uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

u Ottawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Jamal Abdo
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Hardware/Software Implementation of a Test-bed for Management of Sensor Networks

TITRE DE LA THÈSE / TITLE OF THESIS

H. Mouftah
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

N. Georganas
CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

A. Boukerche

C.H. Lung

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Hardware/Software Implementation of a Test-bed for the Management of Sensor Networks

Jamal Abdo

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the MASc degree in Electrical Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-25734-0
Our file Notre référence
ISBN: 978-0-494-25734-0

Abstract

The emergence of new technologies in the wireless realm, coupled with recent advancements in the area of Micro Electro-Mechanical Systems (MEMS) sensor technology, have led the way for current and innovative research in the area of sensor networks. The potential applications of sensor networks range from biomedical to health, military, habitat monitoring, environmental, industrial, traffic control, among others, so more and more wireless sensor networks are being integrated into our daily lives.

An implementation of a sensor network test-bed is presented in this thesis. The test-bed is implemented by combining a FPGA-based Nios II Development kit, Cyclone Edition serially connected to a MIB510 equipped with a MICA2 node with ID0. This test-bed can be used for developing tools for monitoring and managing heterogeneous sensor networks. This thesis deals with building a test-bed that consists of a PC which manages the network through a sensor management Graphical User Interface (GUI) and communicates with the sensor nodes through a Base Station.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ADC	Analog-to-Digital Converter
ANMP	Ad Hoc Network Management Protocol
CDMA	Code Division Multiple Access
CRC	Cyclic Redundancy Check
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DSP	Digital Signal Processing
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FSK	Frequency Shift Keying
GPS	Global Positioning System
GUI	Graphical User Interface
HAL	Hardware Abstraction Layer
HDLC	High Level Data Link Control
I/O	Input/Output
IP	Internet Protocol
ISM	International, Scientific, and Medicine
JTAG	Joint Test Action Group
IDE	Integrated Development Environment
LE	Logic Element
LWIP	Lightweight Internet Protocol
MAC	Medium Access Control
MEMS	Micro Electro Mechanical System
MIB	Mote Interface/ Programming Board
MPR	Mote Processor Radio
NesC	Network-embedded-systems C
PPP	Point-to-Point Protocol
PLL	Phase-Locked Loop

RAM	Random Access Memory
RF	Radio Frequency
SMD	Surface Mounted Device
SMP	Sensor Management Protocol
SOPC	System on a Programmable Chip
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver Transmitter
VHDL	Very High Speed Integrated Circuit Hardware Description Language
WSN	Wireless Sensor Networks

Dedication

To my family:

My mother Rola
My father Mohammad
My brother Ahmad
My brother Mahmoud

Acknowledgments

I would like to thank and acknowledge my thesis supervisors Dr. Nicolas Georganas and Dr. Hussein Mouftah for their support, both academic and financial, during my master studies, without which this work would have been impossible.

Since this project was funded by NSERC, I would like to extend my thanks to them as well.

A special thanks to Mr. Roger Montclam and Mr. Alain Stewart for supporting me through the lending of various hardware equipments.

Finally, I would like to express my thanks to my wonderful family for their love and encouragement.

Chapter 1 : Introduction

1.1 Background

Current advances in sensor technologies, coupled with the integration of wireless communications into the high-tech sphere, have made sensor networks the focus of many research projects. The start of the twenty-first century was marked by an ever-increasing integration of technological advancements into our daily lives. The line separating the digital world and the real world is growing thinner by the day. Sensors act as the means by which the virtual world perceives the real world. The future applications of wireless sensor networks are uncountable; they can only be limited by the human imagination. These applications will have a big impact on improving society and increasing public safety, and it is not irrational to expect that in 10-15 years the world will be covered with wireless sensor networks accessible via the Internet [Sta06].

Sensor networks are unique as much as they are common. From one point of view, we can see that sensor networks share many attributes with other networks having the same architecture or communication type. For example, sensor networks can be centralized, distributed, ad hoc, or peer to peer, and their nodes can communicate over wired or wireless media. As a result, a study of sensor networks cannot help but overlap with other areas of research, and covering all aspects of sensor networks and their different types would thus entangle a researcher in a web spanning most of the networking technologies. From another point of view, there are certain characteristics of sensor networks that differentiate them from others, such as the high density of sensor

nodes that can surpass the 20 nodes/m^3 used in some applications and thus lead to the need for appropriate management protocols.

Sensors size can range from as small as a penny, to as big as radar. And sensor networks have as much range in differences as well. We will focus in this thesis on wireless communication among considerably small battery powered sensors in a variety of environments. Taking its characteristics into consideration, this type of network suffers from many limitations such as limited computation, dynamic topology, and power consumption.

1.2 Motivation

The management of wireless sensor networks is somewhat of a new research area, since the focus of researchers in past years was mainly towards power aware routing algorithms, the aim of which was to efficiently route sensor data from sink to base station(without consuming a lot of power from the low-cost and low-power sensor nodes). Other research was aimed towards energy efficient communications protocols. The management of sensor networks is not a trivial task due to the many limitations and unpredictable behavior of the network caused by the randomness of individual node states [Bud01].

None of the commonly used test-beds for sensor networks is efficient for the testing of management protocols, since most, if not all of them, are designed to collect, analyze, and measure various performance parameters. In this thesis, we present a test-bed that will be used to develop tools for monitoring and managing heterogeneous sensor networks. In such a test-bed, upon a user's request, data (i.e. humidity, temperature, pressure, etc.) can be collected from certain sensors (the location of which is specified by

the user) and stored in a MySQL database for performance analysis (i.e. power consumed during a certain period of time, temperature changes, etc.) and keeping of an historical version of the sensing data. This information can be used later at a pre-design stage for any management protocol for sensor network to predict the behavior of sensor nodes.

1.3 Objectives

The objectives of the present work are the following:

- To design and implement a test-bed that can be used for monitoring and managing heterogeneous sensor networks.
- To use the capabilities of Crossbow sensor nodes, by implementing new nesC applications running on top of the real time operating system TinyOS to retrieve sensing information at various time intervals.
- To design and implement a friendly graphical user interface to ease the management of sensor networks.

1.4 Contributions

The main contributions of this thesis include

- The overall design of the test-bed and the selection of its components.
    - Introducing the use of the Nios II development board, along with MIB510, as a base station in order to allow access to heterogeneous sensor wireless networks.
    - The design and development of a friendly user interface with database access in order to view current and historical versions of the sensing data.

- A study of the effect of the transmitter on the power consumption of a sensor node.

1.5 Used tools

In the course of developing and testing our test-bed, many software tools and hardware components were used. The main tools were as follows: Nios II Development kit, Cyclone Edition from Altera; Crossbow development kit; Rentron RF kit; Zeevo Bluetooth kit; and the Eclipse Environment.

1.6 Thesis Organization

This thesis is divided into seven chapters. Chapter 1 gives an overview of the objectives, contributions, and tools used in this thesis. Chapter 2 provides an overview of wireless sensor networks, with a focus on their unique characteristics and the research conducted on their different layers. Chapter 3 presents the state of the art of the management protocols for ad hoc and sensor networks, as well as a review of various test-beds for sensor networks. Chapter 4 gives an overview of the various software and hardware elements used to build our test-bed. Chapter 5 deciphers the hardware and software implementations of the test-bed developed. In chapter 6, we present the various outputs of the testing conducted on the test-bed and simulation of the forming modules. The last chapter, chapter 7, concludes the thesis with the undergoing work and the future steps of this project.

Chapter 2 : OVERVIEW OF SENSOR NETWORKS

2.1 Definition of Sensor Networks:

Wireless sensor networks are collections of sensor nodes, each having the capabilities of sensing, processing, and communicating, which will lead to thousands of potential applications including biomedical, health, military, habitat, environmental, industrial, traffic control, and many others. Each application has its own set of requirements that differentiates it from others. A sensor network is thus application dependent; in other words, great deals of its characteristics rely on the nature of the application.

The generic architecture of a sensor node is represented in Figure 2-1. It consists of four main parts: the sensing unit, processing unit, transceiver, and power distribution unit. There are also two application-dependent parts: the location finding system and the mobilizing system.

1. *Sensing unit*: Can consist of one or multiple types of sensors (i.e. seismic, low sampling rate, magnetic, thermal, visual, infrared, acoustic, radar) that act as a link between the physical and virtual worlds. After the analog data are collected, the sensor will pass them on to the ADC (Analog to Digital Converter) to be converted to digital.

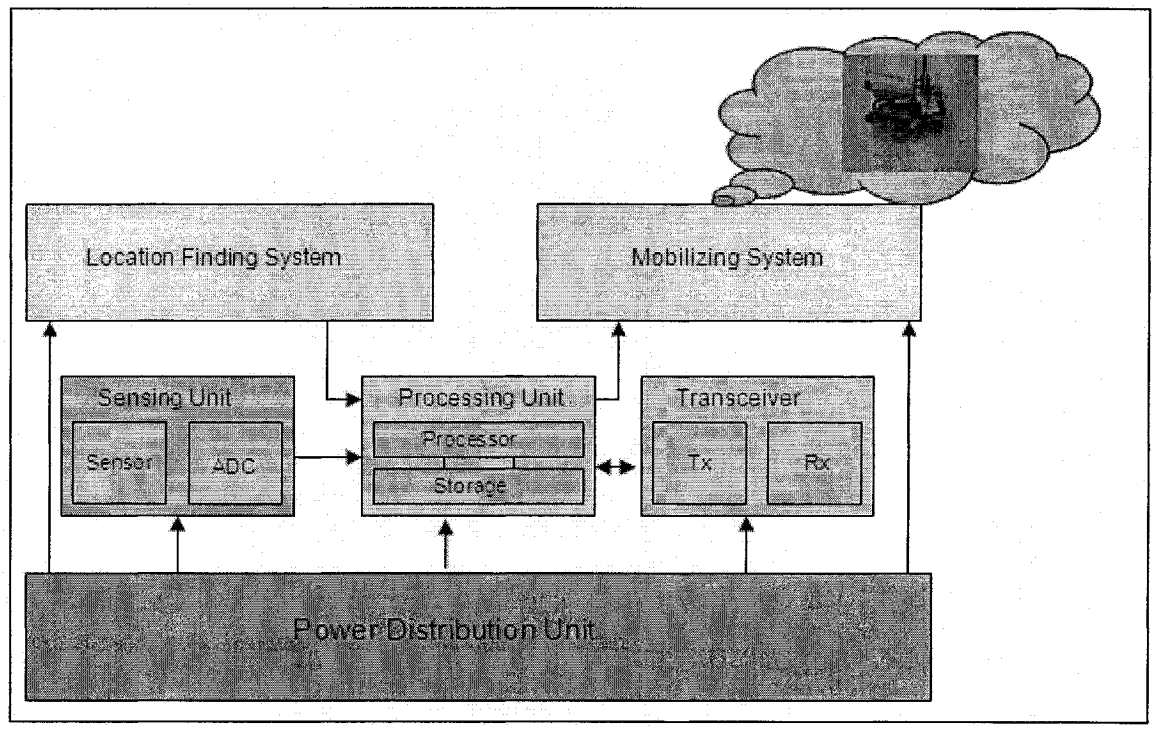

Figure 2-1: Generic architecture of sensor node [Aky02]

2. *Processing unit*: Consists primarily of a processor, which acts as the conductor of an orchestra. It is responsible for controlling the sensor, and for executing communication protocols and signal processing algorithms on the gathered sensor data [Rag04]. A storage unit is also present with limited storage capacity. The most commonly used microprocessors are Atmel AVR, Intel 8051, StrongARM, XScale, ARM Thumb, PowerPC, and SH Risc.
3. *Transceiver*: Is the unit that connects the sensor node to the external world. It sends the processed data over the network channels to the nearest neighbouring sensor node (in the case of a multi-hop communication), or to the base station (in case of a one hop communication). In general, it has a limited range due to constraints in node energy, as well as due to it uses of radio frequency channels (for example 900 MHz) as carrier.

4. *Power Distribution unit:* Is the most important unit in the sensor node's architecture, since the lifetime of the sensor node depends on it. In most cases, assuming that the sensor node is wireless and there is no permanent power generator, the sensor node is powered by 2 AA batteries that can come in three forms: Alkaline, Lithium, and Nickel Metal Hydride.
5. *Location Finding system*: The need for this block arises from the fact that sensor nodes have no IP address (or global identification), due to the large number of sensors used, the restrictions of IPv4 to offer addresses to all these nodes, and the limited processing power insufficient to handle executing the relevant protocol. Thus, the need for location detection emerges as a requirement for routing algorithms and sensing tasks.
6. *Mobilizing system:* It is sometimes needed to move nodes for assigned tasks. This block is mainly used when the sensor node is mounted on robots, as shown in Figure 2-1.

2.2 Various Sensor Node Platforms:

A sensor node that is capable of sensing, processing, and communicating is usually called a "smart sensor node", but the term "sensor node" is used most commonly for the sake of abbreviation. Some sensor node platforms, as well the year of manufacture, are shown in Figure 2-2. As the current trend in most of today's technologies, especially with the advancements in the CMOS fabrication[1], these nodes are becoming smaller and more powerful as the years pass.

[1] For example, Intel is currently producing processors using CMOS at 65 nm.

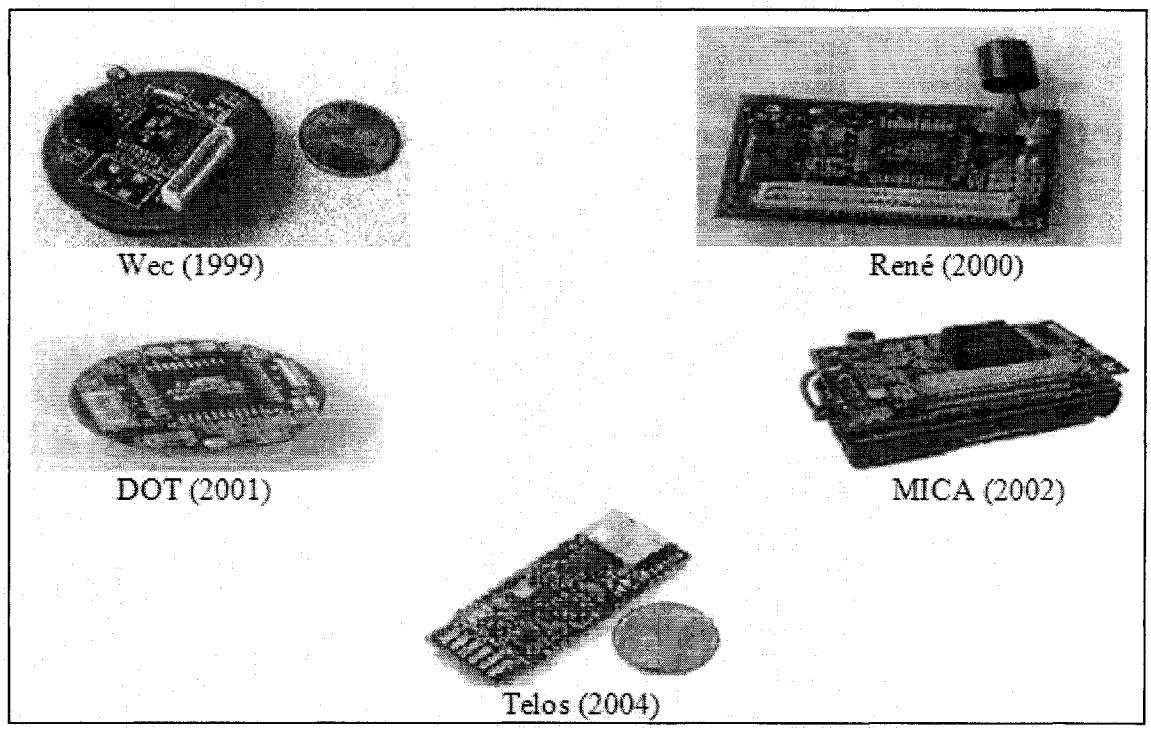

Figure 2-2: Some Sensor node platforms [Aky02]

2.3 Sensor Networks Characteristics:

Now that we have studied the sensor node architecture, we will attempt to explore some of the characteristics of sensor networks, knowing that a full coverage of their characteristics is beyond the scope of this thesis:

Sensors are prone to failure: There are many causes that can lead to a sensor's malfunction; however, they can be narrowed down to two main ones:

- First, the limited power of the sensor restricts its life span considerably, making power the primary cause of failure. Sensors are not usually attached to a permanent power supply since that would limit their mobility, and they thus rely on batteries for energy. After a certain period of time, the battery dies off and the sensor thus becomes useless.

- The second major cause of failure comes from the external environment. A considerable number of applications in which sensors are deployed are marked by hostile surroundings that might steer a sensor into failure. Examples of such types of applications would be military and environmental. Natural elements such as rain, wind, or even an animal in its habitat might render the sensor useless. There is no doubt that when designing a sensor network, a major design factor needs to be fault tolerance. When a sensor dies off, another must take its place and cover its area, to ensure the sensing process continues unaffected. In [Aky02], fault tolerance was defined as "the ability to sustain sensor network functionalities without any interruption due to sensor node failures". The fault tolerance $Rk(t)$ of a sensor node was modeled in [Hob00] using the Poisson distribution to capture the probability of not having a failure within the time interval $(0,t)$:

$$Rk(t) = e^{-\lambda k t},$$

where λk is the failure rate of sensor node k and t is the time period.

Many approaches were suggested to increase the fault tolerance of sensor networks to node or link failure, but we will only explore here the approach that acts at the application layer: PEAS (Probing Environment and Adaptive Sleeping). PEAS is based on the fact that sensor nodes operate in high volumetric densities, more than the required number to guarantee coverage of the area of interest. Thus, the redundant nodes may be put into sleep mode and become active when the neighboring node fails. This approach consists of two simple components. The first is probing environment, where each sleeping node wakes up after a period of time determined by the probability density function to check if there is any working node in its area (circle with radius R_p, where R_p

is the probing range and can be adjusted to satisfy the application requirements). The second is adaptive sleeping, that adjusts dynamically the probing rate λ based on the number of the surrounding sleeping nodes. It is worth noting that this approach has many disadvantages; first, it is based on an unrealistic assumption that all the sensor nodes have the same sensing range, which actually depends on the nodes' remaining energy. Another example is that it is difficult to predict node lifetime, etc.

Sensors are usually inaccessible: Whether used for military, environmental, or any other kind of application, sensors do not have the privilege of being directly accessible to humans, and are thus not repairable or rechargeable. Most of these sensors are embedded within systems or located in unreachable locations such are behind enemy lines or in remote habitats. In [Mai02], inaccessibility was considered as an important feature in habitat monitoring. An increasing concern about the potential impacts of human presence while monitoring plants and animals in field conditions has led to the consideration of sensor networks as an alternative. Human disturbance, according to [Mai02], "may distort results by changing behavioral patterns or distributions, while at worst anthropogenic disturbance can seriously reduce or even destroy sensitive populations by increasing stress, reducing breeding success, increasing predation, or causing a shift to unsuitable habitats". Regarding plant populations, [Mai02] further claims that human disturbance can lead to the introduction of exotic elements and changes in local drainage patterns through path formation. Thus, sensor networks are considered a significant advancement over traditionally invasive methods of monitoring. In short, whether inaccessibility is a desired or undesired characteristic for an application, the fact remains that a sensor must be designed to be as autonomous as possible.

Sensors are becoming less expensive and more redundant: As we all know, the trend in today's high tech society is to make devices smaller and cheaper. Sensors are following that trend as well. Their low cost has given rise to the large numbers in which they can be used. However, the more sensors are used in large numbers, the more traffic is generated. Thus, the trade-off for the increase in the number of sensors is the need to implement further means for processing their information. More sensors means more data, and more data means more traffic. The addressed characteristic also brings into focus the design issue of a sensor network is scalability. Depending on the application, the number of sensors can vary from hundreds to thousands. A key term used in literature on sensor networks is their *density*. In [Aky02], density was defined as the number of nodes within the transmission radius of each node in a region, and was modeled as:

$$\mu(R) = (N. \pi .R2)/A$$

where N is the number of scattered sensor nodes in region A, and R is the radio transmission range.

Sensors are usually dynamic: Sensor networks have a dynamic nature. This is because sensors are mounted on mobile nodes that traverse an area in order to widen the area of surveillance or discovery. Whatever is their purpose, the fact remains that a dynamic topology has its various challenges. As such, sensor network routing becomes a major issue since nodes are constantly breaking the links between them and forming new ones. This means that routing tables need to be constantly updated, thus placing more load on the network bandwidth. Also, there needs to be a location detection mechanism, whether it is GPS or otherwise. In short, this characteristic entails many limitations on the sensor network.

Energy Efficient: As in all mobile devices, power consumption is an important issue in sensor networks since the power supply of a sensor node (a small battery pack in most of the cases) is usually limited. Replacing these batteries is sometimes almost impossible due to the unreachable locations of the nodes and expensive methods of replacement; since nodes in a sensor network are used in very large numbers (i.e. density can surpass the 20 nodes/m^3 used in some cases [Aky02]). This limited power consumption makes it mandatory to use proper power management in order to maximize the lifetime of the nodes, and, consequently that of the sensor network as well. Many solutions have been suggested for this issue such as the usage of energy scavengers (i.e. solar cells, vibration scavengers), reducing the power consumed in each component of the sensor node, redundancy exploitation, and the usage of compression techniques based on the fact that "the communications power is twice greater than the computational power".

Security and Interference: Since the information exchanged between the sensors and the processing station is privileged, and since sensor networks are prone to multiple security threats including impersonation, malicious nodes, and eavesdropping, proper authentication and confidentiality mechanisms must be applied.

Limited computation and data storage: There are three general limitations that exist for sensor networks that pose enormous challenges. The limited computation and data storage of a sensor node, coupled with the limited bandwidth of a wireless sensor network, formulate major obstacles in the processing of sensing data. Sensors in general have the task of collecting data and then sending them to a central station after initial processing. An important issue is deciding the amount of processing required for the raw data gathered. On one hand, sensors do not enable the extensive processing of data due to

size and resource restrictions. However, sending raw data, without deciding what is needed and what is not overloads the network with traffic and overutilizes the bandwidth. Despite this drawback, sending raw data limits the amount of error in the sensing since less information is lost when communicating information at a lower level (e.g., raw signals). However, collaborative processing among sensor nodes leads to more precision in the results and avoids exhausting the network with duplicate results. In the following we will present two approaches to the processing of sensing data; the centralized and the localized (distributed) approach:

- *Centralized Algorithms*: The centralized approach follows the simple architecture of having the sensor nodes communicate directly with the base station, where all the processing of the data is done. This approach limits the amount of error in the data transferred, which is relatively high in wireless communication due to jitter and the variable and dynamic nature of bandwidth, since in transferring raw data less information is lost. Also, processing power and storage constraints limit the amount of computations that may be done on the sensor node itself. Centralized algorithms, however, are undesirable because of high communication costs and lack of robustness and reliability. The amount of bandwidth utilized in transferring raw data exhausts the network. In addition, power dissipated during transmission overshadows that of processing since the transceiver is the greatest source of power consumption in the sensor. There is also a definite scalability issue when considering the centralized approach due to the reasons just addressed, and this is of great concern when dealing with sensor networks in which the quantity of nodes can reach the thousands.

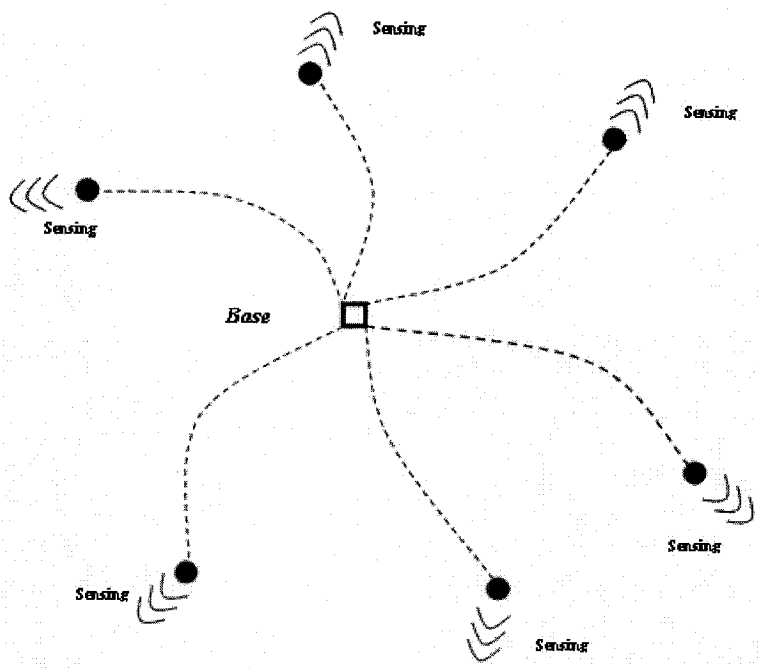

Figure 2-3: Centralized Algorithms

- *Localized Algorithms (Distributed)*: In this approach, sensor nodes only communicate with sensors within a neighborhood. Information is passed from one node to another where data fusion occurs before it is finally sent to the base station. In [Kum03], this type of algorithm was referred to as the Chain Based Algorithm, in which sensors form a chain by communicating with their closest neighbors, fusing data as it is transferred from one node to another, and eventually transmitting it to the base station. An example of this approach would be PEGASIS which employs a chain-based algorithm, for load balancing and minimizing the expenditure of energy. In this algorithm, the nodes take turns transmitting data to the base station. In the following diagram, an illustration of the localized algorithm is given:

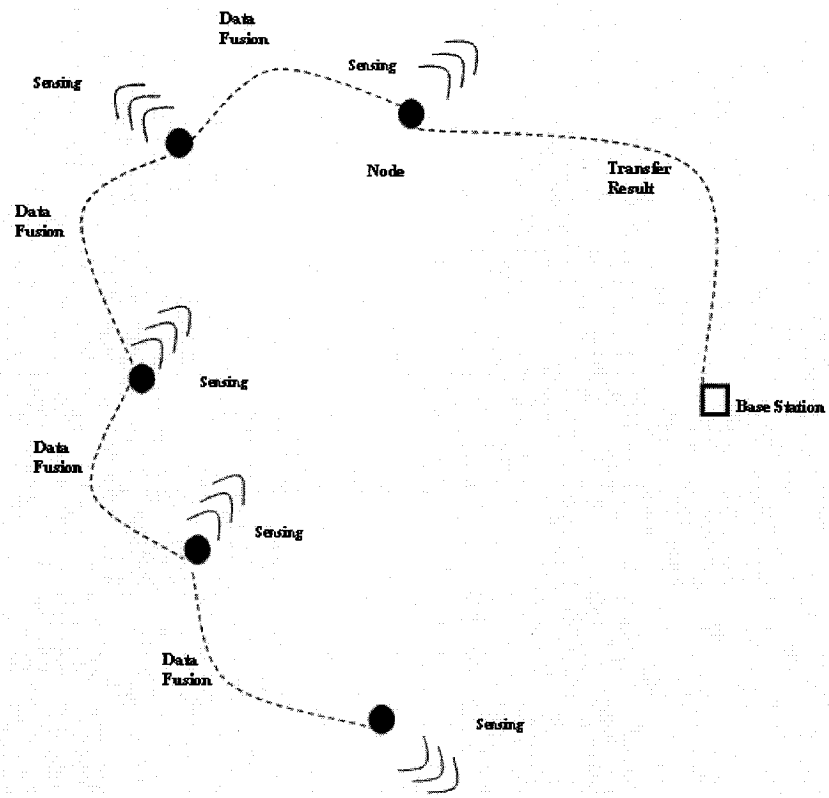

Figure 2-4: Localized Algorithm

Localized algorithms have the advantage of reducing bandwidth utilization since not as many raw data are being sent. They are also robust to network changes and node failures. In addition, scalability does not affect communication considerably. The problem, however, is that localized algorithms are difficult to design because of the collaboration level required between a set of sensor nodes. Also, more computational and storage capacities are required per node, which means more resources and thus higher costs [Cho03].

In the following, we have tabularized a comparison between the localized and centralized algorithms, giving their various advantages and disadvantages.

	Advantages	Disadvantage
Localized	Less Bandwidth utilization	Employs complicated algorithms
	More Scalable	More information loss More resources per node
	Less power consumption	Node collaboration is required.
Centralized	Less resources per node	More bandwidth utilization
	Less information loss	More power consumption
		Less Scalable

Table 2-1: Localized vs. Centralized

- **Clustering Algorithms:** Clustering allows the division of the network into smaller parts for better management and control. Among a set of sensors, one node is selected as a cluster head and all the nodes in its vicinity communicate with it. The cluster head acts as a central point for data aggregation and local decision-making. If the choice of the cluster head is permanent, [Yao02] suggests that it should be dynamically maintained in case of sensor or link failure, and that its physical position must take into consideration the cost of data delivery on bandwidth and power. [Kum03], however, proposes the use of a Low Energy Adaptive Clustering Hierarchy (LEACH) clustering protocol that utilizes the randomized rotation of local cluster heads to evenly distribute the energy load among the sensors in the network. In the following diagram, we explain the clustering algorithm:

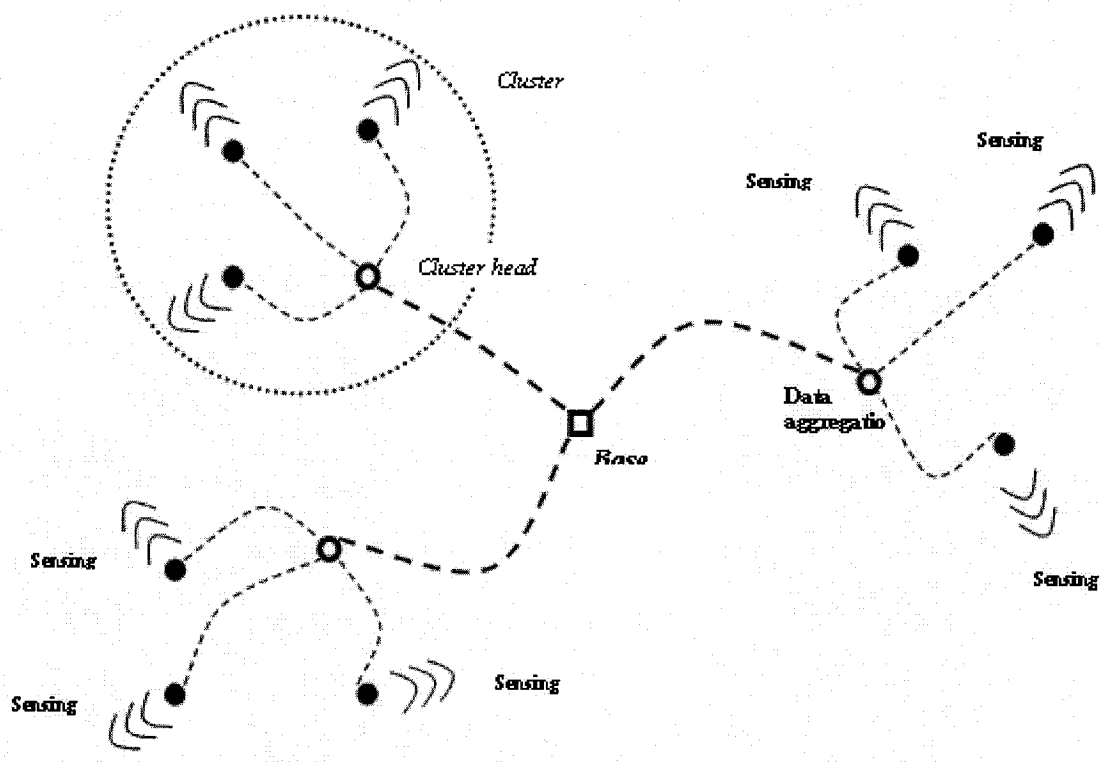

Figure 2-5: Clustering Algorithm

The advantage of this approach is that it allows for the semi localized processing of data. There is no need for sophisticated schemes for data fusion as in the case of pure localized algorithms, yet it requires less bandwidth and power utilization than the centralized approach.

There also exists another way of looking at clustering algorithms. Hierarchal clustering employs intermediate nodes in order to transfer messages from a node to the base station. Data aggregation is done along the way as it moves from one node to another. The network is divided into levels. As we go from level to another, more aggregation is done. The first level of the network would be a collection of peripheral sensors, and the last would be the base station. The following diagram illustrates the approach:

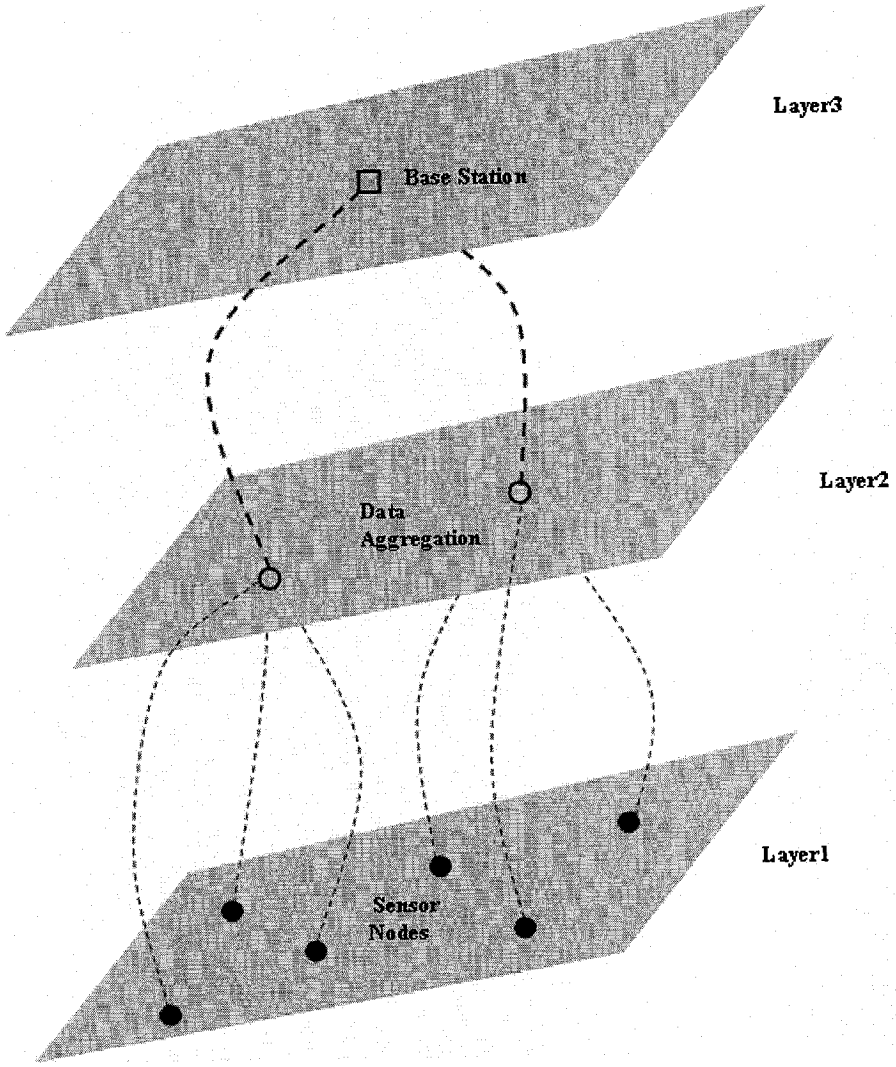

Figure 2-6: Hierarchal Clustering

Self-Healing and Self-Configuring: The network should have the ability to add or remove nodes without resetting the network and be able to establish communication between the nodes without human intervention.

2.4 Sensor Network Protocol Stack

Figure 2-7 shows the different layers used by the sensor nodes. It consists of the following:

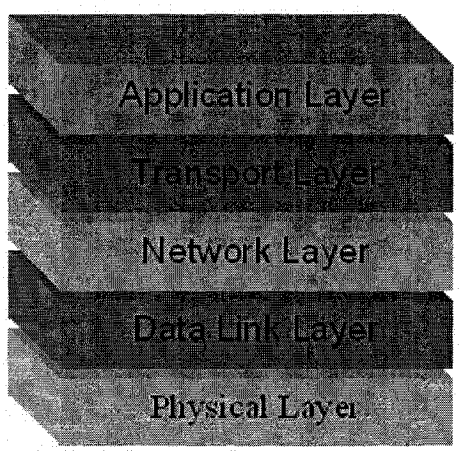

Figure 2-7: Sensor Networks protocol stack

Physical Layer: The communication links between nodes can be radio, infrared, or optical; however, the preferred or most suitable transmission medium for sensor networks is RF. The latter is characterized by small packet size, low data rate, an increase in frequency reuse resulting from the short communications range, and finally by its operation in the ISM band which offers a free and huge allocation spectrum as well as global availability. On the other hand, infrared communication has many advantages such as a cheaper transceiver, easiness of use, and robustness to interference from electrical devices. However, this comes at the expense of having only a line of sight (LOS) capability, which is not possible for most sensor network applications since they operate in harsh environments full of obstacles. The optical fibre medium needs fibre connections between sensor nodes, resulting in an increase in the overall network cost especially due to the fact that E-O-E converters are expensive.

The physical layer has the same role as in computer networks, involving frequency selection, carrier frequency generation, signal detection, modulation, and data encryption.

Data Link Layer: The goal of this layer is to create the network infrastructure and share communication fairly and efficiently between the nodes in the network. The existing

MAC protocols in the cellular system and mobile ad hoc network are not efficient enough to be used in sensor networks, due to the fact that power consumption in the first two networks is of secondary importance, whereas it is a major limitation in sensor networks. Some protocols were proposed, such as:

- *Self-Organizing Medium Access Control for Sensor Networks (SMACS) and the Eavesdrop-And-Register Algorithm (EAR)* [Soh00]**:** SMACS is responsible for establishing communication links (or a connected network). SMACS reduces the probability of collisions; each link operates on a different frequency where the frequency is chosen from a large pool of possible choices when the link is formed. The EAR is only needed when mobile nodes are introduced into the network.
- *CSMA-Based Medium Access*: It is based on constant listening periods and the introduction of random delay.

Network Layer: This layer is responsible for routing the data from source to sink. Many different approaches exist, each of them suitable for a specific application. In what follows, we will explore briefly some of the existing routing protocols:

- *Directed Diffusion:* Is a query/response method used to propagate information through the network. Information required is named by a set of attributes that will be called "interest" and can take the following form:

Type = four-legged animal
Interval = 20 ms.
Duration = 10 s
Rect = [10, 30, 50, 200]

An interest is flooded first at low rate from the sink[2]. The latter will re-send the interest periodically to refresh it and to overcome the unreliability of the wireless links transmission but with increasing timestamp. Every node keeps track of the received interests by saving the task description in its cache (except the information about the sink). When sensing an event, the node searches in its cache for an identical interest. If an identical interest is found, the node (called "source") sends information about the interest to the node from which it receive the event.. If no matching interest is found, the node will simply discard the event. Each node will mark the interest received by a gradient (usually the number of hops) so that when the message arrives at the sink, it will be able to reinforce the best path[3] (the one that offer an empirically low delay path or higher data quality) based on the gradient parameter.

- *LEACH Low Energy Adaptive Clustering Hierarchy [Hei00a]:* Each node in the sensor field decides independently of other nodes in the network whether to be a cluster head based on a random number k (between 0 and 1) generated by each node n. If k is less than a threshold T(n), the node n will elect itself as a cluster head:

$$T(n) = \begin{cases} \dfrac{p}{1 - p * (r \bmod \frac{1}{p})} & \text{Where } n \in G \\ 0 & \text{Otherwise} \end{cases}$$

P is the desired percentage of cluster heads.

[2] The node requiring the data.
[3] Usually multiple sources send information about the same interest.

r is the current round.

G is the set of nodes that have not been cluster heads in the last $\frac{1}{p}$ rounds

After $\frac{1}{p}$ -1 rounds, the nodes that were not yet elected as cluster heads will automatically have T = 1. The nodes elected as cluster heads will broadcast an advertisement message to the rest of the nodes in the field, using the CSMA MAC protocol.

Each node will choose the cluster that it will belong to based on the minimum cost of communication between the node and the cluster head, and it must inform the cluster head that it will be a member of its cluster also using the CSMA MAC protocol.

After the clusters are set up, the cluster heads creates a TDMA schedule, as seen in Figure 2-8, to tell each node within their clusters when it can transmit and broadcast back to the nodes, so the nodes can transmit their collected data during their slots[4] to the cluster head that will perform signal processing functions to compress the received data and then send them to the base station.

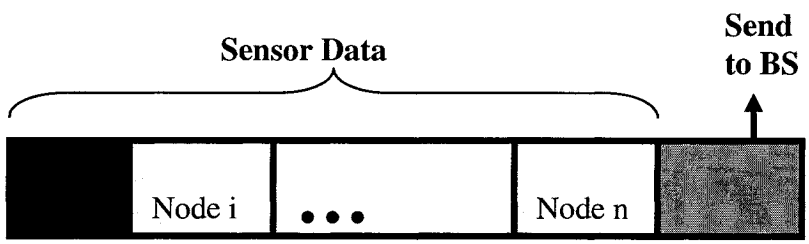

Figure 2-8: TDMA schedule broadcasted by the cluster head

[4] The non-cluster heads can turn their transceivers off after sending their collected data to the cluster head.

In order to avoid intra-cluster communications interference, every cluster communicates with a different CDMA code that will be picked up randomly from a list of spreading codes.

- *SPIN Sensor Protocols for Information via Negotiation [Hei99b]:* SPIN is an intelligent flooding technique that is data-driven and based on the negotiation of the data that the node possesses before the data is sent. It takes into consideration three observations:
    - The exchange of sensor data is expensive, but the exchange of meta-data describing the sensor data is not.
    - Sensors are able, by design, to monitor their own resources.
    - Nodes in close proximity can have similar data, so there is a need to disseminate the data the other nodes do not possess.

Assume x is the meta-data descriptor for sensor data X, so the size of x in bytes should be less than the size of the actual data for the SPIN to be beneficial. The format for the meta-data depends on the application, so it is left open for the designer of the network.

When a SPIN node has data that it is willing to share with other nodes, it sends an advertisement message (ADV) containing a description of the data. The nodes wishing to receive this data will reply with a request message (REQ) listing all the data that they wish to receive (no redundant data are sent into the network), and the source node will then send the actual data with a meta-data header to the other nodes.

When the node receives the data, it aggregates it with its own data (if it has any) and it then advertises it at its turn to all its neighbors except the source node from which it received the data. If a node does not reply with a REQ message, this means it already has the data or its resources are not enough to complete the three stages of the protocol (ADV-REQ-DATA).

SPIN-1 is considered a lossless network since it does not take into consideration the loss of the ADV or REQ message; however, it can be easily adapted to work with loss networks by re-sending the ADV message or sending a message requiring the data in case of timeouts.

Transport Layer: This layer is needed when accessing the Internet or other external network. Some protocols have been implemented for sensor networks since the TCP and UDP protocols are inefficient for sensor networks due to their large overhead. From these protocols we mention the following:

- *Reliable Multi-Segment Transport (RMST):* It is designed to work with Directed Diffusion at the network layer.
- *Sensor Transmission Control Protocol (STCP):* This is a generic protocol, independent of the application.

Application Layer: Application layer for sensor network is a largely unexplored area. In chapter 3, we will explore some application layer protocols.

Chapter 3 : State-of-the Art of Management Protocols and Test-beds for Sensor Networks

3.1 Introduction

This chapter will be devoted to providing an overview of past and current research on management protocols and test-beds for sensor networks. The management of wireless sensor networks is a relatively new area of research, so few protocols have been proposed to manage the large numbers of nodes scattered throughout the networks. Also, the specific attributes and characteristics of sensor networks, explored in chapter 1, differentiate them from other wire-line and wireless networks. Thus, common management protocols (such as SNMP) are not efficient for sensor networks.

3.2 Management Protocols for Ad Hoc and Sensor Networks

Due to shared characteristics[5] between ad hoc and sensor networks, we will first introduce the main two management protocols for ad hoc networks, ANMP(Ad Hoc Network Management Protocol) and Guerilla, as well how these protocols can be modified in order to be suitable for sensor networks. The authors in [Che99] presented ANMP, which they claim provides a complete solution to the management problems of ad hoc networks. It focuses on three categories of management protocols: functionality, data collection, configuration/fault management, and security management.

[5] Major similarities between ad hoc and sensor networks are dynamic topology, limited computational resources, wireless communication, and battery-based devices.

1) Data Collection: Some additional information beyond that collected by SNMP needs to be collected in ad hoc networks, such as battery power, link quality, current location and speed. A new group called anmpMIB has been added to the standard MIB-II of the SNMP protocol. AnmpMIB contains subgroups: power usage, topology maintenance, graphical clustering, mobile hosts, agents' information, power information and events.
2) Configuration/fault management: In contrast to those in the wire-line networks, faults in ad hoc networks should not affect the functionality of the network. Faults can be caused by nodes running out of battery, power themselves off to save energy, or moving geographically out of range. Also, new nodes that join the network should be integrated seamlessly, partitions should be handled, and a manager should be selected for each partition.
3) Security Management: Wireless networks are very susceptible to intrusion, "as they operate in an open medium, and use cooperative strategies for network communications" [Kac02]. They are prone to multiple security threats such as impersonation, malicious nodes, and eavesdropping, and thus proper authentication and confidentiality mechanisms must be applied. In addition to these events, data collection is achieved using a spanning tree algorithm that will cause a security breach if upper level nodes have a higher security clearance. That being said, the data should be encrypted in such a way that the upper level node cannot read it. ANMP uses the multicast secure transmission introduced in [Chi89], in which the Chinese Reminder Theorem (CRT) is used to construct a secure lock that can only be opened by the targeted destination.

ANMP is architecture uses a three-level hierarchical approach in which the lowest level, called agents, represents the sensor nodes. Each agent in the field maintains a MIB where it stores the information that the manager requires, and sends it periodically or upon request to a higher level node, represented by the cluster head. At its turn, the cluster head prepares a summary of the data or simply concatenates the collected data from all the agents in its cluster and sends it to the network. Two algorithms for forming and maintaining clusters are presented in [Che99]: Graph-Based clustering and Geographical clustering. An important note is that if the routing protocol used is cluster-based, ANMP can simply use these clusters for management. If it is not cluster-based, however, one of these two techniques can be adopted:

- Graph-Based Algorithm: This approach views the ad hoc network as a graph in which each node in the graph represents a mobile host in the network. The cluster head is chosen based on the node ID; the node with the minimum ID among its neighbors forms a cluster and becomes the cluster head. Each node considers only its one hop neighbors while forming the cluster. After cluster formation, each node maintains a neighbor list, a cluster list, and a ping counter. The Graph-Based clustering maintains clusters in the face of mobility. There are two cases to be handled:
    - When a mobile node moves out of its cluster and gets disconnected from all the members of its previous cluster, it sends a join request to the nearest cluster head (even if it is two hops away from it). When a cluster head receives a join request from a node, it adds the

node to its cluster, and sends a broadcast message to all the members of its cluster.

- When a cluster head moves away from its cluster, the members of this cluster initiate a cluster formation. The departure of the cluster head can be detected using a ping counter[6] by each node: the counter is incremented at every time step. When the counter crosses a certain threshold, the cluster head sends a message to all its members indicating that it is still alive. If the cluster doesn't hear from its cluster head after this threshold is crossed, it assumes that the cluster head has moved out or is dead.

- Geographical Clustering uses GPS to divide the network into clusters. The network is first divided into rectangular boxes in which the cluster head is the most centrally located node within each box (i.e. as close to the center as possible or having the most available power). Two algorithms run to achieve the final set of clusters; the first one generates the rectangular boxes by splitting the network into ***n*** (n represents the length of the edge of the box/average transmission range) horizontal and vertical strips. Afterwards, three steps are executed: 1) A refinement of the boxes is required so the algorithm splits evenly all boxes that are larger than the range of three transmissions or contain more nodes than the maximum allowed number of nodes within one box (MaxN) 2) deletes the empty boxes and 3) merges the clusters that contain a number of nodes less than a certain threshold (MinN) with a neighboring cluster. To maintain the

[6] A fixed interval when each node creates a list of events that it has observed and sends it to its cluster head.

cluster in the face of mobility, another algorithm runs between the periodic "runnings" of the first algorithm. If a node departs from a box, it broadcasts a message to all of its neighboring nodes in order to join another box. After joining a new box, the cluster head counts the number of nodes in its cluster; if it is greater than maxN, it splits the box evenly and the cluster head of the previous cluster counts the number of nodes in its cluster. If the number is less than MinN, it sends a merge request to a neighboring node. Now, if a cluster head departs from its box, it informs up to three nodes of its cluster and one of these nodes is selected as a cluster head (based on its location , battery power remaining, etc.)

The message cost of maintaining clusters is less in geographical clustering than in Graph-Based clustering, but the percentage of nodes unmanaged by cluster heads in graph-based clustering is higher than those unmanaged by cluster heads in geographical clustering.

ANMP also has a graphical user interface that provides a friendly means to manage the network. The interface provides a graphical representation of the network topology, and indicates the remaining battery power for each node. The manager is able to create SNMP-PDU s using the query box, and gets all the information about a node by selecting the query box.

The authors in [She03] presented Guerilla management protocols that assign different management functionalities for each of the nodes based on their capabilities, especially since nodes in ad hoc networks are heterogeneous and range in functionalities and capabilities (i.e. from a sensor to a laptop). Therefore, the powerful nodes act as

managing nodes, while the others are merely managed nodes. The network is clustered with nomadic nodes that act as cluster heads.

The three different categories assigned to the nodes, as shown in Figure 3-1, are as follows:

- *SNMP agent*: represents the nodes with the lowest capabilities in the network, so they simply execute an SNMP agent.
- *Probe processing module*: has, in addition to an SNMP agent, a lightweight execution environment capable of executing incoming active probes. This represents the nodes with sufficient capability.
- *Nomadic Management Module*: represents the nodes with a sufficient energy level and appropriate processing power. The nodes exchange information between each other, maintain management intelligence and stances, and can migrate to other nomadic managers in case of a capabilities or network change so they can become an SNMP-agent or a probe processing module. Every Nomadic Management Module node has a Guerilla MIB (GMIB) to maintain all management-related data.

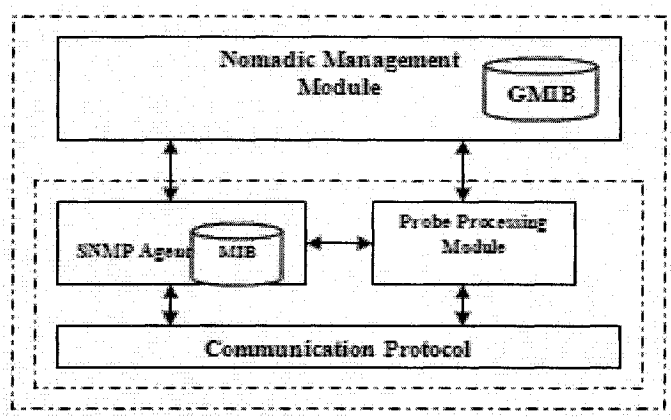

Figure 3-1: Different Categories of Guerilla Management Protocols

In order for nomadic managers to collect management information from remote nodes, small packets called probes are deployed. Those packets are divided into two categories: monitoring probes and task-specific probes. The monitoring probes are used by the nodes that have the ability to become nomadic managers to obtain network information from surrounding nodes, while the task-specific probes are used to collect application-specific information.

Both Guerilla and ANMP protocols can be used to manage sensor networks if the conditions for choosing "cluster head" in the case of ANMP and "nomadic manager" in the case of Guerilla will be limited to the sensor node with the highest remaining power (battery), since most of the applications of sensor networks require homogeneous nodes. The Guerilla management protocol is made in a very general manner, which makes it very suitable for many purposes since it can be implemented easily in many networks [Eik03].

The authors in [Rui03] presents MANNA architecture that identifies a set of WSN-specific managed objects mostly derived from OSI. It considers three-dimensional management architecture that consists of functional, information, and physical architectures:

- *Functional architecture* describes the management functionality of each of the network components: manager, agent, and management information base (MIB).
- *Physical architecture* is the implementation of the functional architecture. The model of mobile and intelligent agents[7] has been proposed as an alternative to

[7] A mobile agent can be defined as a software autonomous entity that fulfills a task specified by its owner. It has the ability to communicate with other agents and host systems, transfer itself to another agent-enabled host on the network, carry its state with it during migration, and make intelligent decisions concerning its goal based on learning [Qi 2003]

other paradigms such as SNMP, ANMP, etc., in which the management intelligence always resides in the managing instance. Due to the ability of mobile agents to support asynchronous communication and flexible query, these agents can reduce network traffic compared to traditional client-server approaches, as well as maintain load balancing, and can thus increase the performance of network nodes in wireless sensor networks.

- *Information architecture* is based on the object-oriented information model that contains object classes representing resources under different levels of abstraction. There are two types of object classes defined in the MANNA architecture: managed object class and support object class. The managed object class relates directly with the network components and with the network itself, and represents network, managed elements, equipment, system, environment, phenomenon, and connection. The support management class can be programmed in to the agent or present in the management application.

The authors in [Yu06] listed some design issues and requirements for proposing any efficient management architecture for wireless sensor networks:

- Lightweight management architecture: the protocol should be lightweight in terms of computation and communication requirements to fit the limited resources of a sensor node.
- Localized management and coordination
- Generic management functions: the management architecture should be application-independent so it accommodates the diversity of various wireless sensor network applications.

- Integration with application-knowledge: since the management architecture is generic and independent of the application, an efficient mechanism is needed to inject application knowledge into the infrastructure of the management architecture in WSNs and also to integrate it with the management services.
- Adaptive Reconfiguration: this is needed due to the long-running of sensor networks, to enable management architecture to reconfigure its operations and functions based on the changes in environments and circumstances.

3.3 Test-beds for Sensor Networks

Several wireless sensor network test-beds were designed and implemented. We will now describe some of them briefly. TrueMobile [Joh05] is a remotely accessible test-bed for mobile wireless and sensor networks, and is based on EmuLab[8], a test-bed management framework. TrueMobile consists of Garcia robots that carry 900 MHz mica2 motes and XScale-based Stargate small computer systems running Linux through a fixed field of sensor-equipped motes. The robots operate wirelessly using 802.11b and each fixed field of motes is attached serially to MIB500CA. The localization of robots is achieved using six ceiling-mounted video cameras and an image processing algorithm to transfer the videos to X, Y coordinates. TrueMobile serves to study the effects of mobility on wireless protocols, location algorithms, and sensor-driven applications. MoteLab [Wer05], open source software installed at MIT, Berkeley, and Harvard, consists of multiple Ethernet-connected sensor network nodes linked to a central server where users employ a reservation-based scheduling system with quotas. It uses a per-experiment

[8] It is a multi-user test-bed environment, so its resources can be shared by a large number of projects. It supports multiple device types, including generic PCs, emulated wide-area network links, and real 802.11 links

MySQL database to store all information needed for test-bed operation and provides to the user two different ways to use it: scheduling a large amount of jobs to be run unattended in a batch fashion, or interacting directly with the running jobs through Serial Forwarder[9]. Mirage [Chu05] is complementary to all of the other test-beds in that it uses combinatorial auction in a sensor network test-bed as a target to explore new models of resource allocation, e.g., market-based models. It is based on the fact that two requirements are needed for a resource allocation system; the first is the need for a systematic way to prioritize resource requests, while the second requires the users to be able to express the importance of their requests to use the test-bed. Users employ a resource discovery service to find a set of motes that meet their constraints, then place bids b_i in the auction:

$$b_i = (v_i, s_i, t_i, d_i, f_{min}, f_{max}, n_i, ok_i)$$

Where v_i = maximum amount the user is willing to pay

s_i, t_i = Time range of the experiment

d_i = usage duration time.

f_{min}, f_{max} = frequency range

ok_i = set of motes (specified in the resource discovery step)

n_i = number of motes.

After the bids are submitted, a set of winning bids is computed using a greedy heuristic[10] algorithm. The EmStar [Gir04] test-bed aims to deploy EmStar software that integrates high capability nodes specifically iPAQ or Crossbow Stargate platforms in order to develop and deploy more complex applications. Scale [Cer03] is built on top of the

[9] Serial Forwarder is an application developed by the staff at Crossbow Technology.
[10] This type of algorithms solves optimization problems by finding locally optimal solutions.

EmStar programming model to measure the link quality of the low radio channels, and displays this data to users through a web interface. A Delay/Fault-Tolerant Mobile Sensor Network (DFT-MSN) test-bed is presented in [Wan06]. It consists of two types of nodes: mobile sensors formed by Mica2 nodes attached to students moving in a library, and sink nodes that consist of one Mica2 sensor node connected serially to a laptop. The DFT-MSN test-bed is used to show the effectiveness and efficiency of a data delivery scheme proposed by the authors of the aforementioned paper. GNOMES [Wel03] is designed to explore the properties of heterogeneous wireless sensor networks. This test-bed uses the GNOMES node and GNOMES operating system. The GNOMES node occupies approximately 55 cm^3 and uses two sources of power; specifically, it uses one of two AA NIMH cells while the other is charged using an external device (i.e. a solar cell). It operates with either a 2.4 GHz Bluetooth radio or a 900 MHz radio module. The GNOMES operating system offers process management and scheduling that allows multiple applications to run on the node concurrently. The heterogeneity is explored where each node can sense a different property of the environment than its immediate neighbor. In [Hav02], the design and verification of a sensor node prototype was discussed. This prototype will be used in a test-bed to investigate the effect of mobility, failure, and communication protocols at many layers as a part of the European research project EYES. The communication function between nodes is realized by a RFM TR1001 hybrid radio transceiver that supports transmission up to 115.2 Kbps. The EYES node uses an MSP-430F149 processor and an extended memory. It uses the PEEROS (Preemptive EYES Real time Operating System) operating system, which is also designed by the EYES project. PEEROS is written in the C language and uses a task

scheduler to manage the multiple tasks running and a messaging system to enable communication between its different components.

Chapter 4 : Test-bed Elements

4.1 Introduction

This chapter presents an overview of the test-bed that is customized to assess sensor network management protocols. Detailed descriptions of each of the hardware and software used components in the test-bed are presented.

4.2 Test-bed Overview

Our test-bed consists of 3 main parts:

- Nios II Development kit + MIB510 with an on-board sensor node, acting as a base station.
- MICA2 sensor nodes + Rentron RF kit + Zeevo Bluetooth kit, acting as the sensor networks.
- PC, acting as an end-user.

Figure 4-1 shows how the various elements are interconnected.

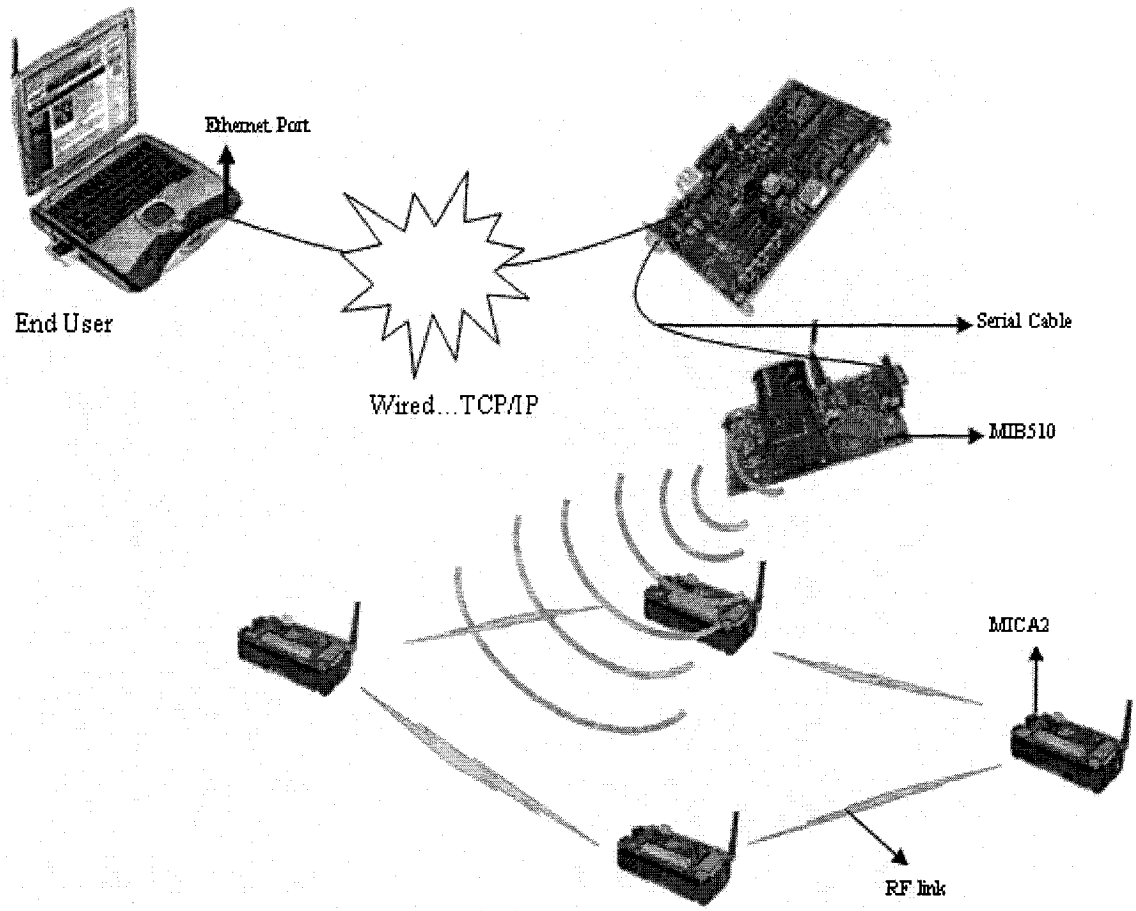

Figure 4-1: Test-bed high-level architecture

In the next four subsections, we will present an overview of each of the components used hardware in the test-bed.

4.2.1 MIB510: The MIB510 interface board is used as a sensor node programming board through its RS-232 interface, as well as a house for the base mote (node with ID0) during data transmission. The default baud rate is 57600 bits per second.

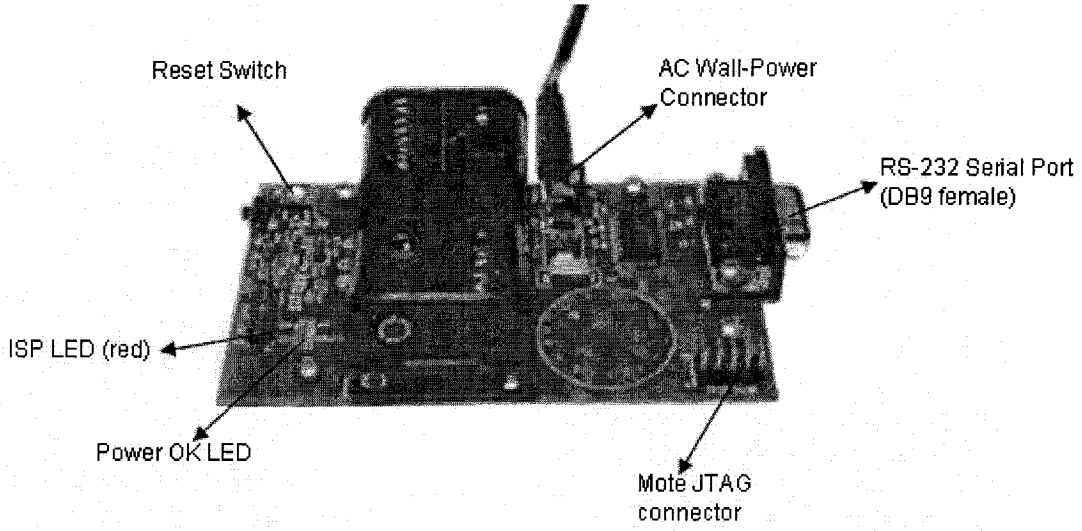

Figure 4-2: MIB510 overview

4.2.2 MTS420CA: The MTS420 sensor board offers five basic environmental sensors:

- Ambient light: It is built using the TLS2550 chip, a digital-output light sensor with a two-wire SMBus serial interface manufactured by TAOS [Tao06]. TLS2550 converts light intensity to a digital signal, and consists of two photodiodes: one sensitive to visible and infrared light, and one sensitive primarily to infrared light.
- Temperature and relative humidity: It is made of the SHT11 single multi-sensor chip, shown in Figure 4-3, from Sensirion [Sen06]. It consists of a calibrated digital output. The default resolution is 14 bit for temperature and 12 bit for humidity, but it can be reduced to 12 and 8 bit through the status register to reduce power consumption.

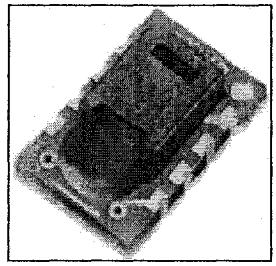

Figure 4-3: SHT11 chip

- 2-Axis Accelerometer: It is a MEMS surface micro-machined 2-axis, ±2g (gravity in m/s^2) device. It can measure both dynamic and static acceleration.
- Barometric pressure: It consists of a MS55ER SMD-hybrid device from Intersema [Int06].
- A GPS module that is used for positional identification of motes deployed in an accessible environment. It consists of a Leadtek LR9546 GPS module board, manufactured by Leadteck [Lea06]. GPS uses "triangulation" from satellites to determine the precise location of motes. A detailed description of the need for localization in sensor networks and the various localization algorithms used by sensor nodes are presented in Appendix A.

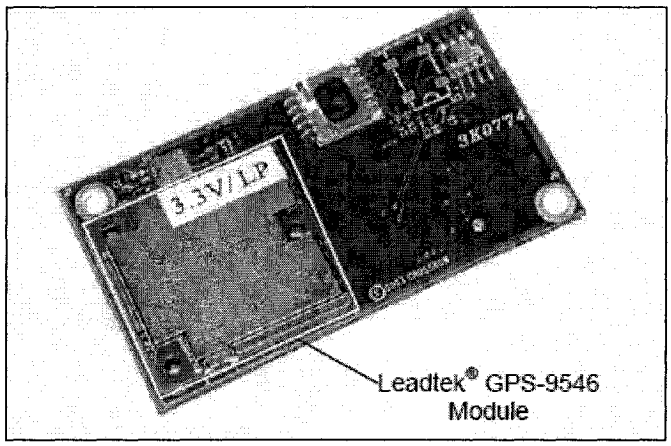

Figure 4-4: MTS420 board [Lea06]

Table 4-1 presents the voltage operating range, sensing range, and accuracy of each sensor on the MTS420.

Sensor	Voltage Operating Range	Range	Accuracy
Pressure	3.6 to 2.2 volts	300 to 110 mbar	± 3.5%
Light	3.6 to 2.7 volts	400 – 100 nm	-
Temperature	3.6 to 2.4 volts	-40°C – 80°C	± 2%
Humidity	3.6 to 2.4 volts	0 – 100 %	±3.5%
2-Axis Accelerometer	3.6 to 3.0 volts	± 2G (1G = 9.81 m/s^2)	±17%
GPS	3.6 to 3.0 volts	10 m, 2D	15 meters, 2D RMS[11], Selective Availability off 7 meters, 2D RMS, WAAS[12] Corrected 1-5 meters, Differential GPS Corrected

Table 4-1: Characteristics of each sensor in MTS420

4.2.3 MICA2: The MPR400 module, shown in Figure 4-5, may be powered by two AA batteries (the characteristics of which are shown in Table 4-3) or it can be fed with power by an external power connector. Its RF module operates at 915 MHz and uses the Chipcon CC1000, FSK modulated radio[13], 8-bitAtmega128L micro-controller with 4 Kb systems RAM, and128 Kb flash program memory. It uses an external ¼ wave whip antenna that offers improved coverage and is inexpensive (sensors should be placed such that the antennas are oriented vertically since a horizontal placement of the antennas will result in a substantial loss of distance). The module also contains an expansion connector that handles different sensor boards and runs the TinyOS multithreading, event-based operating system written in nesC.

[11] RMS: Root Mean Square, A one-dimensional measure of accuracy representing the limits (plus or minus) within which the true value lies 68 percent of the time.
[12] Wide Area Augmentation System - A differential correction system.
[13] The three popular modulation schemes are: OOK (On/Off Key), ASK (Amplitude Shift Key), and the FSK (Frequency Shift Key) but FSK is preferred over OOK and ASK due to its better performance in the presence of interfering signals and it does not require an adaptable threshold or an automatic gain control (AGC) in order to ensure an optimal threshold setting

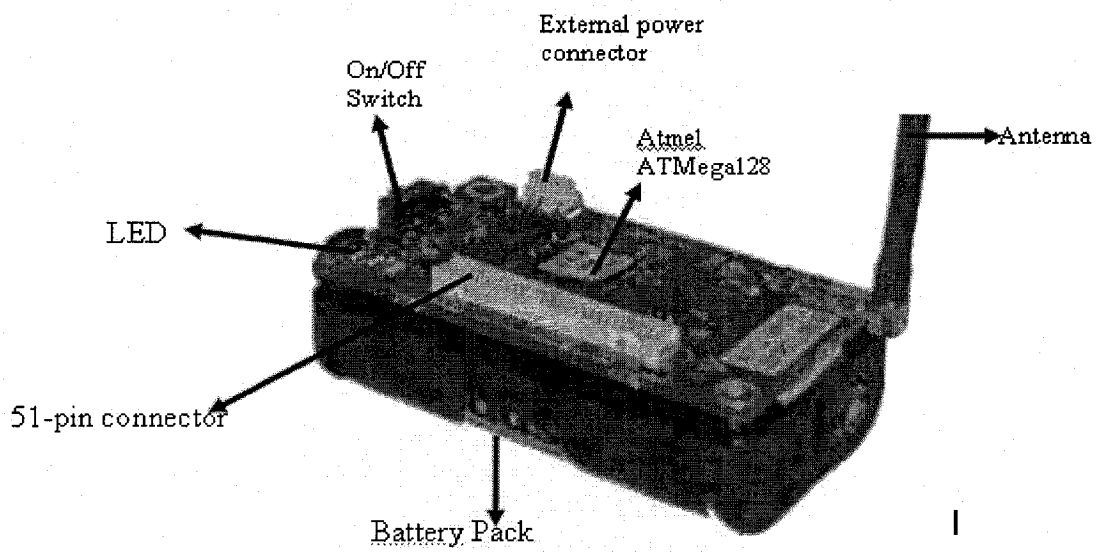

Figure 4-5: Mica2 Components

Width	58mm	**Number of channels**	4
Length	32mm	**Data Rate**	38.4 Kbaud
Height (without battery pack)	7mm	**RF Power**	-20 to +10 dBm
Weight (Excluding batteries)	18g	**Outdoor Range**	1000ft

Table 4-2: Mica2 Characteristics [Cro05b]

Battery Size	Capacity (ma-hr)	Battery Life @ 250 uA (yrs)	Battery Life @ 500 uA (yrs)	Battery Life @ 1ma (yrs)
AA	2000	1	0.5	0.25

Table 4-3: AA battery Capacity [Cro05b]

4.2.4 Nios II development board: The Nios II development board used in this test-bed features a Cyclone EP1C20F400C7 device with 20,060 LEs and 294,912 bits of on-chip memory. As well, the chip contains 48 blocks of DSP and is equipped with various off-chip modules that are connected to the FPGA and can be initialized when assigning the pins at the design phase before running the place and route algorithm. Figure 4-6 shows the board with its different components.

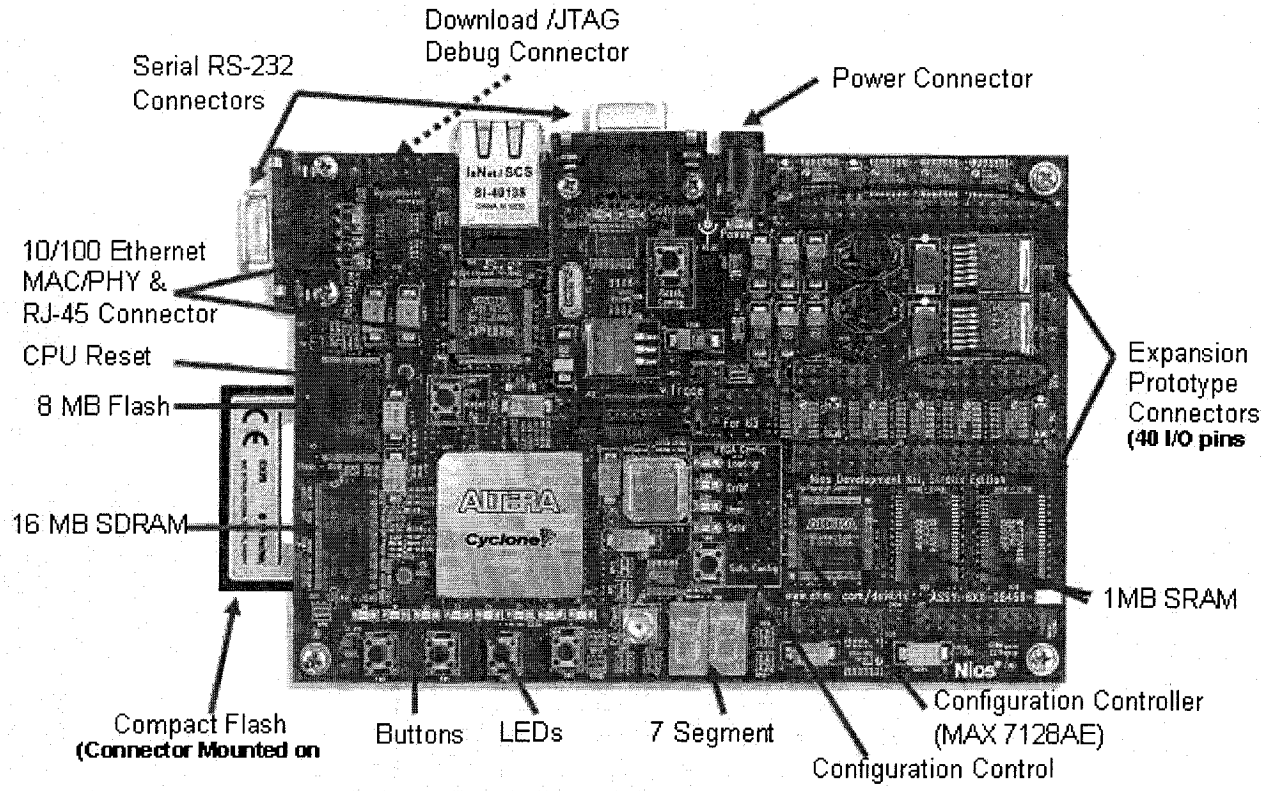

Figure 4-6: Nios Board components [Alt04b]

Many reasons were behind the choice of the Nios II development board [Abd05], mainly:

- Its capability to integrate programmable logic and processors onto a single device, which allows the partitioning of the design into software and hardware.
- The availability of various interfaces (two serial ports, one RJ-45 and many programmable I/O ports), which in our case permits scalability by for example, adding a new RF interface in the case of heterogeneous sensor networks.

4.2.5 Zeevo Bluetooth kit [Zee06]: It is a class II Bluetooth module. This device can be plugged into any UART or RS-232 compatible serial port [Wiro03]. Class II Bluetooth operates in the 2.4 GHz industrial, scientific, and medicine (ISM) band. It is an open standard for short-range radios of up to 10 meters at a maximum rate of 1 Mb/s. It uses

frequency hopping for low interference and fading, and time-division multiplex for full-duplex transmission. It transmits using Gaussian Frequency Shift Keying (GFSK) modulation [Ily02]. The nodes in a Bluetooth network are of two kinds: masters and slaves. A master is responsible for giving slaves (up to seven at one time) access to the channel. A master and seven slaves form a piconet, which is the fundamental building block of a Bluetooth network. The slave device has to be made discoverable/searchable by the master device that issues a discovery service. Figure 4-7 shows the transmitter of the Bluetooth kit.

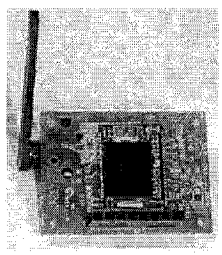

Figure 4-7: Transmitter of Zeevo Bluetooth kit.

4.2.6 Rentron RF kit: This kit is a low cost communication system package. The idea behind introducing these kits in our research is to show the capabilities of the NIOS II development board. The analog devices are connected to the FPGA through custom digital blocks implemented in VHDL and connected to the NIOS II processor as a slave module. When it is used to perform a one-way communication, as illustrated in Figure 4-8, it requires the following components:

- Two antennas: radiate and receive signals.
- HT-640 Encoder IC (A): performs the encoding of the data to be sent.
- HT-648 Decoder IC (D): decodes the data before it enters the FPGA.
- RWS-434 (C): performs a de-modulation of the incoming signal.

- TWS-434A (B): modulates the data at 434 MHz.

The system can operate at a maximum bit rate of 3 kbps and a voltage supply ranging from 2V to 12V. Also, communication between the transmitter and the receiver is performed in an asynchronous manner.

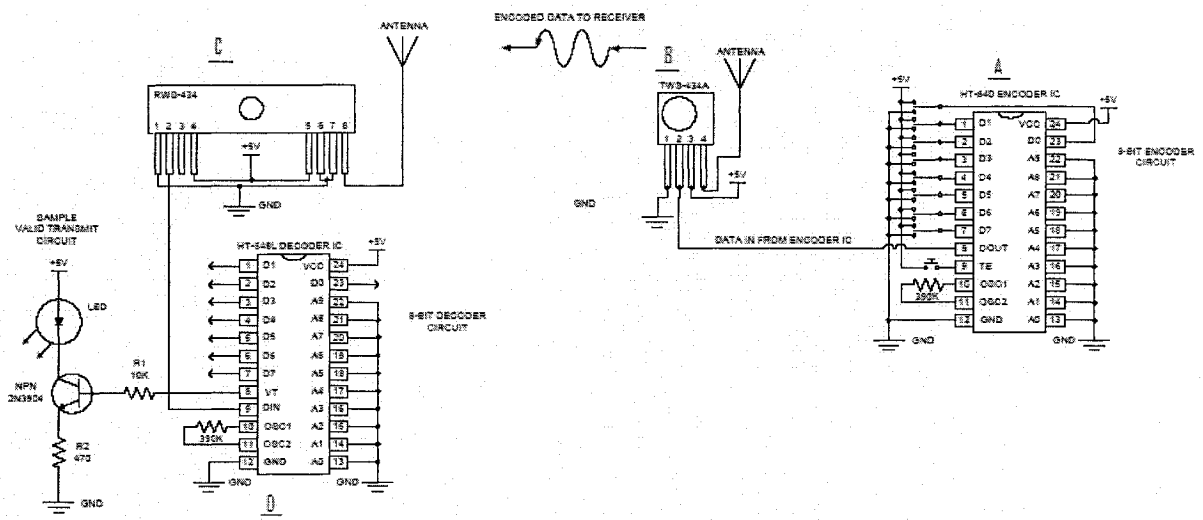

Figure 4-8: Overview of Rentron RF kit [Ren01]

Figure 4-9 shows the pins of the transmitter and the receiver, as well as an illustration of their statures.

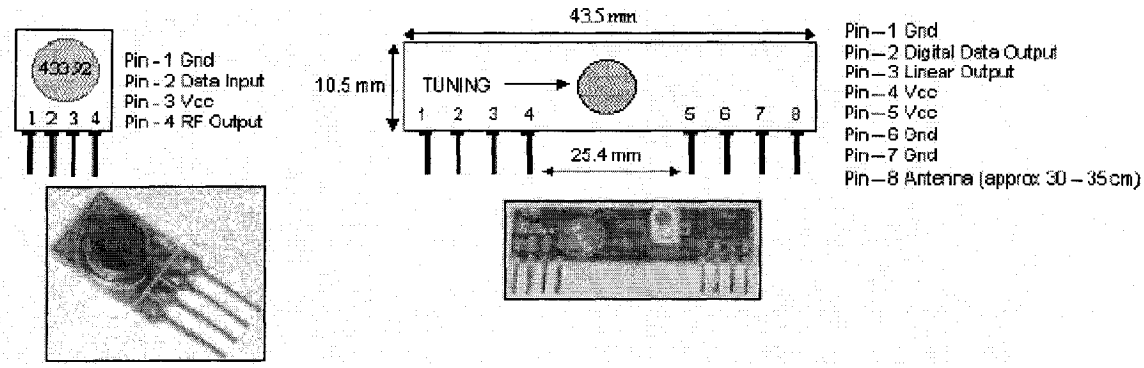

Figure 4-9: A close view of TWS-434A Tx and RWS-434 Rx with a description of the pins [Ren01]

4.2.7 MySQL database: The database used is MySQL 4.1.21, a relational database management system written in C and C++. In what follows, we will list the reasons behind the usage of MySQL [Mys06]:

- It supports many data types: signed/unsigned integers 1, 2, 3, 4, and 8 bytes; FLOAT, DOUBLE, DATE, TIME, YEAR, etc…
- The existing version of MySQL Connector/J allows applications written in Java to connect to a MySQL database and access table information.
- Clients can connect to the MySQL server through TCP/IP connections.
- The table size is only limited by operating system constraints on file sizes (for example, in MySQL 3.23, the maximum table size was increased to 65536 terabytes).

Chapter 5 : Hardware and Software Implementation

5.1 Introduction

This chapter presents a description of the test-bed implementation; we explain the implementation pf both software and hardware components.

5.2 User Interface

A custom graphical user interface (GUI), shown in Figure 5-1, was designed and developed using Java in order to provide a friendly means of managing the test-bed and displaying information of interest. We used the open source "Eclipse" Integrated Java Environment [Ecl06] to edit, compile, and debug our codes. Eclipse has its own **GUI** framework called the Standard Widget Toolkit (SWT). Java was used for the following reasons:

- Platform Independence;
- Good GUI design capabilities and the availability of two libraries AWT and Swing.
- Reliability and security.

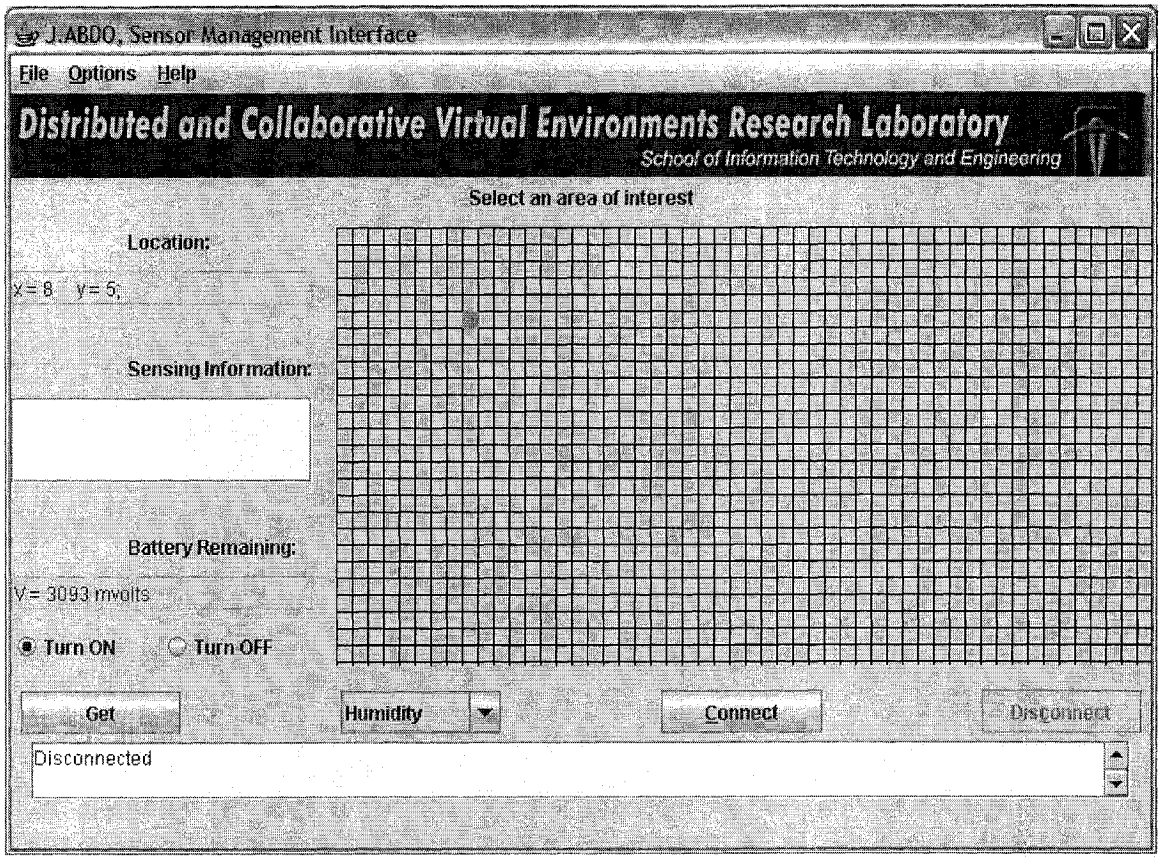

Figure 5-1: User Interface Snapshot

The management and monitoring operations, the flow of which is shown in Figure 5-2, are performed in the following manner: after a successful connection to the base station, the user will be able to choose a location (Green Color) and get the appropriate sensing information or the voltage remaining. It is important to note that we cannot get the current value directly from the motes unless we use a multi-meter to measure the current when the motes are in operation. The user can also toggle the node mode (on/off). A history of the measurements at each point is saved in a MySQL database, accessible through the "Options" menu. A user guide of the test-bed is available from the "Help" menu, in addition to a mapping between the area of the sensor nodes deployed in the GUI and with real life space. Error messages from the base station are posted in the "TextField".

The radio range for a Mica2 sensor node operating at 914 MHz is 100 to 300 feet. Based on that range and taking into consideration the influence of the environment (Foliage and other RF obstacles) on the range distance, the mapping between the area of sensor nodes displayed in the GUI and the real life space obeys the following logic. Each square in the GUI (which is also one unit of the size of the GUI) represents 40 sq feet. The normal size for the area occupied by the sensor nodes in the GUI (directly after launching the GUI) is 51 units (Height)*25 units (Width) = 1275 units, so an area of 51,000 sq feet (1 foot = 30.48 cm $\rightarrow$ 1554 m^2) is covered in real life.

Communication between the GUI and the base station is depicted in the flowchart in Figure 5-2.

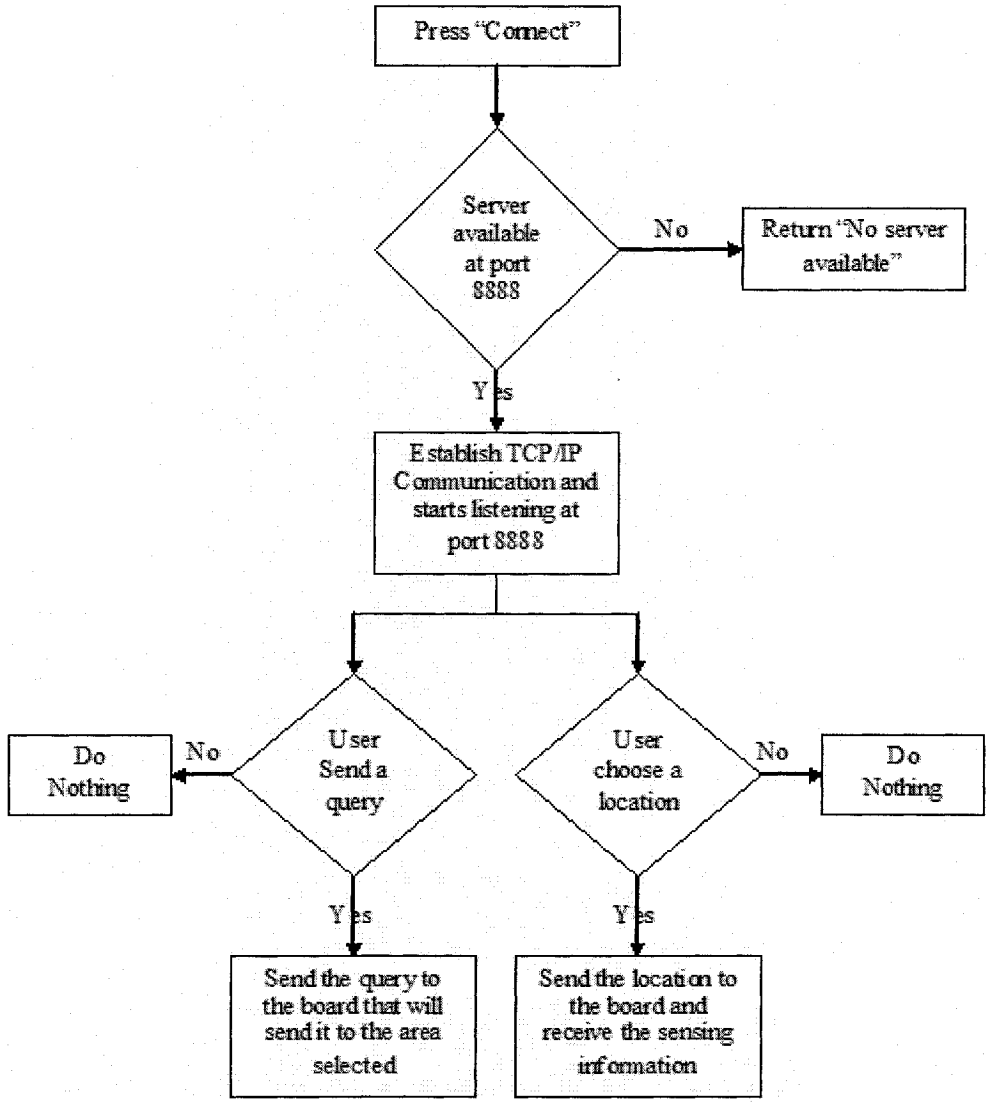

Figure 5-2: Flowchart of the GUI interaction with user and base station.

The functionality of the GUI is achieved by implementing the following functions:

Function	Description
GUI_1	Initiate all of the GUI components and display the frame
Topos: It is composed of two classes. The constructor Topos(Rectangle) and the function paintComponent(Graphics)	Topos(Rectangle): This constructor returns the location clicked. The location is composed of x and y and is computed by dividing the area clicked on the value of each square (10 in our case).
	paintComponent(Graphics): Set the color of

	the border lines to black and the color of the chosen location to green. Every square represents 10 units.
ActionAdapter	Action adapter for easy event-listener coding and for instantly adapting the listener to an event on an individual level.
changeStatusTS(int, boolean)	The thread-safe way to change the GUI components while changing states
changeStatusNTS(int, boolean)	The non thread-safe way to change the GUI components while changing state
cleanup()	Clean after disconnecting: Empty the buffer and close the socket.
Run()	Checks the current state (connected, disconnected, connecting, or disconnecting) and sets the enables/disables of the buttons accordingly.
sendTo(String)	This function sends the information given by the argument String (location + command) to the socket.
Main(String[])	The main procedure checks the current status and the user activity. If a location is chosen and a command is given, this information will be send to the socket using sendTo() command. It also checks whether data is available at the socket and appends it to the appropriate "textField".

Table 5-1: List of the GUI functions

5.4 Software Implementation

The inter-task communication between various developed tasks, needed in order to fulfill the base station functionality within the NIOS, is shown in Figure 5-3.

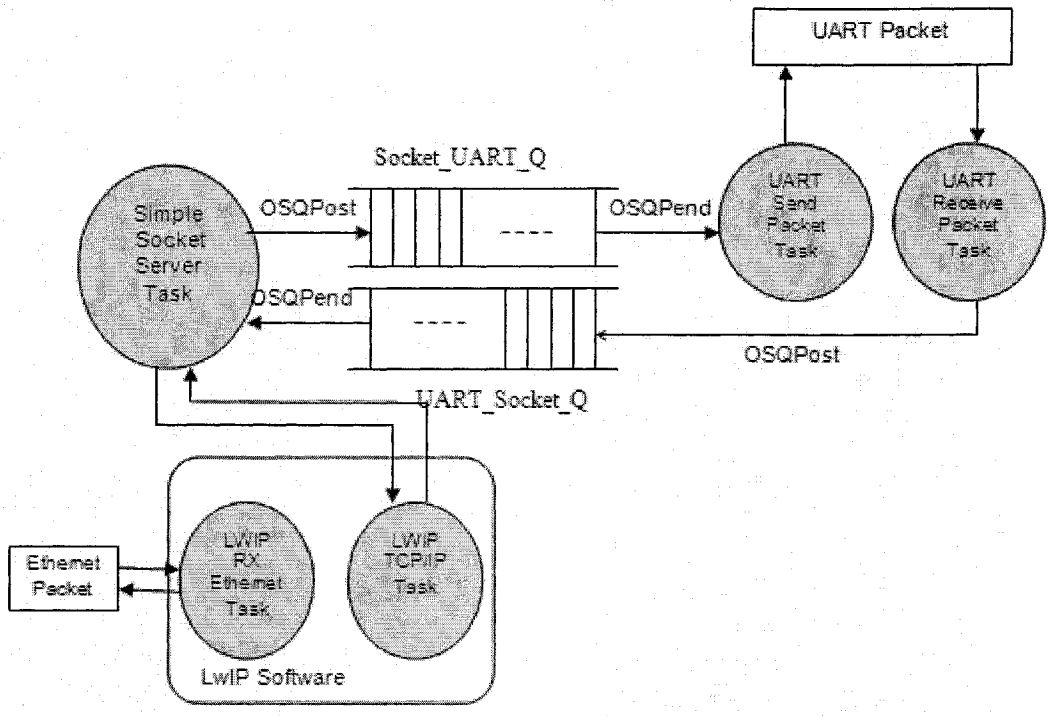

Figure 5-3: Task interactions running on NIOS II

The diagram presented in Figure 5-3 shows the state of the system after initialization. The Ethernet packet containing the queries or management commands sent from the SMP client (user managing the network) is received by the lwIP software component. The latter processes the incoming Ethernet packets and presents the data packet to the "Server task" [Alt04b], which will put it in the queue (Socket_UART_Q). The "UART Send Packet" task, responsible for sending the packet with the appropriate format to the UART hardware, will then send it to the MIB510. These are the actual commands passed from the SMP user to the socket on the NIOS development board, and from there to the UART management task.

CMD_ON_BIT_1	'1'
CMD_OFF_BIT_2	'2'
CMD_Press_BIT_3	'3'
CMD_Temp_BIT_4	'4'
CMD_Lig_BIT_5	'5'
CMD_Volt_BIT_6	'6'
CMD_Hum_BIT_7	'7'
CMD_Axis_BIT_8	'8'
CMD_QUIT	'Q'

The NIOS II development package comes equipped with LwIP which runs on top of MicroC/OS-II. LwIP contains some standard procedures that are used to initiate the hardware and the software needed to perform IP functionality. In Table 5-2 below, we will list the procedures and give an overview of their roles.

Functions	Description
Lwip_stack_init()	Initializes the stack (performs setup for the protocol stack)
Lwip_devices_init()	Initializes all the installed Ethernet device drivers. It returns a non-zero value to indicate a success.
Get_mac_addr()	Sets the MAC address of the device.
Get_ip_addr()	Sets the IP address of the device.

Table 5-2: List of LwIP standard functions used in our implementation

Server Task: This task creates a socket and calls relevant subroutines to manage the connection:

- Ready_Connection(): This routine is called when the listening socket has an incoming connection request. Based on the number of connections available, this incoming request will be rejected or accepted.

- Received_Command_Analysis(): This routine is called when data are being made available to read in a socket, and this data will then be analyzed if valid.

Receive_UART task: This task receives the TinyOS packet from the UART, which keeps listening until it starts receiving data from the MIB. It analyzes this data and is deciphered by calling the following subroutines:

- Binary_to_Hexadecimal(): Converts the format of the received packet from binary to hexadecimal.
- Decipher(): Analyzes the packet by extracting the required information (in our case, the information needed depends on the application or the request of the user, i.e. voltage, temperature, pressure, humidity, etc.). The conversion to engineering units of the TOS_Msg is done using the following routines:
    - *Unframed_packet():* Converts escape sequences from a packetized TOS_Msg to normal bytes.
    - *void xpacket_print_time():* Prints out the timestamp of when the packet was heard.
    - *Alt_u16 battery_conversion():* Converts battery readings from raw ADC data to engineering units using the following formulas.

      BV = RV*ADC_FS/data

      where:

      BV = Battery Voltage, ADC_FS = 1023

      RV = Voltage Reference for mica2 (1.223 volts)

data = data from the ADC measurement of channel 1

BV (volts) = 1252.352/data, BV (mv) = 1252352/data

- *Float acceleration_conversion():* Computes the ADC count of ADXL202E Accelerometer - for X axis reading into Engineering Unit (mg), per calibration.
- *Float Humidity_conversion():* Computes the ADC count of Humidity sensor readings into Engineering Unit (%).
- *Float pressure_conversion():* Computes the pressure ADC count of barometric pressure/temperature sensor readings into Engineering Units (mbar).
- *Float temperature_conversion():* Computes the Temperature reading into Engineering Units (degree Celsius).
- *Float Light_conversion():* Computes the light sensor reading into Engineering units. It returns the light level in lux.

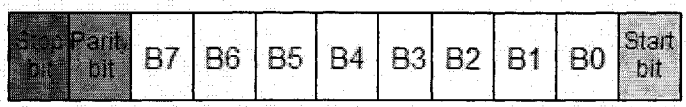

Figure 5-4: UART Packet

Send_UART task: This task receives the Ethernet packet containing the commands and queries from the user, and converts it into a TOSMsg using the structure:

```
typedef struct TOSMessage
{ alt_u16 node_address;
  alt_u8  packet_type;
  alt_u8  group_ID;
  alt_u8  data_length;
  alt_u8  data[length];
  alt_u16 crc;
}
```

Where data[29] is presented by the following structure:

```
typedef struct CommandMessage
{  alt_u8 command;
   alt_u16 source_ID;
   alt_u8  num_hop;
} CommandMessage;
```

SSSInitialTask(): This task initializes the operating system data structures and creates the non-lwip tasks. The highest priority (1) is assigned to this task since it is the first task that the MicroC/OS-II runs in order to get all real-time operating system resources.

Network utility task: It gives the board a static IP address after a certain timeout (if the DHCP server is not responding). The priority of this task is set to 2 since it is not used after assigning an IP address.

5.5 Hardware Implementation

A custom packet, presented in Figure 5-5, is introduced in this work. It is used for demonstration purposes since, in any wireless communication; packets transmitted should include a correction code such as parity and CRC. It consists of a "start bit" followed by 8 bits serially transmitted. The "start bit" will trigger the state machine of the controller of the receiver by indicating that a new packet is on the way.

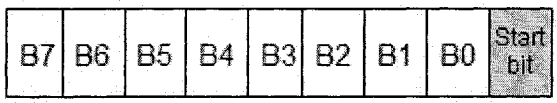

Figure 5-5: Packet format used in the RF transmission

The state machine of the transmitter controller is presented in Figure 5-6; it is formed by three states: "Idle", "StartBit" and "Transmit". When the user wants to send a packet, he/she will write it into the transmitter FIFO, which will set the 'transmit' signal to '1'. At the "StartBit" state, the start bit ('1') is shifted to the output pin of the FPGA, and it will be written on the pin 'D0' of the HT-640 Encoder IC. Then the controller will start shifting the bits of the packet one by one, until the counter initialized at the "StartBit" state indicates 111 (equivalent to 8 in decimal). At this point, the controller will stop, return to the "Idle" state and wait for a new packet to be transmitted.

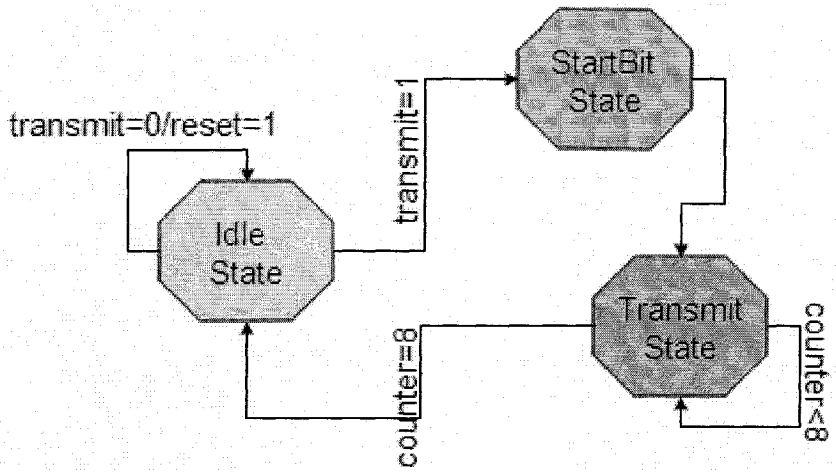

Figure 5-6: State machine diagram of transmitter

The receiver state machine is presented in Figure 5-7. It is comprised of four states: "Idle", "StartBit", "Receive" and "LoadPkt". The operation of the receiver is as follows: as long as the input line is not sensing a high voltage (3.3 V), the controller state remains "Idle". When the input is set to '1', the controller goes in to the "StartBit" state, where the 3-bit counters are enabled. Then it jumps into the "Receive" state and keep shifting

the incoming data until the counter indicates "111". The last state loads the full packet into the receiver FIFO and then goes back to the "Idle" state to wait for a new packet.

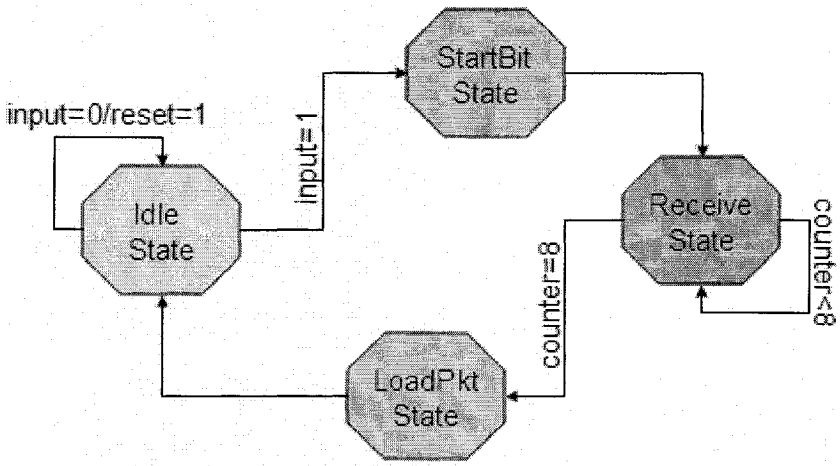

Figure 5-7: State machine diagram of receiver

Both the receiver and the transmitter controllers run on the rising edge of a 2.4 KHz clock, generated by dividing the clock of the "board oscillator" (50MHz). The generation is performed in two stages: the first stage is through "pll_mega_v0" and the second through "clock_divider_ja".

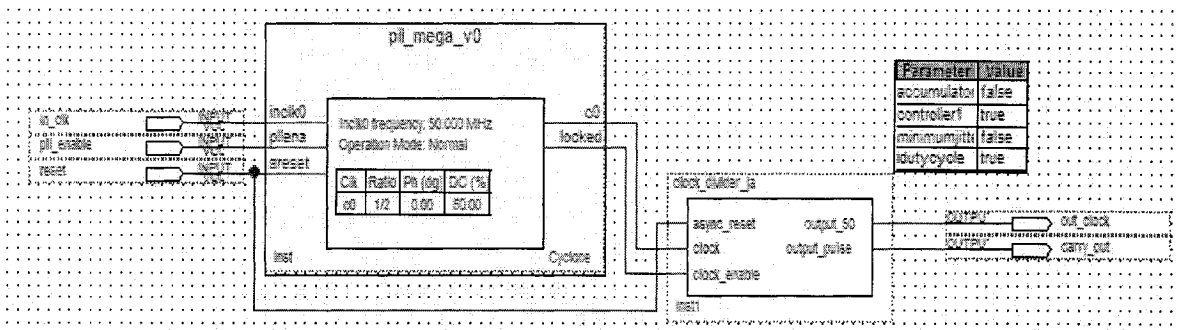

Figure 5-8: Schematic of a 2.4 KHz clock generator

5.6 System Generation

The custom software codes, interpreted using the soft-core 32-bit RISC NIOS II processor, run on top of the MicroC/OS-II[14] real-time operating system (RTOS) [Lab02]. This operating system provides a powerful, embedded, multi-threaded environment, as well as efficient inter-task communication through message passing [Abd05]. TCP/IP communication is done using the popular small footprint Lightweight IP TCP/IP stack [Dun01], which is suited to embedded systems. The NIOS II processor is part of a System-on-chip generated by SOPC builder that is used to connect hardware components of interest before the generated VHDL files are compiled/synthesized using Quartus v4.2. Finally, we use the Byte Blaster cable to download the image to the Cyclone chip. The NIOS II system generated is shown in Figure 5-9.

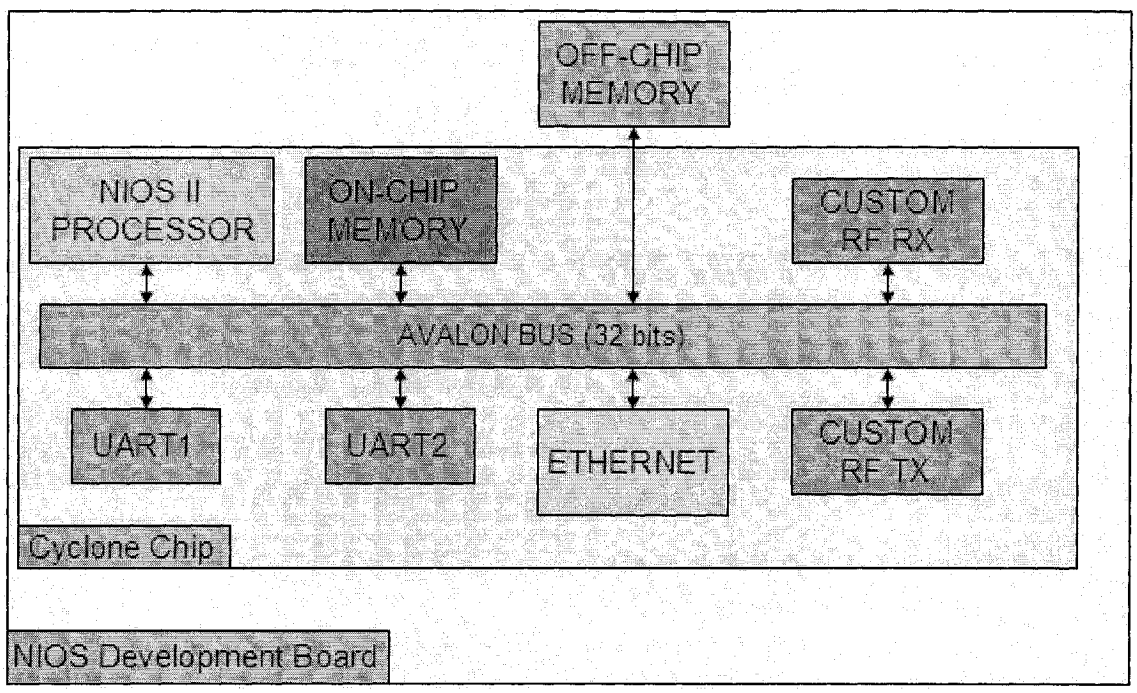

Figure 5-9: NIOS II generated system

[14] Many operating systems are available for the Nios II processor, to name few: ThreadX from Express logic, µcLinux from Microtronix, Nucleus Plus from Accelerated Technology and MicroC/OS-II from Micrium Inc.

The resource utilizations of the system are shown in Table 5-3.

Resource	Number of elements occupied (percentage)
Logic elements	757 (7%)
Total pins	25 (7%)
Memory bits	512 (< 1%)
DSP blocks 9-bit elements	4 (8%)
PLLs	2 (33 %)

Table 5-3: Chip resources utilized by the NIOS II system

5.7 Sensor Field

TinyOS is an event-driven operating system, developed by UC Berkeley to be used in wireless embedded sensor networks. It has a very small footprint, in that the core OS requires 400 bytes for the combination of code and data memory [Gay03]. The TinyOS applications, library, and system are written in nesC, an extension of the C programming language. Every program in nesC is built out of separate components that are wired together using interfaces to form whole programs. The interfaces in nesC are bidirectional. nesC specifies a set of named functions, called commands, to be implemented by the interface's provider, as well as a set of named functions, called events, to be implemented by the interface's user. NesC defines a concurrency model, represented by Tasks and Events. Tasks are not time-critical and run to completion following the FIFO scheduling with respect to other tasks, which implies that only a single stack is needed; however Events may preempt the execution of a task or another event. The networking in TinyOS is done using the Active Messages communication model, in which each Active message contains the name of an application-level handler to be invoked on a target node upon arrival as well as a data payload to pass in as

arguments [Hil03]. The TinyOS serial data packet, shown in Figure 5-10, has the following format [Tho05]:

- It supports variable packet size but has a maximum length of 255 bytes.
- The synchronization byte is 0x7E at both the start and end of the packet.
- It uses an escape byte of 0x7D.
- It uses a 16-bit CRC that is computed over the entire packet for error detections.

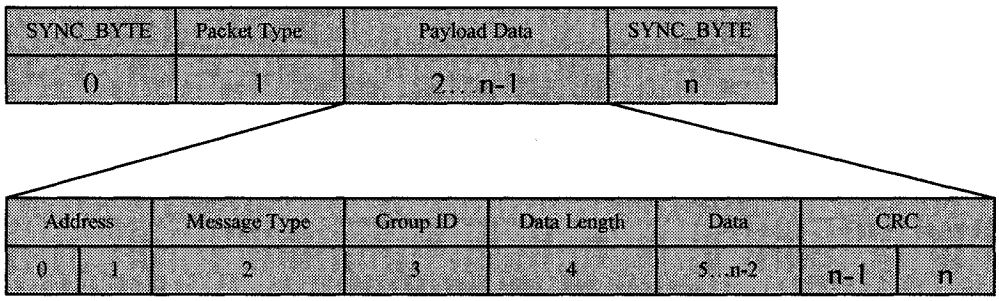

Figure 5-10: TinyOS serial packet format [Tho05]

Byte #	Field	Description
0	Packet frame Synch byte	0x7E
1	Packet Type	P_PACKET_NO_ACK →0x42 P-PACKET_ACK → 0x41 P_ACK → 0x40 P_UNKNOWN → 0Xff
2-3	Message Address	Broadcast Address → 0xFFFF UART Address → 0x007e Node Address
4	Message Type	Each application will have its own message type
5	Group ID[15]	The default value id 0x7d
6	Data Length	The number of bytes (l) of the data payload.
7...n-2	Data Payload	The actual message content
n-1...n	CRC	Two bytes used for error detections.

Table 5-4: TinyOS serial packet detailed description [Tho05]

[15] Only motes with the same group ID will talk to each other.

The protocol used for serial communication by TinuOS-1.1 is loosely based on the PPP (Point-to-Point Protocol) in HDLC-like (High Level Data Link Control) framing.

After communication is established, a topology discovery starts in which the base station returns to the end user the location of all the nodes, which will be colored red in the sensor field. The user will be able to choose a location, and two options will be available:

1. The location is colored green, e.g. it contains a node that is turned off, so the user will have the choice:
    - To turn the node on; or
    - To get the temperature, light, or pressure (depending on the sensors that every mica2 mote has on-board). This information will be calculated from the surrounding nodes.
2. The location is colored red, e.g. it contains a node that is turned on, so the user will have the choice:
    - To turn the node off; or
    - To get the temperature, light, and/or pressure (depending on the sensors that every mica2 mote has on-board).
3. The location is not colored, e.g. there is no node in that location. The user will have the ability to get the temperature and/or pressure data calculated from the nodes surrounding the chosen location.

Every sensor node in the field will apply the command sent by the manager based on what it receives in its payload:

Command	Payload data
Turn On	1
Turn Off	2
Pressure	3
Temperature	4
Light	5
Voltage Remaining	6
Humidity	7

Table 5-5: Commands sent to the motes

5.7.1 Motes Programming

Compile and program a Mica2 mote with the "TOSBase" application, which is provided by Crossbow and can be found in /opt/tinyps-1.x/contrib./xbow/apps/TOSBase directory. The mote that is programmed with "TOSBase" will act as a gateway between the wireless sensor network and the NIOS Development Board. The application will be compiled and stored in the mote's flash memory using the following command:

make mica2 install,0 mib510,com1

0 is the id of the node
Com1 is the port used for installing the application on the mote

The "TOSBase" application acts as a bridge between the wireless link and serial port. The LEDS of the MIB510 are programmed to toggle as follows:

- RED Toggle: Packet received over the serial (UART) channel
- GREEN Toggle: Packet received over the wireless channel
- YELLOW Toggle: Acknowledgment (ACK) sent back to the serial channel

Appendix B shows the wiring of the different components used in TOSBase and Network_MTS420.

After the establishment of communication, each sensor node in the network runs the "network_MTS420" application that has the following algorithm:

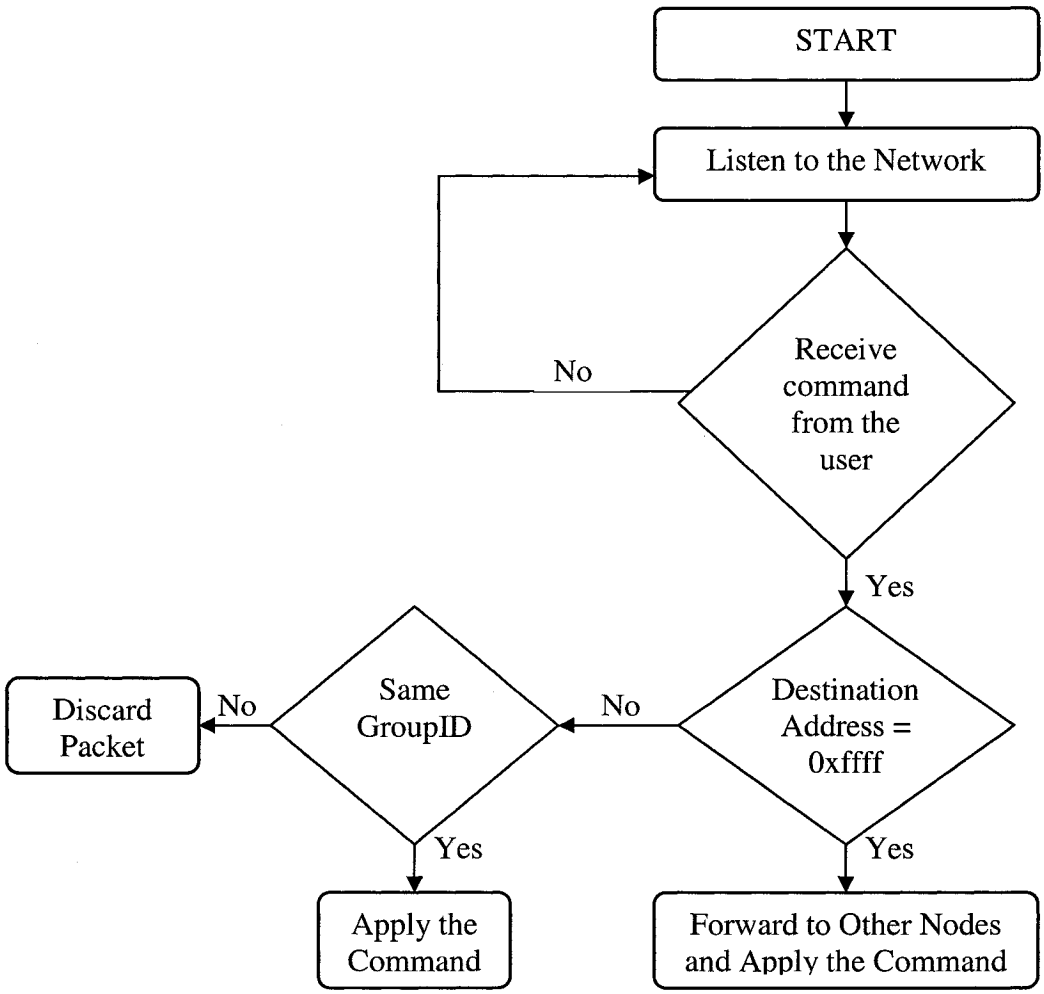

Figure 5-11: Flow control of the running application

Chapter 6 : Experimental Results and Discussion

6.1 Introduction

In this chapter we present the outputs that demonstrate the functionality of the various modules of our implementation, as well as the results generated by different experiments performed using the test-bed.

6.2 Analysis of a TinyOS packet

The format of the packet sent by the sensor node to the MIB510 is illustrated in chapter 5; however, in our implementation we are interested in the packet generated by the MIB510 and sent to the Cycle board, which parses the information of concern and "mails" it to the user. Figure 6-1 shows an example of a packet received by the end user, as well as an illustration of the important parameters included.

Figure 6-1: Sensing values from TinyOS packet

The analysis of the received packet is presented in Table 6-1.

Packet Type	Address	Message Type	Group ID	Data length	Payload
42	ffff	00	7d	1d (29 bytes)	85 01 02 01 96 01 b3 07 ce 18 92 b6 da 93 6d ad 02 ba 75 63 50 4b e5 00 ae 00 f5 01 f2

Table 6-1: TinyOS packet deciphered.

It is worth noting that the data are sent by the mote in *little-endian* format; thus, for example, the two bytes `01 f2` represent a single sensor reading with most-significant-byte `0xf2` and least-significant-byte `0x01`.

6.3 Emulation of the TCP/IP communication

Figure 6-2 is provided to represent the functionality of our "Sensor Management Interface". A snapshot of our GUI is shown while it is performing a two-way TCP/IP communication with another Java-based custom interface (tester). The latter interface is dedicated to emulate the NIOS II development board by opening/closing connections and sending/receiving data. In this case, the data exchange is performed on TCP port "1234", while both interfaces are running on the "localhost". We can see the x and y of the selected location and the sensing information of interest being transmitted to the "tester", which will reply in turn by random sensing information.

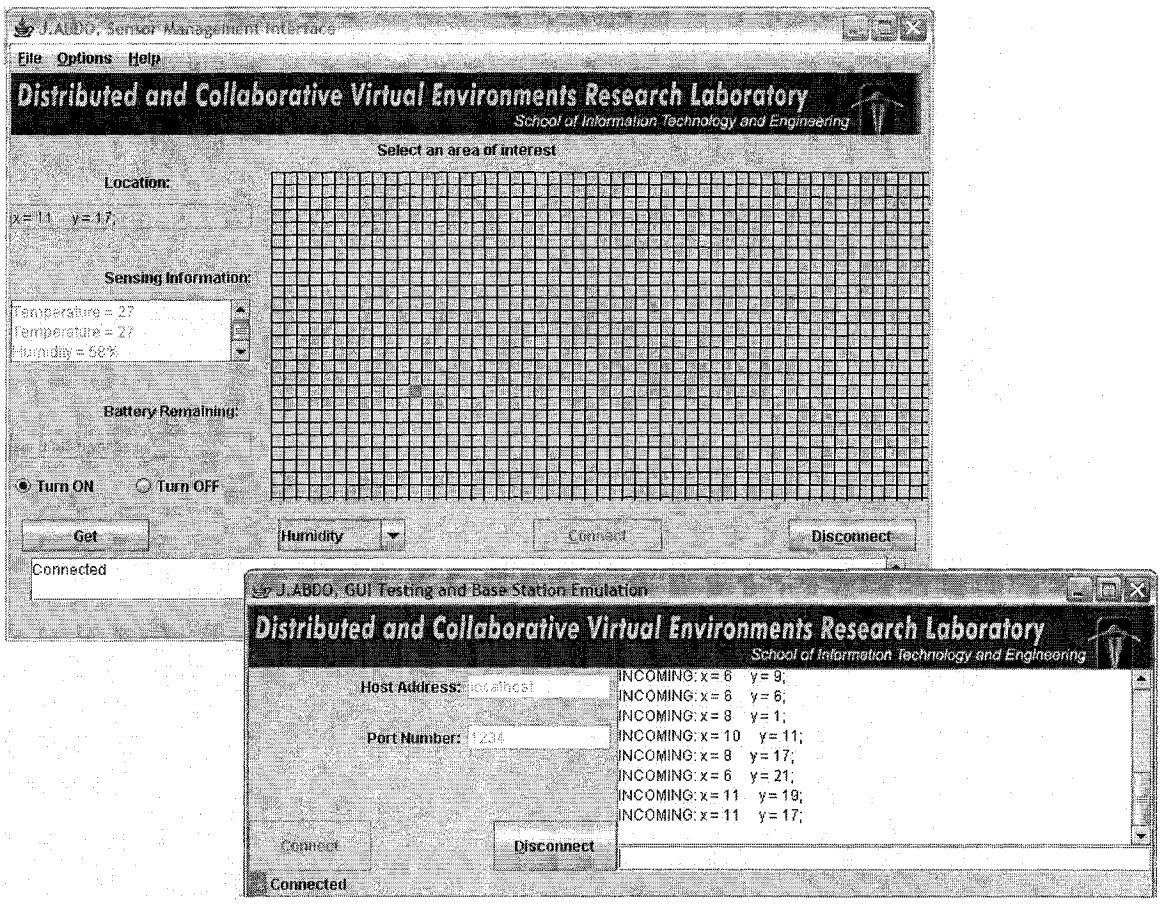

Figure 6-2: Snapshot demonstrating the functionality of the GUI over TCP network

6.4 Data collected by the database

Figure 6-3 shows a sample of the data collected from the sensor nodes and stored in the MySQL database. Here we see the title of each column of the SQL table: location of the node, date and time of reception, temperature, battery remaining, pressure, light, X-Axis Acceleration, and Y-Axis Acceleration measurements.

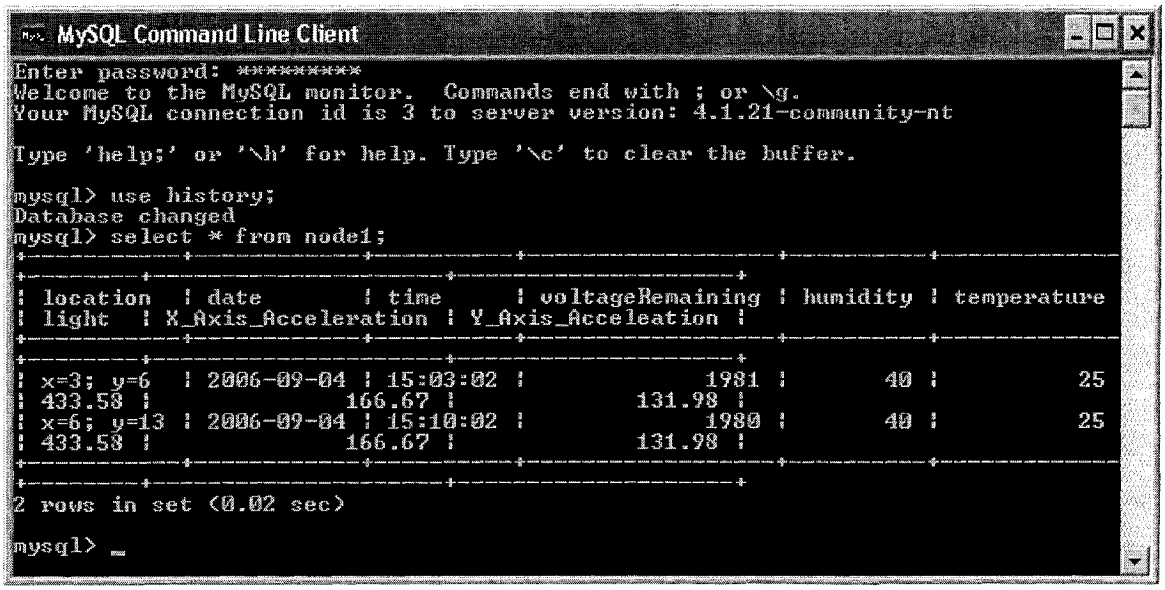

Figure 6-3: Sample of data stored in the database

6.5 Simulation of the RF receiver/transmitter

A simulation of the clock generator, which feeds the digital blocks of the RF transmitter and receiver, is shown in Figure 6-4. We have three inputs (async_reset, clock, and clock_enable) and two outputs (output_50 and output_pulse). The input clock runs at 50 MHz and the output is generated at 2.4 KHz.

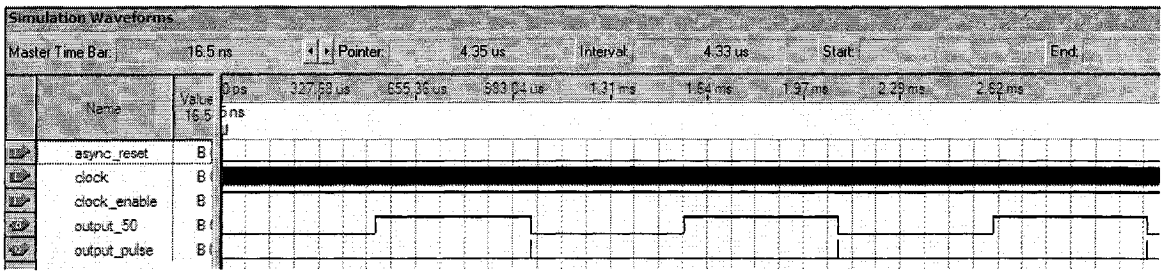

Figure 6-4: Simulation of the data rate clock generator

A simulation of the RF receiver digital block is presented in Figure 6-5. We can see that two incoming packets, at pin 'input_data', are serially entering the block. By looking at the voltage level, we have "101011000", when the first '1' is the start bit of the packet

and the payload is thus "00011010" because we are using a right shift. The second packet is "111011000", also with a start bit of '1' and a payload of "00011011".

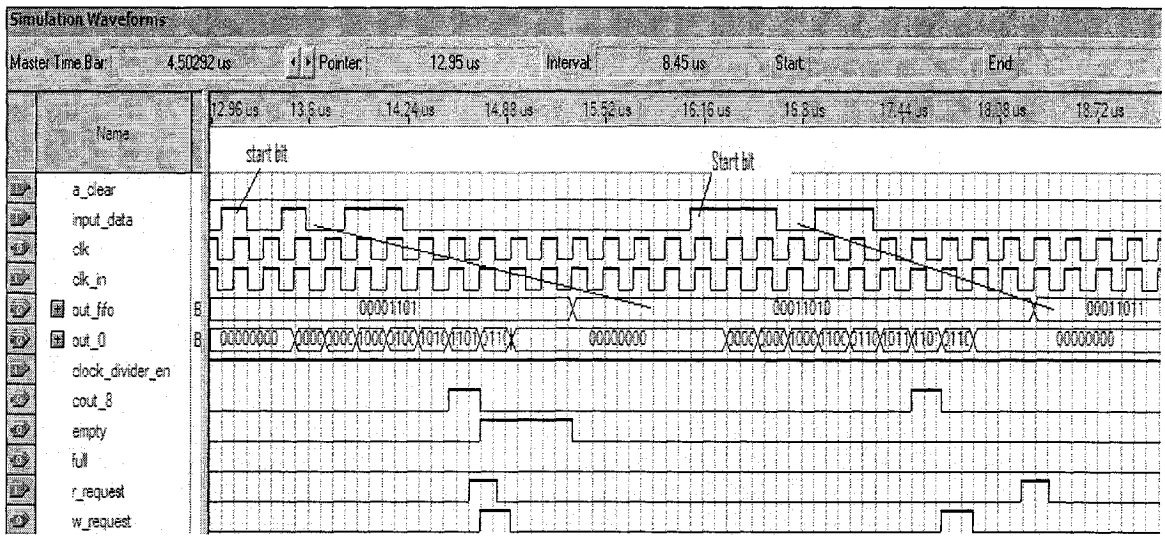

Figure 6-5: Simulation of the receiver' digital block

Figure 6-6 presents a timing simulation of the digital block of the transmitter. It shows two packets, "00110011" and "10101100", that are loaded in parallel to the "input_packet" 8-bit bus. The packets are then transmitted serially, as shown at pin "data_out", where a start bit of 1 is added. The output serial data will be transferred to analog using the RF Rentron kit.

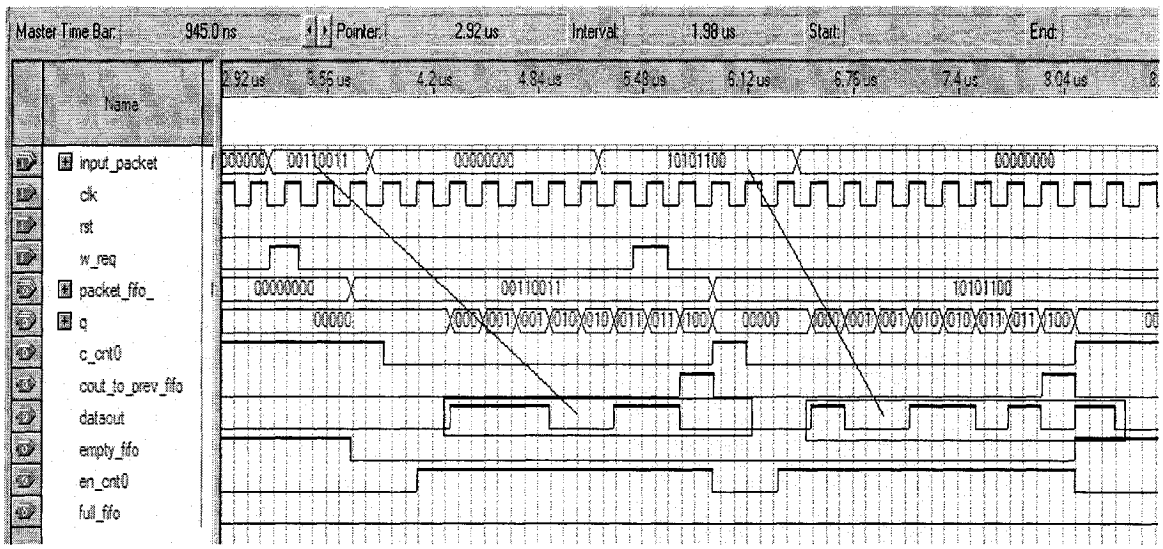

Figure 6-6: Simulation of the transmitter' digital block

6.6 Data collection using the test-bed

6.6.1 Power consumption of the transmitter: One of the usages of our test-bed is to collect and analyze various sensing data from sensor nodes. We ran two scenarios: in the first scenario the sensor node transmitted the information each second (Figure 6-7), while in the second the transmission occurred every 30 seconds (Figure 6-8). Since battery life is critical for low-power sensor nodes, we extracted the voltage remaining parameter from the sensor data sent to the user is PC. We noticed that the voltage dropped by 167 mV in the first case, and by 148 mV in the second case. As mentioned in chapter 1, data transmission is the major source of power consumption in the sensor node, so in considering the drops in voltage, it is clear that since the first scenario required more transmissions, it caused a faster drain of the battery. We assumed in the comparison of the energy loss between the two scenarios that energy consumption due to sensing and processing is negligible with respect to the transmission energy consumption. This assumption is based on the fact that the "Energy cost of transmitting 1 KB a distance of 100 m is approximately the same as that for executing 3 million instructions by a 100

MIPS processor" [Aky02]. In other words, and based on experimental measurements, "the communications power is twice greater than the computational power" [Aky02].

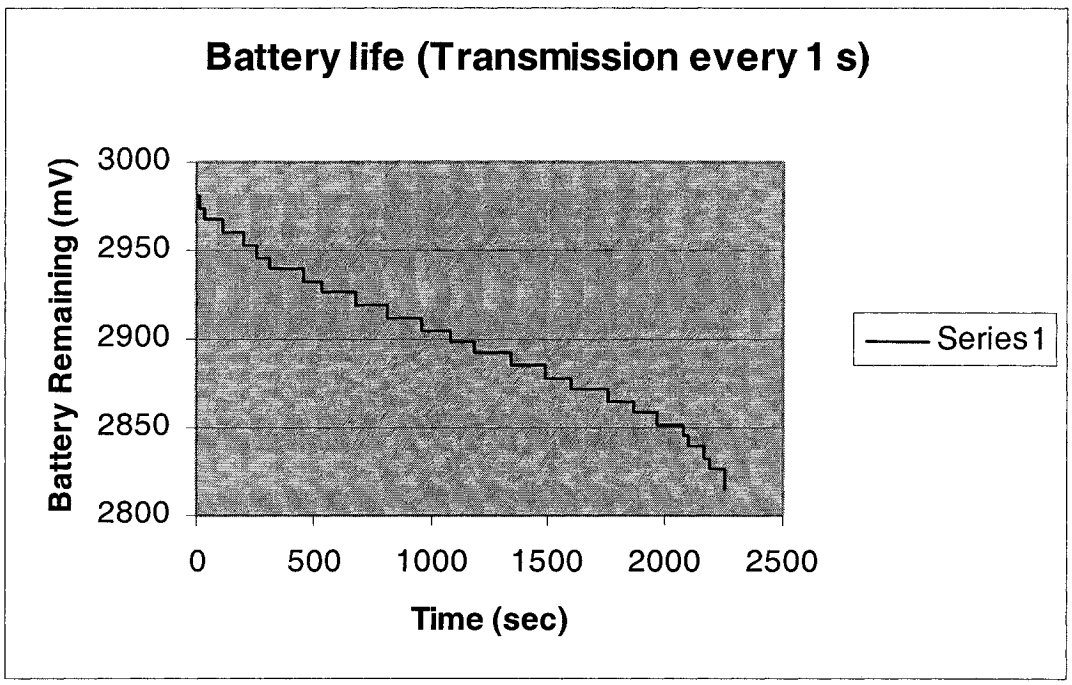

Figure 6-7: Battery Life (Transmission every 1s).

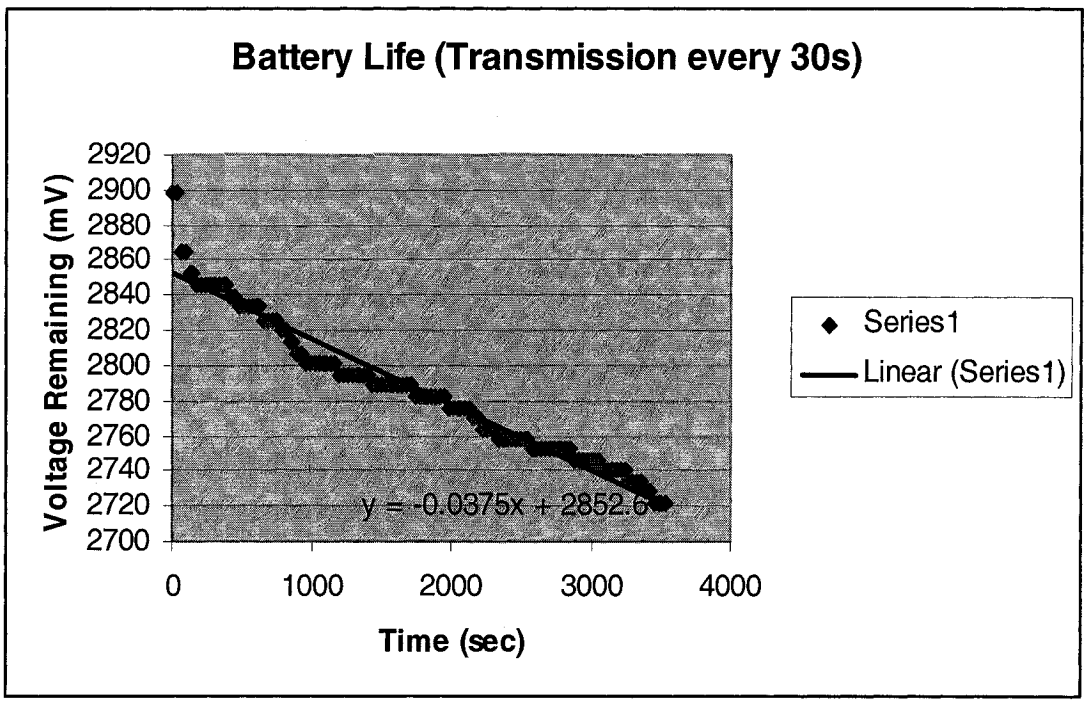

Figure 6-8: Battery Life (Transmission every 30s)

The experiment was conducted in the center of the office where the distance between the node and the base station was around 50' and the transmission power in the above two cases was set in the "MakeXbowlocal" file to "TXPOWER_MAX". We changed the transmission power to "TXPOWER_MIN" and collected data for one hour. Figure 6-9 shows the graph and linear fit for the drop in voltage for transmitting a packet every 30 seconds.

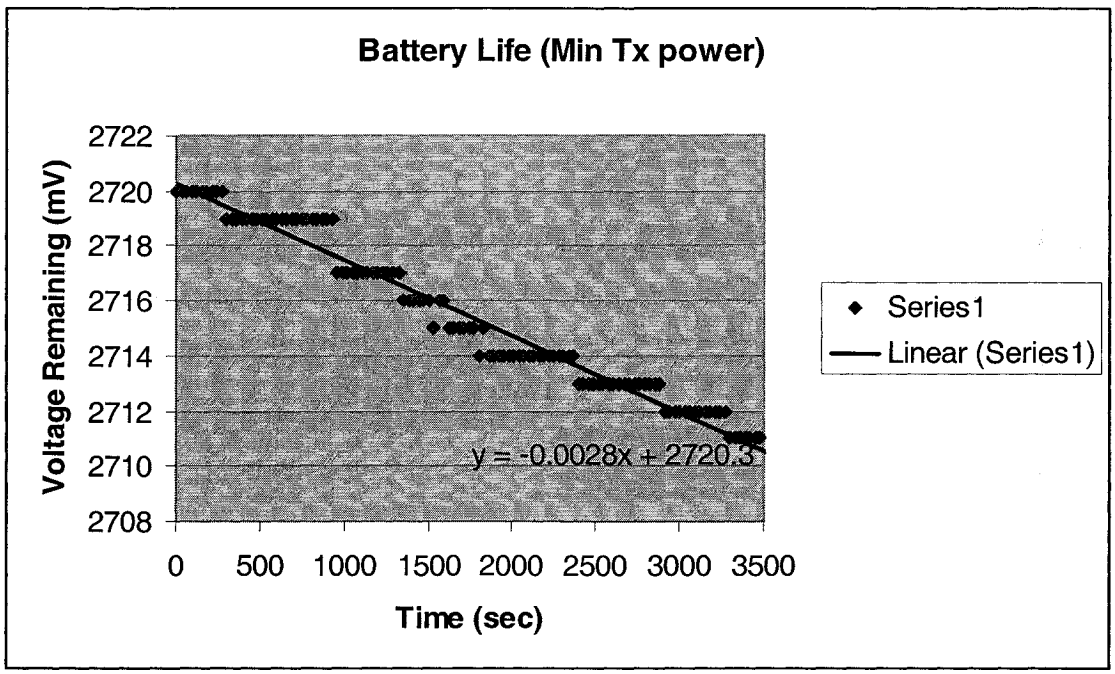

Figure 6-9: Battery Life (Min Tx power)

Figure 6-10 shows the difference in the drop in voltage when the transmission power is set to maximum and minimum respectively. In the first case, the loss rate is 0.0375 mV per second, and 0.0028 mV per second in the second case.

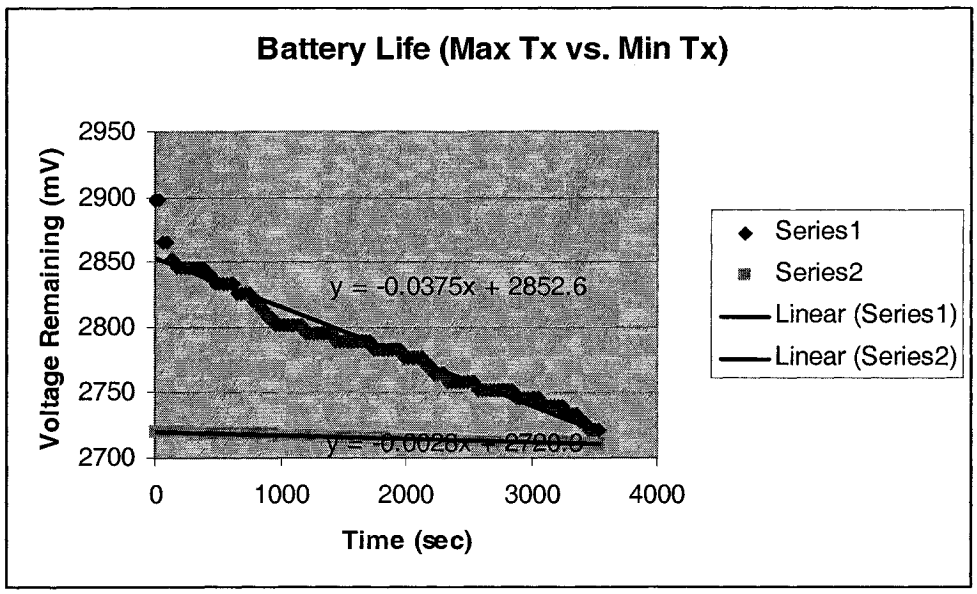

Figure 6-10: Battery Life (Min TX power vs. Max TX power)

6.6.2 Comparison with Crossbow experiment: In the following, we will compare the drop in voltage for running a slightly modified "CntToLedsAndRfm" crossbow application on a mica2 sensor node for 5 hours with the results obtained from the battery life tests for the mica2 node done by Crossbow [Cro04]. The "CntToLedsAndRfm" application displays the lower three bits of the counter value on its LEDs and transmits a TOS packet every four seconds. We set up the experiments in the same environment as the Crossbow experiment:

Transmission power = 0dBm,

Distance between Mica2 Sensor node and Base Station = 30 meters.

Two fully charged batteries.

Temperature = 25 degree Celsius (In our case, it was around 23~24°)

The only difference between the experiments is that Crossbow ran the experiment for 172 hours, which was not possible in our case, so we simply compared the graphs for the first

five hours. Figure 6-11 shows the graph for the drop in voltage during the five hours of the experiment:

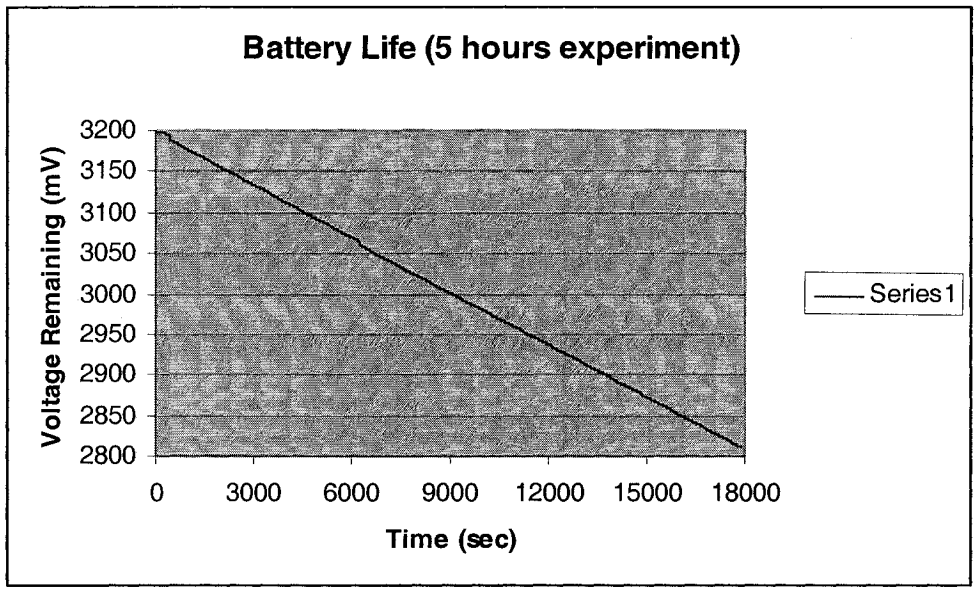

Figure 6-11: Battery Life (five hours experiment)

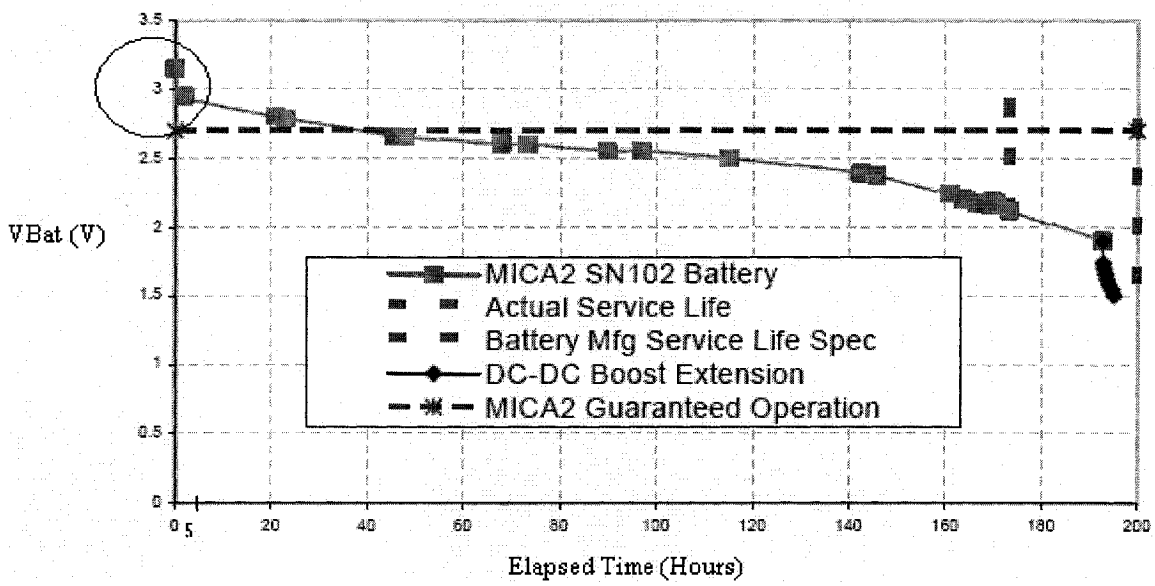

Figure 6-12: Crossbow Experiment (for 200 hours) [Cro04]

Chapter 7 : Conclusion and Future Research

7.1 Summary and Conclusive Remarks

We have described our experience in creating a heterogeneous sensor network test-bed that will be used for the performance benchmarking of existing and novel sensor network management protocols. Building this test-bed required us to interact with many tools such as NIOS II Development board, wireless sensor networks Crossbow products, Bluetooth kit from Zeevo, etc. As well, we had to solve a number of hard and interesting problems, particularly with respect to the inter-task communication between the various developed tasks on the NIOS II processor as well as the programming of the Mica2 sensor node. Scalability is guaranteed due to two main causes. First is the ability to include more soft-core processors in the on-chip system to help with hardware blocks monitoring and communication with the GUI. Second, the existence of free Hardware Logic Elements and interfaces allows us to program any custom or standard wireless communication means with VHDL/Verilog, which enables us to test sensor networks formed by heterogeneous nodes with different communication standards as long as they work on dissimilar frequencies. Furthermore most of the communications of the NIOS System are performed on-chip and in our design, this will increase system performance compared to inter-chip communication.

7.2 Future Research

Wireless sensor networks are becoming the focus of many research projects. Unfortunately, few tools exist for managing such types of networks, since they have many unique characteristics that differentiate them from other wired and wireless networks. A lot of features can be added to our test-bed and to other existing test-beds such as giving more capabilities to the administrator i.e. by enabling he/she to manage security keys, deal with mobile sensor nodes (nodes mounted on robots), and re-configure the sensor network. Another important improvement to the test-bed lies in the integration of web services, in which an XML description of the information is sent to the user who has the right to accept the data or refuse them based on his/her needs. Also an improvement can be made to the "Sensor Management Interface" by making it downloadable through a URL so that it is easy to access all over the world. Such features may require enabling secure login to the application.

References

[Abd05] Abdo, A.; Hall, T.; "***Programmable Traffic Generator with Configurable Stochastic Distributions***", Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE 2005), May 2005 Saskatoon, Canada, pp. 747 – 750.

[Aky02] Akyildiz, L.; Weilian S.; Sankarasubramaniam, Y.; Cayirci, E.;"***A survey on sensor networks***", IEEE Communications Magazine, Volume 40, Issue: 8, Aug. 2002, pp. 102 – 114.

[Alt04a] "Nios Development Kit - Cyclone Professional Edition", Altera, Aug. 2004.

[Alt04b] "Nios II Software Developer's Handbook", Altera, December 2004.

[Bud01] Budhaditya, D.; Sudeept, B.; Nath, B.;" ***A Topology Discovery Algorithm for Sensor Networks with Applications to Network Management***", DCS Technical Report DCS-TR-441, Rutgers University May 2001.

[Cer03] Cerpa, A.; Busek, N.; Estrin, D.; "***Scale: A tool for Simple Connectivity Assessment in Lossy Environments***", Center for Embedded Networked Sensing, U of C, Los Angeles, Tech. Rep. CENS Technical Report 0021, September 2003.

[Che99] Chen, W.; Jain, N.; Singh, S.; "***Anmp: Ad Hoc Net-work Network Management Protocol***", IEEE Proceedings of the JSAC, Volume 17, Issue: 8, August 1999, pp. 1506 – 1531.

[Cho03] *Chong, C.; Kumar, S.P.,* ***"Sensor networks: evolution, opportunities, and challenges"***, IEEE Proceedings, Volume: 91, Issue: 8, Aug. 2003, pp. 1247 – 1256.

[Chi89] Chiou, G.; Chen, W.;"***Secure broadcasting using secure lock***", Transactions on Software Engineering, Aug. 1989, pp. 929–933.

[Chu05] Chun, B.N; Buonadonna, P.; AuYoung, A.; Parkes, C.; Shneidman, J.; Snoeren, A.C.; Vahdat, A.; "***Mirage: A Microeconomic Resource Allocation System for SensorNet Testbeds***", Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors, Sydney, Australia, May 2005, pp. 19-28.

[Cro05a] "***www.xbow.com/Support/manuals.htm***", 2005 Crossbow Technology, Last viewed: 10 September 2006.

[Cro05b] Crossbow; "***Wireless Systems for Environmental Monitoring***", Smart Dust Application Note.

[Cro04] Suh, J.; "MICA2 ***AA Battery Pack Service Life Test***", Crossbow Technology Inc., 2004.

[Dun01] Dunkels, A.; "***Minimal TCP/IP implementation with proxy support***", Master thesis, SICS - Swedish Institute of *Computer* Science, 2001.

[Ecl06] "***www.eclipse.org***", The Eclipse Foundation, September 10, 2006.

[Est02] *Estrin, D.; Culler, D.; Pister, K.; Sukhatme, G.*; "***Connecting the physical world with pervasive networks***", Proceedings of the IEEE on the Pervasive Computing, March 2002, pp.: 59 – 69.

[Gay03] Gay, D.; Levis, P.; Culler, D.; Brewer, E.;"***nesC 1.1 Language Reference Manual***", Technical Report, May 2003.

[Els04] Elson, J.; Girod. L.; Estrin, D.; "***EmStar: development with high system visibility***", IEEE Wireless Communications, Volume 11, December 2004, pp. 70 – 77.

[Har05] Harold, E.;"***Java Programming Language***", 3rd Edition, Beijing, Sepastobol, CA: O'Reilly, C2005, ISBN: 0596007213.

[Hav02] Havinga, P.;"***System Architecture Specification, EYES Project Deliverable 1.1***", Technical Report, October 2002.

[Hei00a] Heinzelman, W.; Chandrakasan, A.; Balakrishnan, H.;"***Energy-efficient Communication Protocol for Wireless Micro Sensor Networks***", Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, June 2000, pp. 3005-3014.

[Hei99b] Heinzelman, W.; Kulik, J.; Balakrishnan, H.;"***Negotiation-based protocols for Disseminating Information in Wireless Sensor Networks***", Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, March 1999, pp. 169 – 185.

[Hil03] Hill, J.; "***System Architecture for Wireless Sensor Networks***", Ph.D. dissertation, University of California, Berkely, Spring 2003.

[Hob00] Hoblos, G.; Staroswiecki, M.; Aitouche, A.;"***Optimal Design of Fault Tolerant Sensor Networks***", Proceedings of the IEEE Intellectual Conference, Anchorage, AK, September 2000, pp. 467–472.

[Ily02] Ilyas, M. (Editor), "***Handbook of Ad Hoc Wireless Networks***", CRC Press, 2002.

[Int06] "***www.intersema.ch***", Pressure Sensor Products, Last viewed: 10 September 2006.

[Joh05] Johnson, D.; Stack, T.; Fish, R.; Flickinger, D.; Ricci, R.; Lepreau, J.;" ***TrueMobile: A Mobile Robotic Wireless and Sensor Network Testbed***", Flux Technical Note FTN-2005-02, University of Utah, April 2005.

[Kac02] Kachirski, O.; Guha, R.; "***Intrusion detection using Mobile Agents in Wireless Ad Hoc networks***", Proceedings of Knowledge Media Networking, July 2002, pp.153 – 158.

[Kum03] Kumar, M.; ***"A Consensus Protocol for Wireless Sensor Networks"***, Master thesis, Graduate School of Wayne State University, Michigan, 2003.

[Lab02] Labrosse, J.; "***MicroC/OS-II, the Real-Time Kernel***", 2nd EDITION, CMPBooks, Kansas, 2002, ISBN: 1-57820-103-9

[Lan03] Langendoen, K.; Reijers, N.;"***Distributed localization in wireless sensor networks :a quantitative comparison***", International Journal of Computer and Telecommunications Networking, Volume 43, November 2003, pp. 499–518.

[Lew01] Lewis, J.; Loftus, W.;"***Java Software Solutions: foundations of program design"***, 2nd EDITION, 2001, ISBN: 0-201-72597-5.

[Lea06] "***http://www.leadtek.com/newGPS/GPS_Main_1.htm***", last viewed: 10 September 2006.

[Mai02] Mainwaring, A.; Culler, D.; Polastre, J.; Szewczyk, R.; Anderson, J.; ***"Applications and OS: Wireless sensor networks for habitat monitoring"***, Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, September 2002, pp. 88 – 97.

[Mig03] Migas, N.; Buchanan, W.J.; McArtney, K.A.; ***"Mobile agents for routing, topology discovery, and automatic network reconfiguration in ad-hoc networks"***, Proceedings of the 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, April 2003, pp. 200 – 206.

[Mye04] Myers, G.;"***The Art of Software Testing"***, 2nd EDITION; Revised and updated by Tom Badgett and Todd Thomas, with Corey Sandler, 2004, ISBN: 0-471-46912-2.

[Mys06] MySQL;"***MySQL 3.23, 4.0, 4.1 Reference Manual"***, from http://downloads.mysql.com/docs.

[Par06] Park, S.; Kim, J.; Lee. K.; Shin, K.; Kim, D.;"***Embedded sensor networked operating system",*** Proceedings of the 9th IEEE International Symposium Object and Component-Oriented Real-Time Distributed Computing, April 2006, pp. 1-5.

[Qi06] Qi, H.; Xu, Y.; Wang, X.; "**Mobile-agent-based Collaborative Signal and Information Processing in Sensor Networks**", IEEE Proceedings, Volume 91, August 2003, pp. 1172 – 1183.

[Qua05] "***Quartus II Development Software Handbook***", Altera Corp., 2005.

[Rag04] Raghematan, V.; Schurgers, C.;"***Energy Efficient Design of Wireless Sensor Networks***", Kluwer Academic Publishers, USA , 2004, pp. 51 – 69.

[Ren01] Reynold Electronic's TWS-434 combo kit, data sheet. Available at: www.rentron.com

[Rui03] Ruiz, L.; Nogueira, J.; Loureiro, A.; "***MANNA: Management Architecture for Wireless Sensor Networks***", IEEE Communications Magazine, Volume 41, February 2003, pp. 116 – 125.

[Sen06] Sensirion; "***http://www.sensirion.com***", last viewed: 15 September 2006.

[She02] Shen, C.; Jaikaeo, C.; Srisathapornphat, C.; Huang, Z.;"***The Guerrilla management architecture for ad hoc networks***", Proceedings of the IEEE of MILCOM 2002, October 2002, pp. 467 – 472.

[Soh00] Sohrabi, K.; Gao, J.; Ailawadhi, V.; Pottie, G.J.;"***Protocols for self-organization of a wireless sensor network***", IEEE Personal Communications, Volume 7, October 2000, pp. 16 – 27.

[Sta06] Stankovic, J.;"***Wireless Sensor Networks***", Department of Computer Science, University of Virginia, June 2006.

[Tao06] Texas Advanced Optoelectronic Solutions; "***http://www.taosinc.com***", Last viewed: 15 September 2006.

[Tho05] Thorn, J.; "***Deciphering TinyOS Serial packets***", Octave Tech Brief #5-01, March 2005.

[Wan06] Wang, Y.; Wu, H.;"***DFT-MSN: The Delay/Fault-Tolerant Mobile Sensor Network for Pervasive Information Gathering***", Proceedings of the IEEE of *INFOCOM'06*, Barcelona, Spain, April 2006, pp. 120-126.

[Wel03] Welsh, E.; Fish, W.; Frantz, P.;"***GNOMES: A Test-bed for Low-Power Heterogeneous Wireless Sensor Networks***", Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Bangkok, Thailand, May 2003, pp. 836 - 839.

[Wer05] Werner-Allen, G.; Swieskowski, P.; Welsh, M, "***MoteLab: a wireless sensor network test-bed***", IEEE Proceedings of Information Processing in Sensor Networks, April 2005, pp. 483 – 488.

[Wir03] WIROBOT,"***MCB3100 WiRobot Serial Bluetooth Wireless Module User Manual",*** Version: 1.0.1, November 2003.

[Yao02] Yao, Y.; Gehrke, J.;"***Articles: The cougar approach to in-network query processing in sensor networks***", Record ACM SIGMOD, Volume 31, September 2002, pp. 9 – 18.

[Yu06] Yu, M.; Mokhtar, H.; Merabti, M.;"***A Survey of Network Management Architecture in Wireless Sensor Network***", Proceedings of the 6th Annual Postgraduate Symposium on the Convergence of Telecommunications, June 2006, pp. 202-208.

Appendix A: Localization Algorithms

In order for a sensor to provide context aware services, the sensing data it has gathered has to be complimented by the position of the sensor. It would be futile to gather data from a sensor when the location of that sensor cannot be determined accurately. Some of the mechanisms used for localization are classified below:

- GPS based Localization: GPS would be utilized by the sensors to correctly determine their position. This information would be sent along with the sensing data for aggregation. This approach, however, assumes that there is a GPS system embedded within the sensor node, and that they can interpret GPS information to correctly locate neighboring nodes.
- Relative based Localization: In this approach, each node would record its movement relative to a universal map. An update of its recent motion would be tagged along with any data sent. This approach, however, assumes that the sensors are aware of their initial deployment position, which is not always true.
- Two-phase sensing based Localization: In this approach, there would be another set of sensors that record the positions of the first set. This means that the data sensing nodes would be themselves sensed for localization by one or more sensors. An example of an application of this system would be a military airplane deploying a set of sensors and hovering around the monitored area to act as a base station for the sensors while monitoring their positions at the same time.

Many algorithms were designed for the purpose of determining the position of the sensor nodes without relying on external infrastructure. The authors in [Lan03] present

three of these algorithms, shown in Table A-1: Localization Algorithms 1. They follow the same steps but with different implementations:

Phase	Ad-Hoc positioning	Robust positioning	N-hop multilateration
1. Distance	Euclidean	DV-hop	Sum-dist
2. Position	Lateration	Lateration	Min-max
3. Refinement	No	Yes	Yes

Table A-1: Localization Algorithms 1

When we specify a number of nodes with predetermined location (called anchors), the position of the other nodes in the sensor field will be determined depending on the anchors' positions following three steps:

Step 1: Derive the distances of a node from a number of anchors.

Step 2: Derive the position of the node depending on the distances obtained in step 1.

Step 3: Refine the result.

The distances from the anchors can be implemented using various methods:

- **Sum-dist**: Anchor nodes flood information into the network. Each node receiving the information determines its distance from the anchor by adding the ranges at each hop. The main drawback of this approach is that errors are accumulated at each hop.
- **DV_hop**: The distance from the anchor is computed by multiplying the number of hops by the average hop distance where average hop distance = distance between two anchors/number of hops between these 2 anchors.

The main drawback of this approach is its inefficiency in highly irregular networks.

- **Euclidean:** In this approach, when a node receives messages from two neighbors aware of their distances from an anchor and from each other, it calculates its distance using:

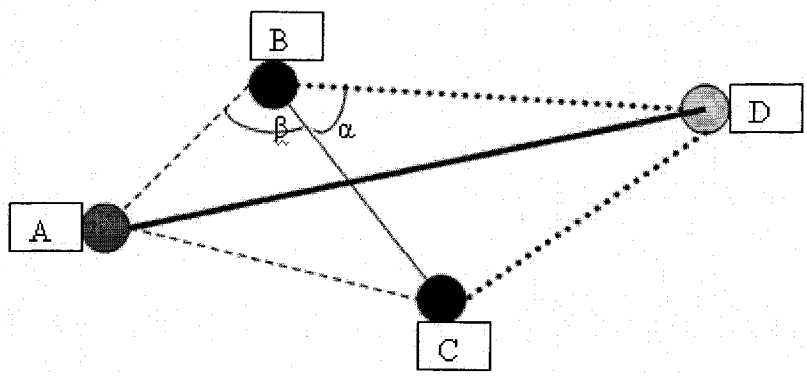

Figure A0-1: Euclidean Approach

A: An Anchor
B & C: 2 nodes that know their distances from the anchor.
D: Node with the unknown distance to the anchor

In the figure above, the distances AB, AC, BC, BD, and CD are known. We need to calculate AD.

$CD^2 = BD^2 + BC^2 - 2*BD*BC* \cos(\alpha)$; α can be calculated
$AC^2 = AB^2 + BC^2 - 2*AB*BC* \cos(\beta)$; β can be calculated

$\Gamma = \alpha + \beta$;
$AD^2 = AB^2 + BD^2 - 2*AB*AD* \cos(\Gamma)$;

The above follows the same concept as the Euclidean approach presented in [Langendoen 2003] but with a different geometric solution. The nodes determine their position based on the distances calculated in step 1 compared to a number of anchors using Lateration or Min-max.

- **Lateration**: Assume a node with coordinate (x, y) knows its distances d_n from n anchors with position (x_i , y_i); then it satisfies the equations:

$$(x_1 - x)^2 + (y_1 - y)^2 = d_1^2$$
$$\vdots$$
$$(x_n - x)^2 + (y_n - y)^2 = d_n^2$$

  Solving this system, x and y can be obtained.

- **Min-max**: A bounding box as the one shown in Figure A0-1 can be constructed from the distances to the anchors derived in step 1.

Appendix B

Figure B0-1 and Figure B0-2 show the wiring of the different components used in the applications TOSBase and Network_MTS420 respectively.

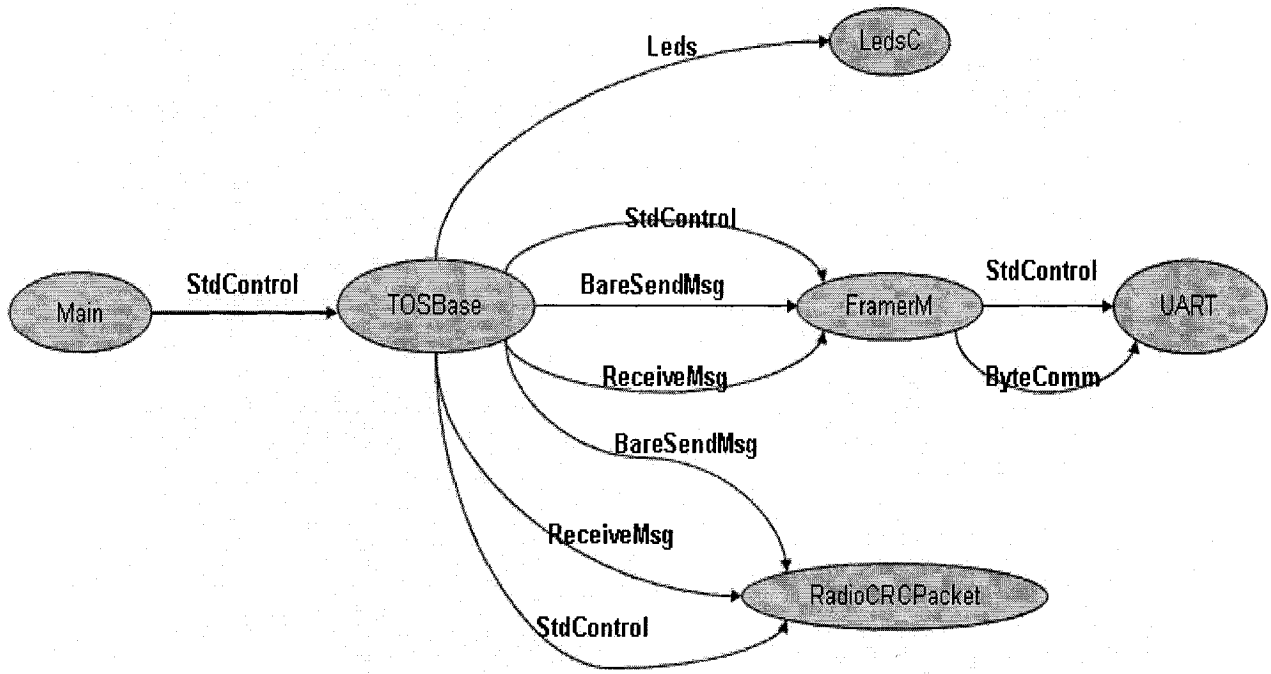

Figure B0-1: TOSBase Components wiring

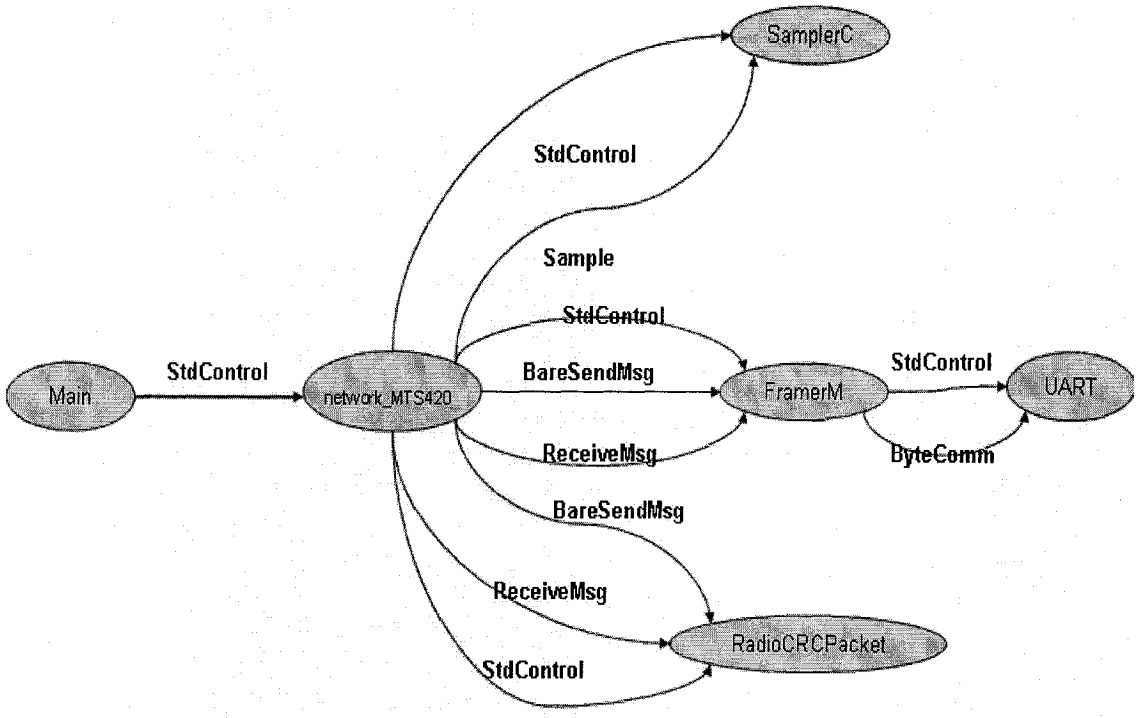

Figure B0-2:Network_MTS420 Components wiring