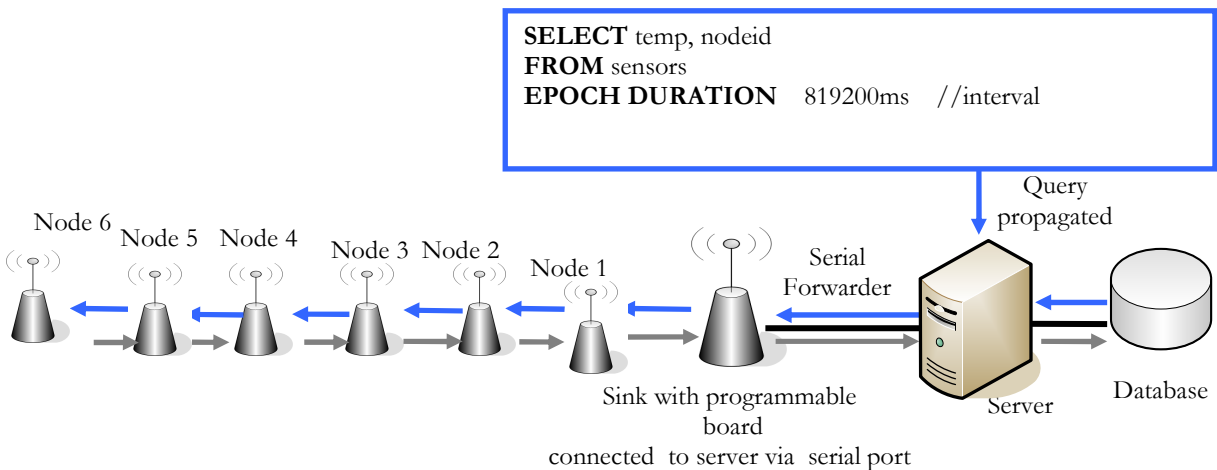


Figure 8.17 Topology for TinyDB response time experiment.

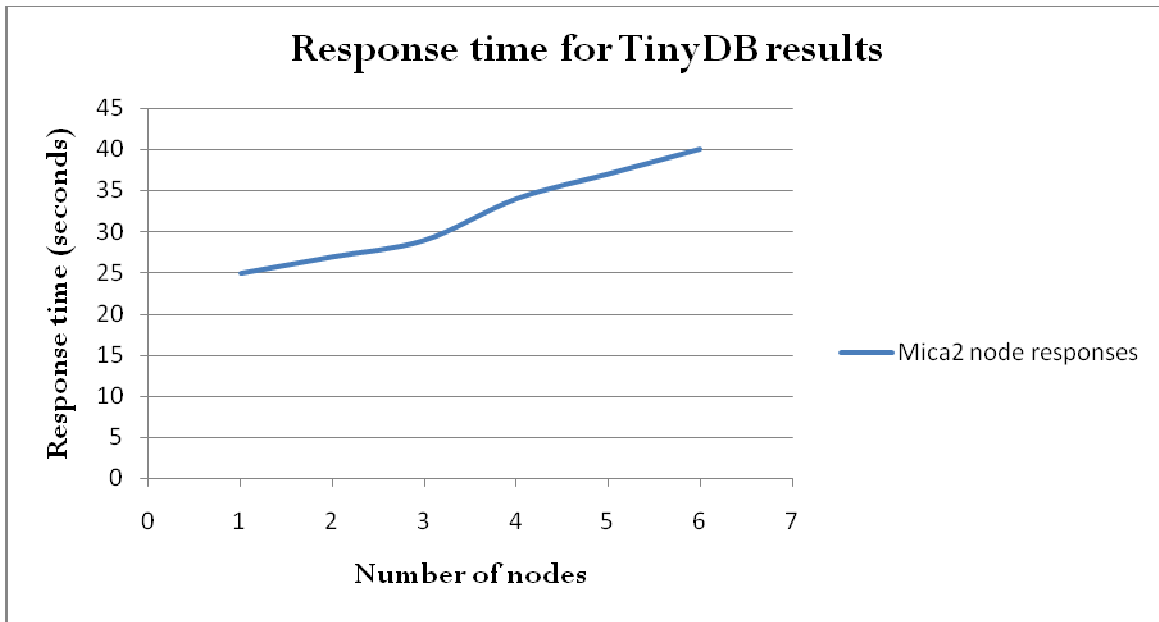


Figure 8.18 Response time for TinyDB results in Mica2 nodes placed 40 metres apart

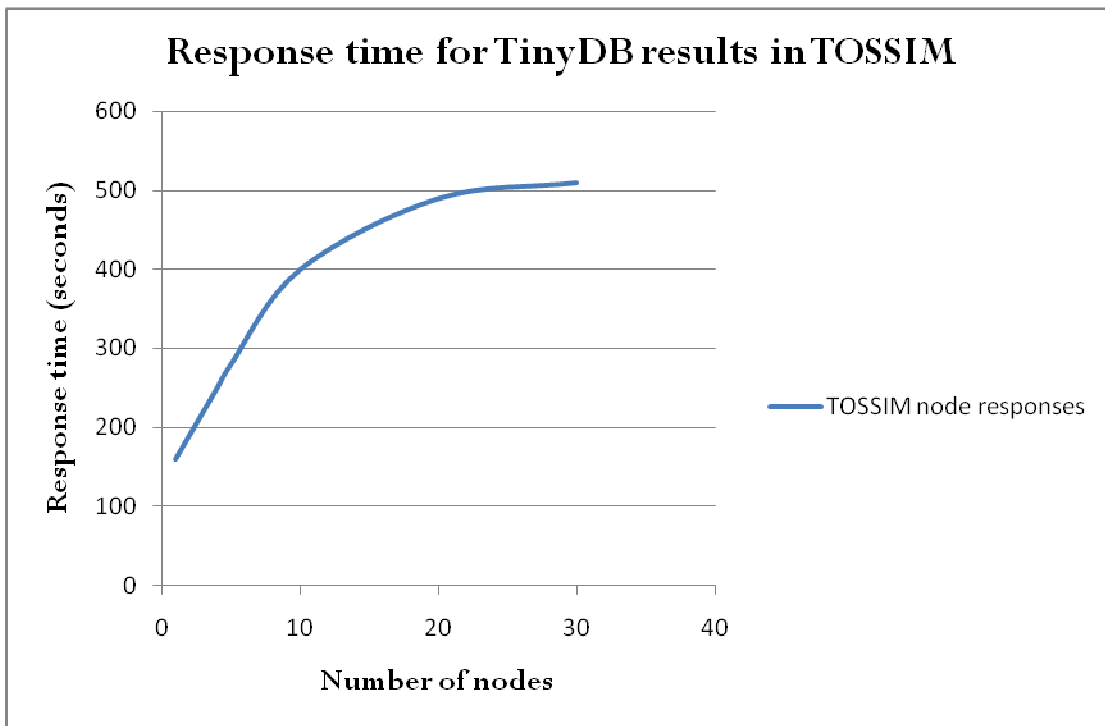


Figure 8.19 Response time for TinyDB results in TOSSIM nodes placed 10 feet apart

## 8.5. Evaluation of event based querying

A network of TinyDB nodes is resource-constrained, so besides continuous querying, event based querying was also evaluated, using a simulator with a layout similar to Figure 8.20, with distances of 80 metres between nodes. Event based querying allows the system to remain in sleep mode until the external condition warrants re-awakening of the devices. This lowers power consumption and improves packet delivery.

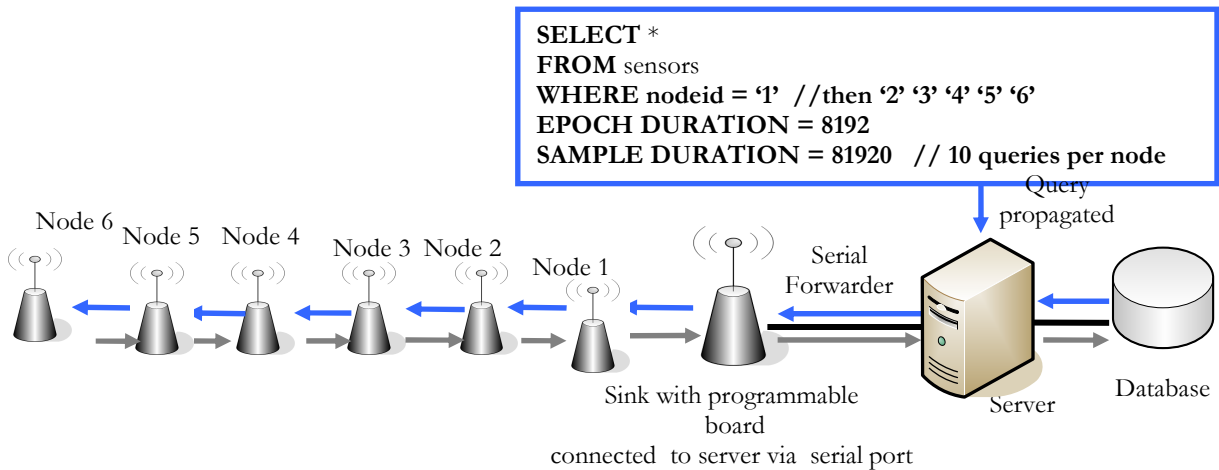


Figure 8.20 Event based querying of sensor nodes

The results from the query were plotted in Figure 8.21. The number of packets were still reflective of the expected packets which the event based query was propagated into the network.

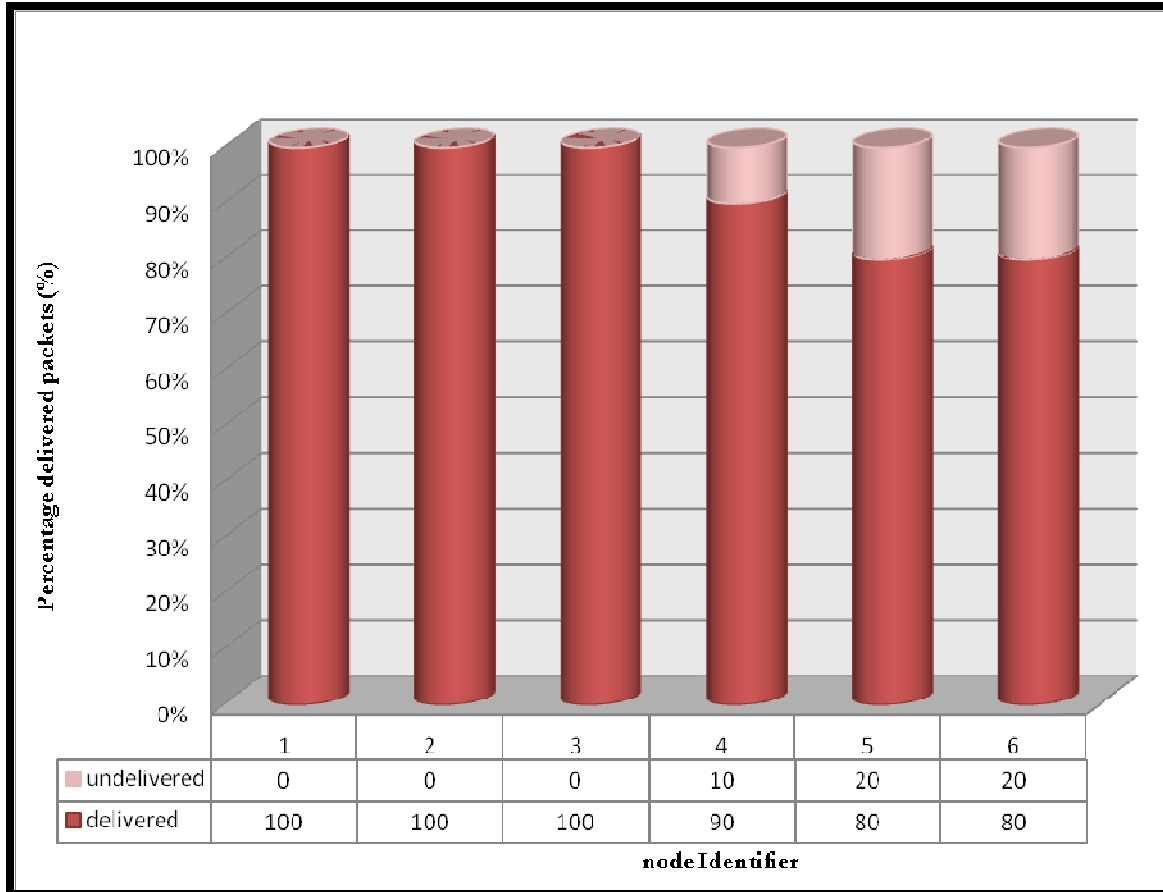


Figure 8.21 Event based query results

## 8.6. Experiment results discussions

The results plotted in all the experiments reflect that wireless sensor networks are resource constrained devices. A linear topology is incapable of scaling many nodes when continuous queries are issued to it, and is unlikely to scale too many either when a event based query is issued to the network. The reasons why a linear sensor network behaves this way is because:

- Packets are lost due to the instabilities of the wireless sensor network,
- The collision of packets as they are transmitted to the base station, results in them being lost enroute to this base station,

## Development of a web-based interface for a wireless sensor network monitoring system

- The nodes in the network are only capable of communicating directly with one or at most two nodes. If the communication link is down between these two nodes, the packets are lost.



## Chapter 9

### Conclusions and Future work

*This chapter summarizes the work undertaken in this project, listing also successes and failures in the deployment of the wireless sensor network. Furthermore, suggestions of further research are outlined.*

#### 9.1. Introduction

This thesis described a system we have built to acquire, store and display data collected from sensors in a linear network, to support work to investigate whether these devices would scale in real- world deployments on long distance transmission lines. This system uses a multi-tier Java 2 Enterprise Edition (J2EE) architecture built on open source platforms. As a consequence, it is capable of running on any operating system that supports the Java runtime environment. The design and relations among the components in this architecture are important as they present the first attempt to design a general framework for structural analysis.

## 9.2. Project Summary

This project involved the deployment of a post processing application to organize and manage overhead transmission line related data retrieved from a wireless sensor network.

The complete design considered as suitable for organizing and managing the data from the sensors consisted of data acquisition tools and a wireless sensor application to the organize data.

Mica2 sensor nodes and the equivalent virtual TOSSIM simulator were chosen to measure and store sensor readings in a temporal database during wireless sensor system deployments, after which the selection of an appropriate sensor querying tool to control the devices was made.

Experiments to evaluate the data acquisition system performances were carried out in order to characterize their hardware capabilities and determine the feasibility in remote monitoring overhead transmission line quantities.

A wireless sensor application was developed to access the temporal database and display the readings with the aid of alert icons and graphs, per user request, on a web interface for easier parameter assessments.

An experimental evaluation of the system was conducted by setting out wireless sensor nodes, issuing queries for data collection and testing the post processing capabilities of the wireless sensor application.

### 9.3. Project Evaluation

The project was successful in providing an in-depth understanding on suitable Java-based open source tools to develop wireless sensor data acquisitions and applications. A post-processing application to display and allow easier analysis sensor data on a web-based interface was developed. Other objectives achieved include:

- Obtaining an in-depth understanding on wireless sensor and associated technologies currently available on the market,
- Identifying, using experiments, the device and media constraints affecting the deployment of a linearly aligned wireless sensor network,

The experiments carried out showed that linear aligned Mica2 and TOSSIM nodes are not scalable. The post processing application developed was able to display the sensor data, but was not suitable for a linear aligned network since the data packets from nodes far from the base station did not arrive at all.

### Future Work

The investigation of the performance of other wireless sensor network topologies is required in future. The post processing application may be useful in monitoring these structures, where the sensor nodes are deployed.

Wireless sensor networks are susceptible to intruder attacks; hence there may be a need to incorporate security mechanisms into this technology because of the sensitive nature of some applications. An investigation into such technologies may be necessary to reduce or eliminate the vulnerabilities in wireless sensor applications. Encryption of data before its transmission is one technique that would ensure security in wireless sensor networks. Protocols like Elliptic Curve Cryptography (ECC), for the encryption of wireless sensor data already exist, but caution has to be exercised in actual implementations because they may have a negative impact on already resource constrained devices (Karlof and Wagner, 2003).

## References

Al-Obaisat, Y., Braun, R. 2007. On Wireless Sensor Networks: Architectures, Protocols, Applications, and Management. University of Technology, Sidney, Australia. Available: <http://epress.lib.uts.edu.au/dspace/handle/2100/160>

Akkaya, K., Younis, M. 2005. A survey on Routing Protocols for Wireless Sensor Networks. Elsevier Ad Hoc Network Journal, vol. 3, no. 3. pp 325-349. Available: [http://www.cs.umbc.edu/~kemal1/mypapers/Akkaya\\_Younis\\_JoAdhocRevised.pdf](http://www.cs.umbc.edu/~kemal1/mypapers/Akkaya_Younis_JoAdhocRevised.pdf)

Akyldiz, I. et al. 2002. A survey on sensor networks. IEEE Communications Magazine, Vol. 40, No. 8, pp 102-116. Available: <http://www.ece.gatech.edu/research/labs/bwn/sensornets.pdf>

Anushruthi, R., Nalini, V. 2005. Implementation and Testing of a Technique for Node Localization in Wireless Ad-Hoc Sensor networks. Visveswariah Technological University, Belgaum. Available:

Apache Foundation website. 2007. Available: <http://tomcat.apache.org/>

Baghdasaryan, Y. 2005. Network Reprogramming for Mica2 Motes. Available: [http://dcg.ethz.ch/theses/ss05/mics-embedding\\_report.pdf](http://dcg.ethz.ch/theses/ss05/mics-embedding_report.pdf)

Bajcsy, P. et al. 2005. Towards hazard aware spaces: Knowing where, when and what hazards occur. Technical Report: alg05-001. Available: <http://algdocs.ncsa.uiuc.edu/TR-20050607-1.pdf>

## Development of a web-based interface for a wireless sensor network monitoring system

Beutel, J. 2005. Design and Deployment of Wireless Networked Embedded System. PhD thesis. Swiss Federal Institute of Technology, Zurich, Switzerland. ISBN 3-832247602

Bharathidasan, A., Ponduru, V.A.S. 2005. Sensor Networks: An overview. Available: <http://www.csif.cs.ucdavis.edu/~bharathi/sensor/survey.pdf>

Bramwell, M. 2006. Implementing a Mica2 mote sensor network. University of Waterloo. Available: <http://hamster.foxhollow.ca/TinyOS/Guide/Mica2-Install-Guide.pdf>

Bouyssounouse, B., Sifakis, J. Embedded computer systems / Design and construction. ISBN 3540251073. pp 398

Callaway, E.H. 2004. Wireless Sensor Networks: Architecture and protocols. CRC Press. ISBN 0849318238. pp 33

Cao, Q. et al. 2006. Efficiency Centric Communication Model for Wireless Sensor Networks. Available: <http://www.cs.virginia.edu/papers/qing-infocom2006.pdf>

Carter, B., Ragade, R.K. 2006. Message Transformation Services for Wireless Sensor Networks (MTS-WSN). International Conference on Wireless Networks, Barcelona, Spain, June 26-29 2006. ISBN 1-60132-000-0. pp 1-12

Cavaness, C. 2004. Programming Jakarta Struts. Second Edition. O'Reilly Media. ISBN 0596006519. pp 9

Cheng, P., Chuah, C., Lui, X. 2004. Energy-aware node placement in wireless sensor networks. IEEE Globecom'2004. Vol. 5, Issue 29, pp 3210-3214. Available: [http://serrano.cs.ucdavis.edu/publications/2004\\_liu\\_2004-12-09\\_10\\_13\\_46.pdf](http://serrano.cs.ucdavis.edu/publications/2004_liu_2004-12-09_10_13_46.pdf)

## Development of a web-based interface for a wireless sensor network monitoring system

CSIR. 2007. SA public to get fire updates on tv. Available: [http:](http://wwwold.csir.co.za/plsql/ptl0002/PTL0002_PGE157_MEDIA_REL?MEDIA_REL)

[//wwwold.csir.co.za/plsql/ptl0002/PTL0002\\_PGE157\\_MEDIA\\_REL?MEDIA\\_REL](http://wwwold.csir.co.za/plsql/ptl0002/PTL0002_PGE157_MEDIA_REL?MEDIA_REL)  
[EASE\\_NO=7388202](http://wwwold.csir.co.za/plsql/ptl0002/PTL0002_PGE157_MEDIA_REL?MEDIA_REL)

Crossbow. 2003. TinyOS Getting Started Guide

Crossbow website.2007. Available : [www.xbow.com](http://www.xbow.com)

Culler, D. et al. 2004. Overview of Sensor Networks. IEEE Computer Society. Available:

<http://www.ics.uci.edu/~dutt/ics212-wq05/ieeecomput-sensornet-aug04.pdf>

Cygwin website. 2007. Available: [www.cygwin.org](http://www.cygwin.org)

Das, A.N., Lewis, F.L., Popa, D.O. 2006. Data-logging and Supervisory Control in Wireless Sensor Networks. Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06). Volume 00. ISBN: 0-7695-2611-X pp 330 - 338

Eskom. 2007. Reduction in water consumption. Available:

[http://www.eskom.co.za/live/content.php?Item\\_ID=2785](http://www.eskom.co.za/live/content.php?Item_ID=2785)

Fernandes, R. et al. 1984. Apparatus for measuring the temperature and other parameters of a electric power conductor. European Patent Application 0125050

FreePatentsOnline. 2004. Method for controlling an electric power transmission network.

European Patent EP1418477. Available:

<http://www.freepatentsonline.com/EP1418477.html>

## Development of a web-based interface for a wireless sensor network monitoring system

Garcia, R. 2006. Understanding the Zigbee stack. Available:

[http://www.eetasia.com/ARTICLES/2006JAN/PDF/EEOL\\_2006JAN02\\_RFD\\_NET\\_D\\_TA\\_01.pdf](http://www.eetasia.com/ARTICLES/2006JAN/PDF/EEOL_2006JAN02_RFD_NET_D_TA_01.pdf)

Gilbert, D. 2004. The JFreeChart Class Library Developer Guide. Available:

<http://www.utdallas.edu/kcooper/teaching/jfreechart-0.9.21-US.pdf>

Gumbo, S., Muyingi, H. 2007. Development of a web-based interface for remote monitoring of a long-distance power transmission overhead line. SATNAC 2007, Sugar Beach Resort, Mauritius. ISBN 978 0 620 39351 5

Gutierrez, A. et al. 2001. IEEE 802.15.4: A developing standard for low-power low-cost wireless personal area networks. IEEE Network. Volume 15, Issue 5. pp 12-19

Hafez, R. 2005. Building Wireless Sensor Networks. Available:

<http://mwrf.com/Articles/Index.cfm?Ad=1&ArticleID=11071>

Hill, J.L. 2003. System architecture for Wireless Sensor Networks .PhD dissertation. Dept. of Computer Science, University of California, Berkeley. Available :

JFreeChart website. 2007. Available: <http://www.jfree.org/jfreechart/>

Karlof, C., Wagner, D. 2003. Securing routing in wireless sensor networks: Attacks and countermeasures. Elsevier's Adhoc Networks Journal. Special Issue on Sensor Network Applications and Protocols, 1(2-3). pp 293-315

Kay, R., Mattern, F. 2004. The Design Space of Wireless Sensor Network. IEEE Wireless Communication 11 (6). pp 54-61

## Development of a web-based interface for a wireless sensor network monitoring system

Kiessling, F. 2005. Overhead Power Lines. ISBN 3540002979. pg 151

Kim, Y. et al. 2005. Implementing a Prototype System for Power Facility Management using RFID/WSN. International Journal of Applied Mathematics and Computer Sciences, Volume 2, Number 2. ISSN 1307-6906. Available:  
<http://www.waset.org/ijamcs/v2/v2-2-15.pdf>

Koubaa, A. et al. 2005. Lower Protocol Layers for Wireless Sensor Networks: A Survey. Available: [www.hurray.isep.ipp.pt](http://www.hurray.isep.ipp.pt)

Kuorilehto, M. et al. 2005. A survey of Application Distribution in Wireless Sensor Networks. EURASIP Journal on Wireless Communications and Networking 2005:5. pp 774-788

Landsiedel, O. et al. 2005. Enabling Detailed Modeling and Analysis of Sensor Networks. Center for Embedded Network Sensing. Available :  
<http://www.cs.ucla.edu/~palsberg/paper/praxis05.doc>

Law, Y., Havinga, P. 2005. How to secure a Wireless Sensor Network. ISSNIP 2005. pp. 89-96

Leskovar, S. 1977. Telemetering post for measuring variables in a high-voltage overhead line . United States Patent 4158810

Lewis, P., Lee, N. 2003. TOSSIM: A Simulator for TinyOS Networks. Available:  
<http://www.cs.berkeley.edu/~pal/pubs/nido.pdf>



## Development of a web-based interface for a wireless sensor network monitoring system

Lynch, J.P., Loh, K. J. 2006. A summary Review of Wireless Sensors and Sensor Networks for Structural Health Monitoring. The Shock and Vibration Digest, Vol. 38, No. 2. pp 91-128. Available: <http://svd.sagepub.com>

Madden, J. et al. 2003. TinyDB: In-Network Query Processing in TinyOS. Available: <http://telegraph.cs.berkeley.edu/tinydb/tinydb.pdf>

Madden, S.R. et al. 2003. TinyDB: An Acquisitional Query Processing system for Sensor Networks. ACM Transactions on Database Systems, Volume 30, Issue 1. pp 122 – 177. ISSN: 0362-5915

Mainwaring, A. et al. 2002. Wireless Sensor Networks for Habitat Monitoring. Proc. 1<sup>st</sup> ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA'02). pp 88-97. Available: <http://www.polastre.com/papers/wsna02.pdf>

Major, J. M. 2006. Ensuring the Health of Our Power Lines. Available: <http://www.swri.org/3pubs/ttoday/Summer06/PoweLines.htm>

Maroti, M. 2004. Directed Flood-Routing Framework for Wireless Sensor Networks. Available: <http://www.isis.vanderbilt.edu/projects/nest/documentation/Middleware-Routing.pdf>

Martincic, F., Schwiebert, L. 2005. Introduction to Wireless Sensor Networking. Handbook of Sensor Networks: Algorithms and Architectures. ISBN: 978-0-471-68472-5

Mayer, K., Taylor, K. 2003. TinyDB by Remote. World Conference on Integrated Design and Process Technology, Texas. Available: <http://mobile.act.cmis.csiro.au/documentation/RemoteTinyDB.pdf>

## Development of a web-based interface for a wireless sensor network monitoring system

Megerian, S., Potkonjak, M. Wireless Sensor Networks. Available:

[http://www.ece.wisc.edu/~megerian/papers/sensor\\_nets.pdf](http://www.ece.wisc.edu/~megerian/papers/sensor_nets.pdf)

Meguerdichan, S. et al. 2001. Coverage Problems in Wireless Ad-hoc Sensor Networks.

Proc. IEEE INFOCOMM '01. pp 1380 – 1378. Available:

[http://www.cs.ucla.edu/~miodrag/papers/Meguerdichian\\_Infocom\\_01.pdf](http://www.cs.ucla.edu/~miodrag/papers/Meguerdichian_Infocom_01.pdf)

Netbeans website.2007. Available: [www.netbeans.org](http://www.netbeans.org)

Nordman, M. 2004. An architecture for Wireless Sensors in Distributed management of Electrical Distribution systems. PhD dissertation. Dept. Elect.Eng. Helsinki University of Technology, Finland. Available: <http://lib.tkk.fi/Diss/2004/isbn9512272989/>

Papageorgiou, P.2003. Literature Survey on Wireless Sensor Networks. Available:

<http://www.cs.umd.edu/users/pavlos/papers/unpublished/papageorgiou03sensors.pdf>

Pathan, A. K. et al.2006. Smartening the Environment using Wireless Sensor Networks in a developing country. Proc. Of 8<sup>th</sup> IEEE ICACT, Volume 1, Phoenix Park, Korea. pp 705-709

PostgreSQL website.2007. Available: [www.postgres.org](http://www.postgres.org)

pgAdmin website.2007. Available: <http://www.pgadmin.org/>

Raghavendra, C.S. et al. 2004. Wireless Sensor Networks. ISBN 1402078838. pp 14-16

## Development of a web-based interface for a wireless sensor network monitoring system

Romer, K. et al. 2002. Middleware Challenges for Wireless Sensor Networking. ACM SIGMOBILE Mobile Computing and Communications Review, Volume 6, Number 2, pp 59-61. Available: <http://www.vs.inf.ethz.ch/res/papers/wsn-middleware.pdf>

Seth, S. et al. 2004. Feasibility of real-time distributed structural control upon a wireless sensor network. Proc. 42<sup>nd</sup> Allerton Conference on Communication, Control and Computing. Available:  
<http://www.personal.umich.edu/~jerlynch/papers/allerton04.pdf>

Stavrou, E. 2006. Wireless Sensor Networks. Available:  
<http://webhosting.devshed.com/c/a/Web-Hosting-Articles/Wireless-Sensor-Networks-pt-1-Introduction/>

Sun Microsystems. Available: <http://java.sun.com/>

Suramanian, M. 2005. Investigation of application deployment of wireless sensor devices in home or business automation scenarios. Masters thesis. Available:  
<http://users.cs.cf.ac.uk/M.Subramanian/Thesis/Mahesh-Thesis.pdf>

Surve, V. 2006. A Wireless Communication Device for Short Messages. Masters thesis. Available: [www.certec.lth.se/doc/awireless.pdf](http://www.certec.lth.se/doc/awireless.pdf)

Thom, J. 2005. Deciphering TinyOS serial packets. Octave Brief #5-01. Available:  
<http://www.octavetech.com/pubs/TB5-01%20Deciphering%20TinyOS%20Serial%20Packets.pdf>

TinyDB article. 2003. TinyDB: A Declarative Database for Sensor Networks. Available:  
<http://telegraph.cs.berkeley.edu/tinydb/>

## Development of a web-based interface for a wireless sensor network monitoring system

TinyOS article. 2003. Serial-line communication in TinyOS-1.1. Available:

<http://www.tinyos.net/tinyos-1.x/doc/serialcomm/index.html>

TinyOS website. Available: [www.tinyos.net](http://www.tinyos.net)

Ye, W., Heidemann, J. 2003. Medium Access Control in Wireless Sensor Networks.

USI/ISI Technical Report ISI-TR-580. Available: <http://www.isi.edu/~weiye/pub/isi-tr-580.pdf>