# APPLICATION OF WIRELESS SENSOR NETWORKS FOR ENVIRONMENTAL MONITORING AND DEVELOPMENT OF AN ENERGY EFFICIENT CLUSTER BASED ROUTING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

**Master of Technology**
**In**
**Electrical Engineering**

By

**ROHIT VAISH**

Under the guidance of

**Prof. J.K. SATAPATHY**

**Department of Electrical Engineering**

**National Institute of Technology**

**Rourkela**
2008-2009

# Application of Wireless Sensor Networks for Environmental Monitoring & Development of an Energy Efficient Hierarchical Cluster based Routing

*A Thesis Submitted in Partial Fulfillment for the Award of the Degree*

*Of*

**MASTER OF TECHNOLOGY**
*In*
**ELECTRICAL ENGINEERING**

*By*

*Rohit Vaish*

Electrical Engineering Department
National Institute of Technology
Rourkela 769008

**2008-2009**

# National Institute of Technology
# Rourkela

## CERTIFICATE

This is to certify that the thesis entitled "Application of Wireless Sensor Networks for Environmental Monitoring & Development of an Energy Efficient Hierarchical Cluster based Routing" submitted by Mr. ROHIT VAISH in partial fulfillment of the requirements for the award of Master of Technology Degree in Electrical Engineering with specialization in "Electronics System & Communication" at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Prof. J.K. SATAPATHY
National Institute of Technology
**Rourkela-769008**

# ACKNOWLEDGEMENT

I would like to express my deep sense of profound gratitude to my honorable, esteemed guide, Prof. J.K. SATAPATHY for his guidance and constant support. Over the time he has introduced me to the academic world. His perspective on my work has inspired me to go on. I am glad to work with him.

I extend my thanks to our HOD, Dr. B.D.SUBUDHI for his valuable advices and encouragement.

I would like to thank all my friends and especially ESC friends for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious.

I cannot end without thanking my lovely family, on whose encouragement, support, and love, I have relied throughout my studies.

I would like to thank to all those who are directly or indirectly supported me in carrying out this thesis work successfully.

ROHIT VAISH
Roll No. 207EE106
Department of Electrical Engg.

# CONTENTS

PAGE NO.

## 1. INTRODUCTION TO WIRELESS SENSOR NETWORKS

## 2. LITERATURE SURVEY

## 3. SOFTWARE AND PROGRAMMING DESCRIPTION

## 4. SENSOR KIT DETAILS ( HARDWARE DESCRIPTION)

## 5. FUNDAMENTALS OF CLUSTER BASED ROUTING

# ABSTRACT

Wireless Sensor Networks (WSNs) have attracted the attention of many researchers. Wireless Sensor Networks (WSNs) are used for various applications such as habitat monitoring, automation, agriculture, and security. Since numerous sensors are usually deployed on remote and inaccessible places, the deployment and maintenance should be easy and scalable. Wireless sensor network consists of large number of small nodes. The nodes then sense environmental changes and report them to other nodes over flexible network architecture. Sensor nodes are great for deployment in hostile environments or over large geographical areas.

The measurement of temperature & light by the use of Crossbow sensor kit in which there are different nodes/motes placed at different locations. These nodes are having different node identification & they will sense the temperature & light of there surrounding location and send it to the base station node which is connected through USB port to the computer by the use of MoteView & MoteConfig environment. The data acquisition board that we have used is MDA100CB (Mote Data Acquisition). The programming of the sensor nodes is done by MoteConfig & live data is viewed through MoteView environment. The nodes that we have used are MicaZ, the MDA100CB board is fixed over these nodes by means of 51 Input/output pins.

An energy efficient hierarchical cluster-based routing protocol for continuous stream queries in WSN. We introduce a set of cluster heads, head-set, for cluster-based routing. The head-set members are responsible for control and management of the network. On rotation basis, a head-set member receives data from the neighboring nodes and transmits the aggregated results to the distant base station. For a given number of data collecting sensor nodes, the number of control and management nodes can be systematically adjusted to reduce the energy consumption, which increases the network life. Nodes in a sensor network are severely constrained by energy, storage capacity and computing power. To prolong the lifetime of the sensor nodes, designing efficient routing protocols is critical.

# LIST OF FIGURES

Page No.

# LIST OF TABLES

# Chapter 1

## INTRODUCTION TO WIRELESS SENSOR NETWORKS

## 1.1 Introduction

Wireless sensor networks have recently come into prominence because they hold the potential to revolutionize many segments of our economy and life, from environmental monitoring and conservation, to manufacturing and business asset management, to automation in the transportation and health care industries. The design, implementation, and operation of a sensor network requires the confluence of many disciplines, including signal processing, networking and protocols, embedded systems, information management and distributed algorithms. Such networks are often deployed in resource-constrained environments, for instance with battery operated nodes running untethered. These constraints dictate that sensor network problems are best approached in a hostile manner, by jointly considering the physical, networking, and application layers and making major design trade-offs across the layers.

Advances in wireless networking, micro-fabrication and integration (for examples, sensors and actuators manufactured using micro-electromechanical system technology, or MEMS), and embedded microprocessors have enabled a new generation of massive-scale sensor networks suitable for a range of commercial and military applications. The technology promises to revolutionize the way we live, work, and interact with the physical environment. In a typical sensor network, each sensor node operates untethered and has a microprocessor and a small amount of memory for signal processing and task scheduling. Each node is equipped with one or more sensing devices such as acoustic microphone arrays, video or still cameras, infrared (IR), seismic, or magnetic sensors. Each sensor node communicates wirelessly with a few other local nodes within its radio communication range.

Sensor networks extend the existing Internet deep into the physical environment. The resulting new network is orders of magnitude more expansive and dynamic than the current TCP/IP networks and is creating entirely new types of traffic that are quite different from what one finds on the Internet now. Information collected by and transmitted on a sensor network describes conditions of physical environments for example, temperature, humidity, or vibration  and requires advanced query interfaces and search engines to effectively support user-level functions. Sensor networks may inter-network with an IP core network via a number of gateways. A gateway routes user queries or commands to appropriate nodes in a sensor network. It also routes sensor data, at times aggregated and summarized, to users who have requested it or are expected to utilize the information. A data repository or storage service may be present at the gateway, in addition to data logging at each sensor. The repository may serve as an intermediary between users and sensors, providing a persistent data storage. It is well known that communicating 1 bit over the wireless medium at short ranges consumes far more energy than processing that bit.

The information management and networking for sensor networks will require more than just building faster routers, switchers, and browsers. A sensor network is designed to collect information from a physical environment. In many applications, it is more appropriate to address nodes in a sensor network by physical properties, such as node locations or proximity, than by IP addresses. How and where data is generated by sensors and consumed by users will affect the way data is compressed, routed, and aggregated. Because of the peer-to-peer connectivity and the lack of a global infrastructure support, the sensors have to rely on discovery protocols to construct local models about the network and environment.

Wireless sensor networks are a trend of the past few years, and they involve deploying a large number of small nodes. The nodes then sense environmental changes and report them to other nodes over a flexible network architecture. Sensor nodes are great for deployment in hostile environments or over large geographical areas. The sensor nodes leverage the strength of collaborative efforts to provide higher quality sensing in time and space as compared to traditional stationary sensors, which are deployed in the following two ways:

- Sensors can be positioned far from the actual phenomenon, i.e. something known by sense perception. In this approach, large sensors that use some complex techniques to distinguish the targets from environmental noise are required.
- Several sensors that perform only sensing can be deployed. The position of the sensors and communications topology is carefully engineered. They transmit time series of the sensed phenomenon to central nodes where computations are performed and data are fused.

A wireless sensor network is a collection of nodes organized into a cooperative network. Each node consists of processing capability (one or more microcontrollers, CPUs or DSP chips), may contain multiple types of memory (program, data and flash memories), have a RF transceiver (usually with a single Omni-directional antenna), have a power source (e.g., batteries and solar cells), and accommodate various sensors and actuators. The nodes communicate wirelessly and often self-organize after being deployed in an ad hoc fashion. Currently, wireless sensor networks are beginning to be deployed at an accelerated pace. It is not unreasonable to expect that in 10-15 years that the world will be covered with wireless sensor networks with access to them via the Internet. This can be considered as the Internet becoming a physical network. Wireless Sensor Network is widely used in electronics. This new technology is exciting with unlimited potential for numerous application areas including environmental, medical, military, transportation, entertainment, home automation and traffic control crisis management, homeland defense, and smart spaces.

## 1.2 Wireless Sensor Network vs. Ad hoc Network

A mobile ad hoc network (MANET), sometimes called a mobile mesh network, is a self-configuring network of mobile devices connected by wireless links. Each device in a MANET is free to move independently in any direction, and will therefore change its links to other devices frequently. The difference between wireless sensor networks and ad-hoc networks are outlined below:

- The number of sensor nodes in a sensor network can be several orders of magnitude higher than the nodes in an ad hoc network.
- Sensor nodes are densely deployed.
- Sensor nodes are prone to failures.
- The topology of a sensor network changes very frequently.
- Sensor nodes mainly use broadcast communication paradigm whereas most ad hoc networks are based on point-to-point communication.
- Sensor nodes are limited in power, computational capacities, and memory.
- Sensor nodes may not have global identification (ID) because of the large amount of overheads and large number of sensors.
- Sensor networks are deployed with a specific sensing application in mind whereas ad-hoc networks are mostly constructed for communication purpose.

To summarize, the challenges we face in designing sensor network systems and applications include:-

1. **Limited hardware**: Each node has limited processing, storage, and communication capabilities, and limited energy supply and bandwidth.
2. **Limited support for networking**: The network is peer-to-peer, with a mesh topology and dynamic, mobile, and unreliable connectivity. There are no universal routing protocols or central registry services.
3. **Limited support for software development**: The tasks are typically real-time and massively distributed, involve dynamic collaboration among nodes, and must handle

multiple competing events. Global properties can be specified only via local instructions. Because of the coupling between applications and system layers, the software architecture must be codesigned with the information processing architecture.

## 1.3 Clustering in WSN (Wireless Sensor Network)

It is widely accepted that the energy consumed in one bit of data transfer can be used to perform a large number of arithmetic operations in the sensor processor. Moreover in a densely deployed sensor network the physical environment would produce very similar data in near-by sensor nodes and transmitting such data is more or less redundant. Therefore, all these facts encourage using some kind of grouping of nodes such that data from sensor nodes of a group can be combined or compressed together in an intelligent way and transmit only compact data. This can not only reduce the global data to be transmitted and localized most traffic to within each individual group, but reduces the traffic and hence contention in a wireless sensor network. This process of grouping of sensor nodes in a densely deployed large-scale sensor network is known as clustering. The intelligent way to combined and compress the data belonging to a single cluster is known as data aggregation.

There are some issues involved with the process of clustering in a wireless sensor network. First issue is, how many clusters should be formed that could optimize some performance parameter. Second could be how many nodes should be taken into a single cluster. Third important issue is the selection procedure of cluster-head in a cluster. Another issue that has been focused in many research papers is to introduce heterogeneity in the network. It means
that user can put some more powerful nodes, in terms of energy, in the network which can act as a cluster-head and other simple node work as cluster-member only. Considering the above issues, many protocols have been proposed which deals with each individual issue.

## 1.4 Energy Advantages

Because of the unique attenuation characteristics of radio-frequency (RF) signals, a multihop RF network provides a significant energy saving over a single-hop network for the same distance. Consider the simple example of an $N$-hop network. Assume the overall distance of transmission is $Nr$, where $r$ is the one-hop distance. The minimum receiving power at a node for a given transmission error rate is $P_{receive}$ , and the power at a transmission node is $P_{send}$. Then, the RF attenuation model near the ground is given by

$$P_{receive} \; \alpha \; \frac{P_{send}}{r^{\alpha}} \qquad (1.1)$$

where r is the transmission distance and α is the RF attenuation exponent. Due to multipath and other interference effects, α is typically in the range of 2 to 5. Equivalently,

$$P_{send} \; \alpha \; r^{\alpha}.P_{receive} \qquad (1.2)$$

Therefore, the power advantage of an *N*-hop transmission versus a single-hop transmission over the same distance *Nr* is

$$\eta_{rf} = \frac{P_{send}(Nr)}{N.P_{send}(r)} = N^{\alpha - 1} \qquad (1.3)$$

A larger *N* gives a larger power saving due to the consideration of RF energy alone. However, this analysis ignores the power usage by other components of an RF circuitry. Using more nodes not only increases the cost, but also the power consumption of these other RF components. In practice, an optimal design seeks to balance the two conflicting factors for an overall cost and energy efficiency.

## 1.5 Detection advantage

Each sensor has a finite sensing range, determined by the noise floor at the sensor. A denser field improves the odds of detecting a signal source within the range. Once a signal source is inside the sensing range of a sensor, further increasing the sensor density decreases the average distance from a sensor to the signal source, hence improving the signal-to-noise ratio (SNR).

## 1.6 Sensor Network Applications

A sensor network is designed to perform a set of high-level information processing tasks such as detection, tracking, or classification. Measures of performance for these tasks are well defined, including detection of false alarms or misses, classification errors, and track quality. Applications of sensor networks are wide ranging and can vary significantly in application requirements, mode of deployment (e.g., ad hoc versus instrumented environment), sensing modality, or means of power supply (e.g. battery versus wall socket). Sample commercial and military applications include:

- Environmental monitoring (e.g. traffic, habitat, security)
- Industrial sensing and diagnostics (e.g. appliances, factory, supply chains)
- Infrastructure protection (e.g. power grid, water distribution)
- Battlefield awareness (e.g. mutitarget tracking)
- Context-aware computing (e.g. intelligent home, responsive environment)

## 1.7 Motivation of the Work

Current research in the areas of wireless communications, micro-electromechanical systems and low power design is progressively leading to the development of cost effective, energy efficient, multifunctional sensor nodes. Sensing, communication, processing and battery units are the primary components of a sensor node. Individual sensors have the capacity to detect events occurring in their area of deployment.

Reliable data transport is an important facet of dependability and quality of service in several applications of wireless sensor networks. Different applications have different reliability requirements, for example an application to collect environmental parameters like temperature, humidity etc periodically can ignore an occasional loss of a value from a particular sensor but for an application in which the data collected by every sensor is a critical piece of information then end-to-end reliability has to be guaranteed for every individual packet.

Routing protocols providing an optimal data transmission route from sensor nodes to sink to save energy of nodes in the network. Data aggregation plays an important role in energy conservation of sensor network. Data aggregation methods are used not only for finding an optimal path from source to destination but also to eliminate the redundancy of data, since transmitting huge volume of raw data is an energy intensive operation, and thus minimizing the number of data transmission. Also multiple sensors may sense the same phenomenon, although from different view and if this data can be reconciled into a more meaningful form as it passes through the network, it becomes more useful to an application.

An example for an application that requires guaranteed end-to-end reliability is an integration of Radio Frequency Identification (RFID) and wireless sensor network for automated inventory management and tracking. In this application setup the sensor devices called motes are attached with RFID readers to record RFID tag information on the objects.

These sensor motes have a critical piece of information to be sent to the sink. Therefore reliable sensor-to-sink communication has to be guaranteed for such applications. This is the main motivation behind studying the various issues and strategies of reliable communication in this thesis.

## 1.8 Objective of the Thesis

- To measure or sense temperature and light parameters through Crossbow Sensor Kit by using MoteView and MoteConfig environment.
- The energy efficient hierarchical cluster-based routing for Wireless Sensor Networks (WSN).

# Chapter 2

## LITERATURE SURVEY

## 2.1 Introduction

Researchers have focused on Wireless Sensor challenges that have limited resource capabilities of the hardware i.e. memory, processing power, bandwidth and energy deposits. Much research is currently being conducted in the following areas:

- Increasing network lifetime.
- Improving reliability of data transfer.
- Finding solutions to assist easy deployment and maintenance.
- Developing techniques that will enforce secure, private and trustworthy networks.

A wireless sensor network (WSN) has important applications such as remote environmental monitoring and target tracking. This has been enabled by the availability, particularly in recent years, of sensors that are smaller, cheaper, and intelligent. These sensors are equipped with wireless interfaces with which they can communicate with one another to form a network. The design of a WSN depends significantly on the application, and it must consider factors such as the environment, the application's design objectives, cost, hardware, and system constraints. We give an overview of several new applications and then review the literature on various aspects of WSNs.

A sensor network system consisting of a large number of small sensors with low-power can be an effective tool for collection and integration of data by each sensor in a variety of environments. The collected data by each sensor node is communicated through the network to a single base station that uses all collected data to determine properties of the data. Clustering sensors into groups, yields that sensors communicate information only to cluster heads and then the clusterheads communicate the aggregated information to the base station. We estimate the optimal number of cluster-heads among randomized sensors in a bounded region. The algorithm minimize the total energy spent in the wireless sensor network when all sensors communicate data from the cluster-heads to the base station.

## 2.2 Evolution of Sensor Nodes

There has been a long history for (remote) sensing as a means for humans to observe the physical world. For example, the telescope invented in the 16th century is simply a device for viewing distant objects. As with many technologies, the development of sensor networks has been largely driven by defense applications.

## 2.3 Military Networks of Sensors

Since the early 1950s, a system of long-range acoustic sensors (hydrophones), called the Sound Surveillance System (SOSUS), has been deployed in the deep basins of the Atlantic and Pacific oceans for submarine surveillance. Beams from multiple hydrophone arrays are used to detect and locate underwater threats. Recently, SOSUS has been replaced by the more sophisticated Integrated Undersea Surveillance System.

Networks of air defense radars can be regarded as an example of networked large scale sensors. Both ground-based radar systems and Airborne Warning and Control System (AWACS) planes are integrated into such networks to provide all-weather surveillance, command, control, and communications.

Another early example of sensing with wireless devices is the Air Delivered Seismic Intrusion Detector (ADSID) system, used by US Air Force in the Vietnam War. Each ADSID node was about 48 inches in length, nine inches in diameter, and weighted 38 pounds.

## 2.4 Next Generation Wireless Sensor Nodes

### 2.4.1 WINS from UCLA

In 1996, the Low Power Wireless Integrated Microsensors (LWIMs) were produced by UCLA and the Rockwell Science Center. By using commercial, low cost CMOS fabrication, LWIMs demonstrated the ability to integrate multiple sensors, electronic interfaces, control, and communication on a single device. LWIM supported over 100 Kbps wireless communication at a range of 10 meters using a 1 mW transmitter. In 1998, the same team built a second generation sensor node the Wireless Integrated Network Sensors (WINS). Commercial WINS from Rockwell Science Center each consists of a processor board with an Intel Strong Arm SA1100 32-bit embedded processor (1 MB SRAM and 4 MB flash memory), a radio board that supports 100 Kbps with adjustable power consumption from 1 to 100 mW, a power supply board, and a sensor board. These boards are packaged in a 3.5"x3.5"x3" enclosure. The processor consumes 200 mW in the active state and 0.8 mW when sleeping.

### 2.4.2 Motes from UC Berkeley

While WINS offer relatively powerful processing and communication capabilities, other research efforts have been developing smaller and cheaper nodes with less power consumption. The Mica family was released in 2001, including Mica , Mica2, Mica2Dot, and MicaZ. While Mica still used an 8-bit 4 MHz microcontroller (ATmega103L), it offered enhanced capabilities in terms of memory and radio, compared with preceding products. Mote architecture allowed several different sensor boards, or a data acquisition board, or a network interface board to be stacked on top of the main processor/radio board. The follow ups to Mica, Mica2 and Mica2Dot were built in 2002 with an ATmega128L microcontroller that reduced standby current (33 mW active power and 75 µW sleep power). One year later, MicaZ was produced with a Chipcon CC2420 wideband radio module that supported 802.15.4 and ZigBee protocols, with a data rate up to 250 Kbps. This radio module also supported on-chip data encryption and authentication.

The latest member in the family, Telos, was released in 2004. Telos offered a set of new features: (1) a microcontroller from Texas Instruments with 3 mW active power and 15µW sleep power, (2) an internal antenna built into the printed circuit board to reduce cost, (3) an on-board USB for easier interface with PCs, (4) integrated humidity, temperature, and light sensors, and (5) a 64-bit MAC address for unique node identification. The integrated RAM and flash memory architecture has greatly simplified the design of the mote family. However, the tiny footprint also requires a specialized operating system, which was developed by UC Berkeley, called TinyOS. TinyOS features a component-based architecture and event driven model that are suitable for programming with small embedded devices, such as motes. The combination of Motes and TinyOS is gradually becoming a popular experimental platform for many research efforts in the field of WSNs.

### 2.4.3 Medusa from UCLA

The design philosophy and operational space of motes are quite different from those of WINS. On one hand, motes are designed for simple sensing and signal processing applications, where the demand for computation and communication capabilities is low. On the other hand, WINS are essentially an embedded version of PDAs, for more advanced computationally intensive applications with large memory space requirements. To bridge the gap between the two extremes, the Medusa MK-2 sensor node was developed by the Center for Embedded Networked Sensing (CENS) at UCLA in 2002. One distinguishing feature of Medusa MK-2 is that it integrates two microcontrollers. The first one, ATmega128, is dedicated to less computationally demanding tasks, including radio base band processing and sensor sampling. The second one, AT91FR4081, is a more powerful microcontroller (40 MHz, 1 MB flash, 136 KB RAM) that can be used to handle more sophisticated, but less frequent signal processing tasks (e.g., the Kalman filter). The combination of these two microcontrollers provides more flexibility in WSN development and deployment, especially for applications that require both high computation capabilities and long lifetime.

## 2.5 Why Microscopic Sensor Nodes?

The transition from large to small scale sensor nodes has several advantages.

(1) Small sensor nodes are easy to manufacture with much lower cost than large scale sensors. They are even disposable if the envisioned US$1 target price can be realized in the future.

(2) With a mass volume of such low cost and tiny sensor nodes, they can be deployed very closely to the target phenomena or sensing field at an extremely high density. Therefore, the shorter sensing range and lower sensing accuracy of each individual node are compensated for by the shorter sensing distance and large number of sensors around the target objects, which generates a high signal to noise ratio (SNR).

(3) Since computing and communication devices can be integrated with sensors, large-sample in-network and intelligent information fusion becomes feasible. The intelligence of sensor nodes and the availability of multiple onboard sensors also enhances the flexibility of the entire system.

(4) Due to their small size and self-contained power supply, sensor nodes can be easily deployed into regions where replenishing energy is not available, including hostile or dangerous environments. The survivability of nodes also increases with reduced size.

(5) The high node density enables system-level fault tolerance through node redundancy.

# Chapter 3

## SOFTWARE AND PROGRAMMING DESCRIPTION

## 3.1 Introduction

### 3.1.1 MoteConfig

MoteConfig is a Windows-based GUI (Graphical user interface) utility for programming Motes. This utility provides an interface for configuring and downloading pre-compiled XMesh/TinyOS firmware applications onto Motes. MoteConfig allows the user to configure the Mote ID, Group ID, RF channel and RF power. High-power and low-power XMesh applications are available for each sensor board and platform manufactured by Crossbow as part of the MoteView install.

Each Mote has a 512kB external non-volatile flash divided into 4 slots. These slots have a default size of 128 kB. Slot 0 is reserved for the OTAP (Over-The-Air-Programming). The Over-The-Air-Programming (OTAP) feature allows users to reprogram a Mote over a wireless link. Slots 1, 2 and 3 can be used for user-specified firmware.

During the OTAP process, the server sends a command to the Mote to reboot into the OTAP image (slot 0). A user-specified firmware image is broken up into fragments and transmitted to the Mote and stored into Slot 1, 2 or 3. The server can send a message to transfer the newly uploaded firmware into the program flash and reboot the Mote.

## 3.2 Installation

### 3.2.1 Supported Platforms

MoteConfig is supported on the following operating systems:

- Windows XP Home
- Window XP Professional
- Windows 2000 with SP4

## 3.3 Installing MoteView on a Windows PC

Before you can use MoteView you have to install it on a PC. The requirements necessary to properly install MoteView are below:

1. A PC with one of the following operating systems
    - Windows XP Home/Professional
    - Windows 2000 with SP4
2. An NTFS file system.

3. Screen resolution must be at least 800 × 600 or the interface will require scrollbars.
4. Administrative privileges to write to Windows registry.
5. Prior to installing MoteView, it is highly recommended that you shut down all the programs running on your computer.

### 3.3.1 Installation Steps:

1. Insert the MoteWorks Support Tools CDROM into the computer's CD drive.
2. Double-click on MoteView_2.0.F_Setup.exe from "**MoteView**" folder.
3. Select the desired installation directory (the default installation directory is **C:\Program Files\Crossbow\MoteView**)
4. Select all available installation tasks.
5. Install the following during installation:
    - MoteView application
    - PostgreSQL 8.0 database service
    - PostgreSQL ODBC driver
    - Microsoft .NET framework

MoteView has four main user interface sections which we can browse and use.

- **Toolbar / Menus:** Allows the user to specify actions and initiate command dialogs.
- **Node List:** Shows all known nodes in a deployment and health status summary.
- **Visualization Tabs:** Enables the user to view the sensor data in various ways.
- **Server Messages:** Displays a log of server events and incoming messages.

## 3.4 PC Interface Port Requirements

The gateway platform used in the base station determines the PC interface port required by MoteConfig.

1. For a **MIB510** serial gateway: an RS-232 serial port.
2. For a **MIB520** USB gateway: a USB port.

3. For a **MIB600** Ethernet gateway: A wired Ethernet or 802.11 wireless card (if the MIB600 is on a LAN with wireless access).

**Table 3.1 Pre-compiled MICAz XMesh applications**

| **MICAz Mote (MPR2400 and MPR2600)** | |
|---|---|
| Board Model | Binary file name |
| **MDA board** | |
| MDA100CA | *XMDA100CA_2420_<mode>.exe* |
| MDA100CB | *XMDA100CB_2420_<mode>.exe* |
| XBW-DA100CA | *XBW-DA100CA_2420_hp.exe* |
| XBW-DA100CB | *XBW-DA100CB_2420_hp.exe* |
| MDA300 | *XMDA300_2420_<mode>.exe* |
| MDA300 (precision) | *XMDA300p_2420_<mode>.exe* |
| MDA320 | *XMDA320_2420_<mode>.exe* |
| XBW-DA325 | *XDA325_2420_<mode>.exe* |
| Base Station (common to all boards) | |
| *XMeshBase_2420_<mode>.exe* | |

<mode> = hp or lp.

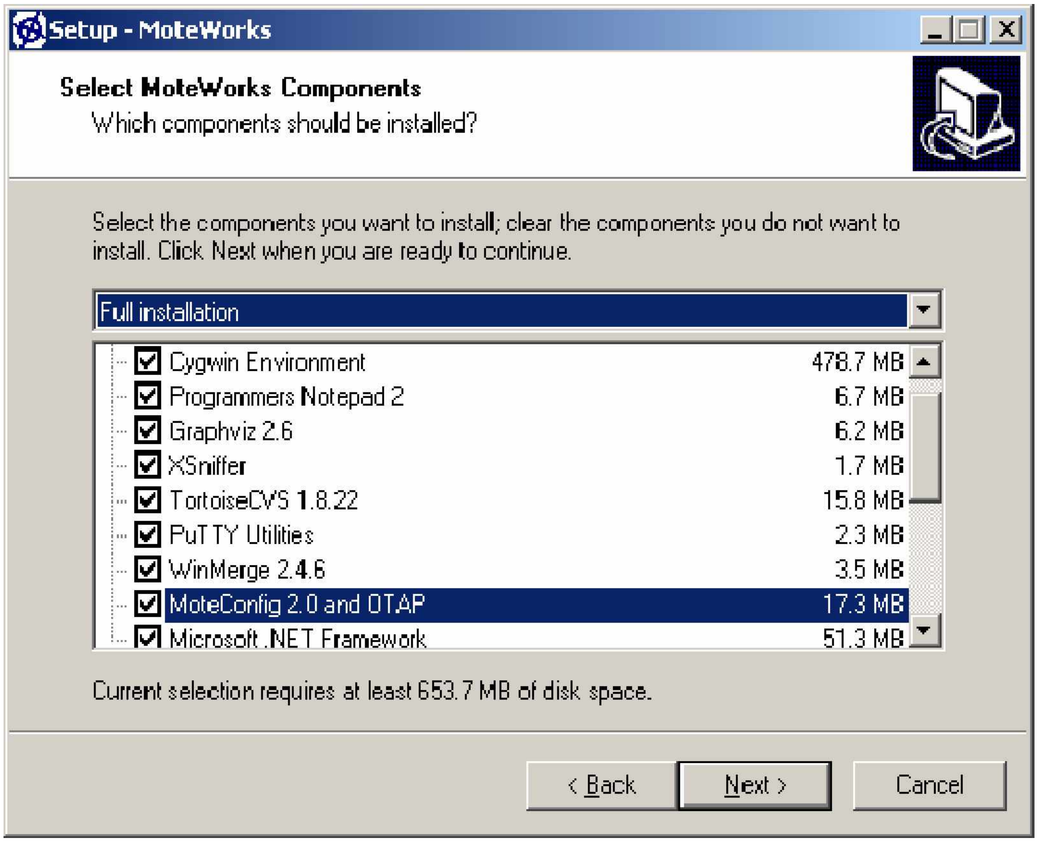
hp = high power mesh networking. lp = low-power mesh networking.

## 3.5 Installation Steps for MoteConfig

MoteConfig is shipped as a component of MoteView and MoteWorks:

1. MoteConfig is automatically installed with the MoteView installer.

2. MoteConfig is an optional component in the MoteWorks installer. Make sure that MoteConfig 2.0 and OTAP item is selected as shown in figure.



**Select MoteWorks Components**

Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Full installation

| | |
|---|---|
| ☑ Cygwin Environment | 478.7 MB |
| ☑ Programmers Notepad 2 | 6.7 MB |
| ☑ Graphviz 2.6 | 6.2 MB |
| ☑ XSniffer | 1.7 MB |
| ☑ TortoiseCVS 1.8.22 | 15.8 MB |
| ☑ PuTTY Utilities | 2.3 MB |
| ☑ WinMerge 2.4.6 | 3.5 MB |
| ☑ MoteConfig 2.0 and OTAP | 17.3 MB |
| ☑ Microsoft .NET Framework | 51.3 MB |

Current selection requires at least 653.7 MB of disk space.

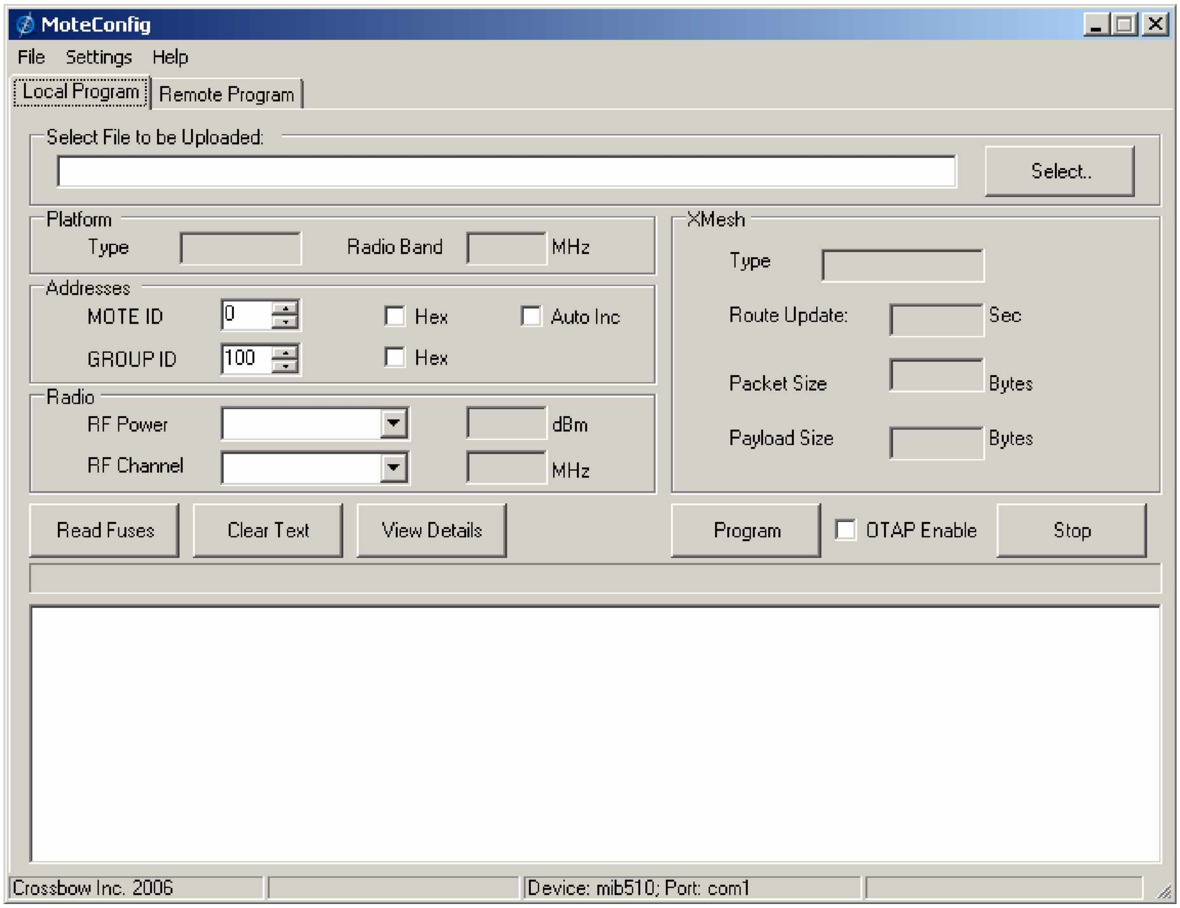
Figure 3.1 MoteConfig 2.0 and OTAP – MoteWorks Installer

## 3.6 Starting MoteConfig

If MoteConfig was installed using the MoteView installer, use the following steps:

Open MoteView 2.0F by either clicking on the shortcut located on the Desktop, or by going to Start > Programs > Crossbow > MoteView 2.0F.

- Press the Program Mote button on the MoteView toolbar to spawn the MoteConfig GUI as shown in figure.



MOTE ID 0
GROUP ID 100
Crossbow Inc. 2006
Device: mib510; Port: com1

Figure 3.2 MoteConfig Application GUI

If MoteConfig was installed using the MoteWorks installer

Click on the shortcut located on the Desktop, or select Start > Programs > Crossbow > MoteConfig 2.0.

## 3.7 Local Programming

The Local Program tab is used to upload firmware onto the Motes via a gateway.

To program motes correctly, set up the hardware as follows:

1. The gateway should be powered and connected to the PC via a serial, USB or Ethernet port.

2. If using the MIB510, the SW2 switch should be in the “OFF” position.

3. The motes should be firmly attached to the gateway.

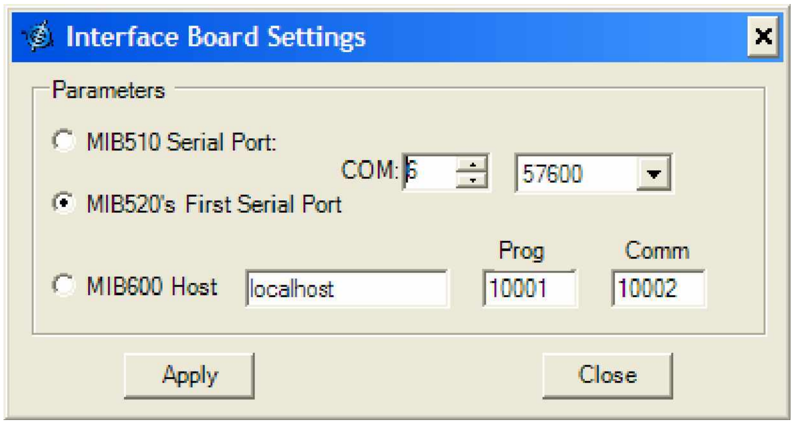
4. The motes should be turned off before the programming.

## 3.8 Settings

Click on Settings > Interface Board to select the correct gateway and port settings.

The MIB520 virtual COM port drivers will install two sequential ports on the PC. The low-numbered port is used for programming and the high-numbered port is used for communication. Figure shows the Interface Board Settings for a MIB520 that has created COM 6 and 7 on the PC. In this example, COM must be selected as the serial port.

**NOTE**: The MIB520 requires the installation of the FTDI FT2232C drivers. Once these drivers are installed, the Device Manager (Start > Control Panel > System > Hardware) will display the MIB520 as two new virtual com ports.



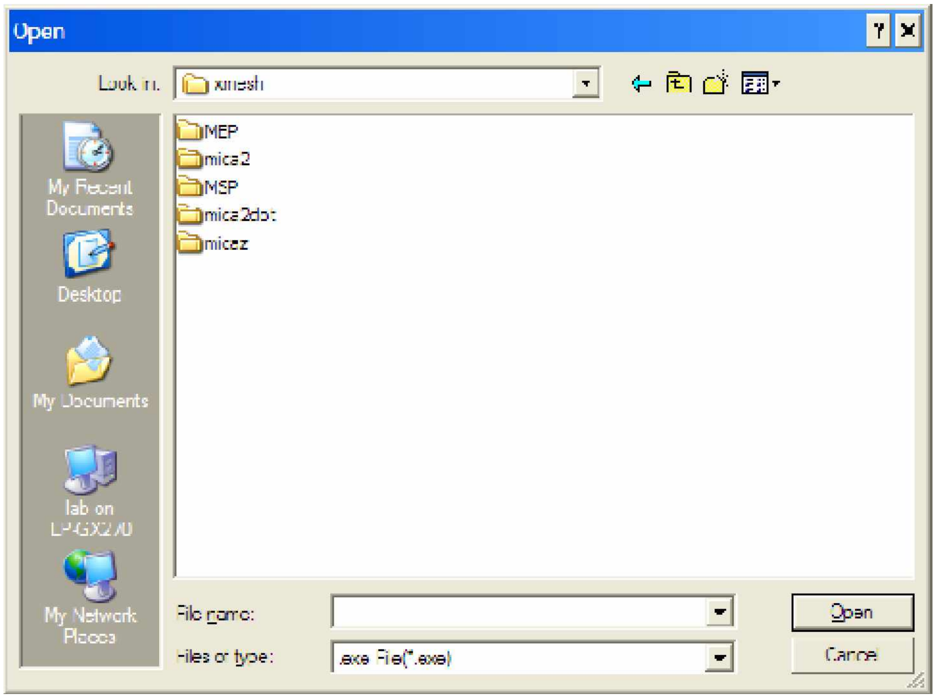
○ MIB510 Serial Port:
● MIB520's First Serial Port
COM: 6
57600
○ MIB600 Host localhost
Prog 10001
Comm 10002

Figure 3.3 MIB520 Gateway Settings

## 3.9 Programming

The pre-compiled XMesh applications installed with MoteView are located in C > Program Files > Crossbow > MoteView > XMesh.

Press the Select button to open a file browser as shown in Figure. Navigate to the folder that corresponds to your Mote processor/radio board, radio frequency (for micaz and MICA) and sensor board type.



Look in: xmesh

MEP
mica2
MSP
mica2dot
micaz

File name:

Files of type: .exe File(*.exe)

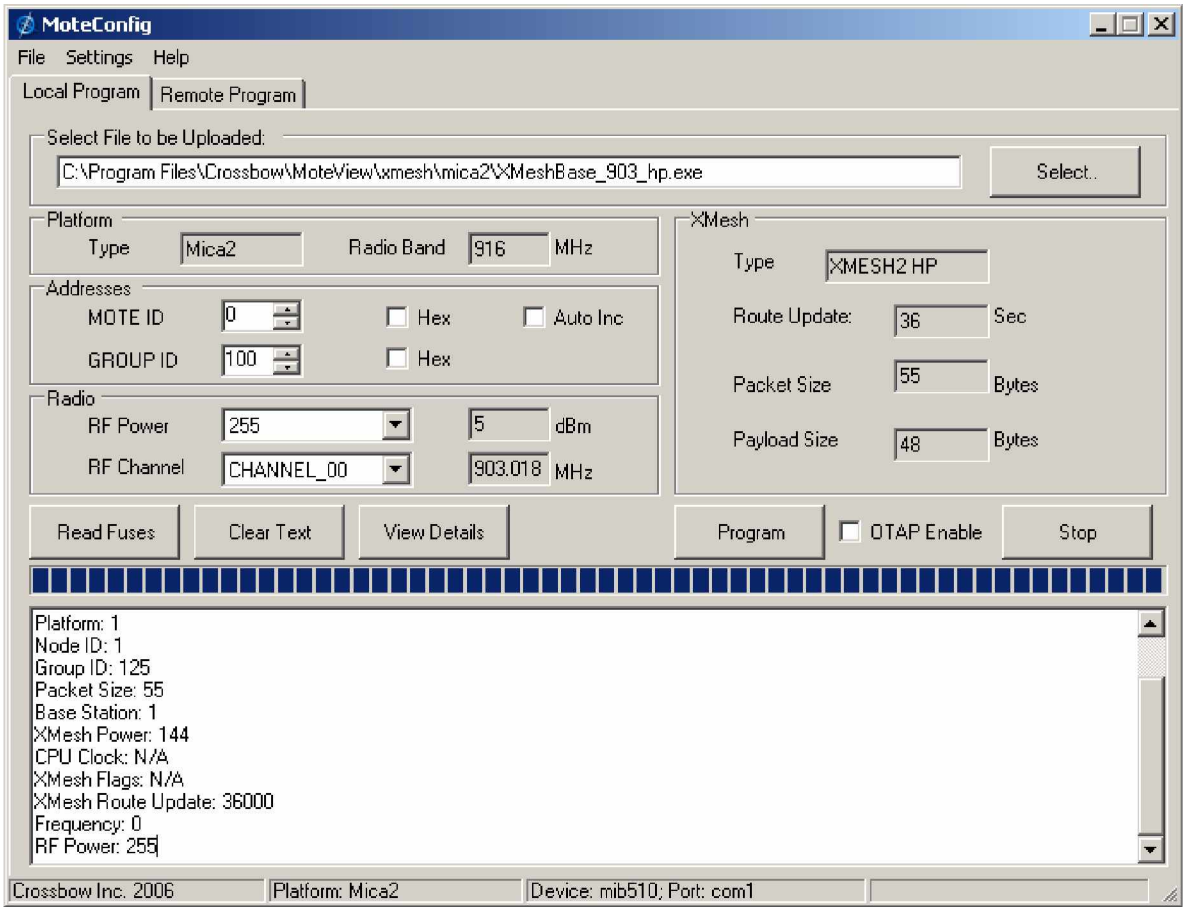
Figure 3.4 File Browser for selecting XMesh applications

Low-power and high-power applications have been included for most sensor boards.

**Note**: 1. The MEP and MSP node firmware is located in separate named folders.

2. The base station Mote must be programmed with XMeshBase_2420_<hp or lp>.exe and a node ID of 0.

After an application has been selected, the binary scan feature built into MoteConfig will display the default parameters programmed into the application (see Figure).



Select File to be Uploaded: C:\Program Files\Crossbow\MoteView\xmesh\mica2\XMeshBase_903_hp.exe

Platform
Type: Mica2
Radio Band: 916 MHz

Addresses
MOTE ID: 0 ☐ Hex ☐ Auto Inc
GROUP ID: 100 ☐ Hex

Radio
RF Power: 255 5 dBm
RF Channel: CHANNEL_00 903.018 MHz

XMesh
Type: XMESH2 HP
Route Update: 36 Sec
Packet Size: 55 Bytes
Payload Size: 48 Bytes

```
Platform: 1
Node ID: 1
Group ID: 125
Packet Size: 55
Base Station: 1
XMesh Power: 144
CPU Clock: N/A
XMesh Flags: N/A
XMesh Route Update: 36000
Frequency: 0
RF Power: 255
```

Crossbow Inc. 2006 | Platform: Mica2 | Device: mib510; Port: com1

Figure 3.5 Binary Scan Result of an XMeshBase application

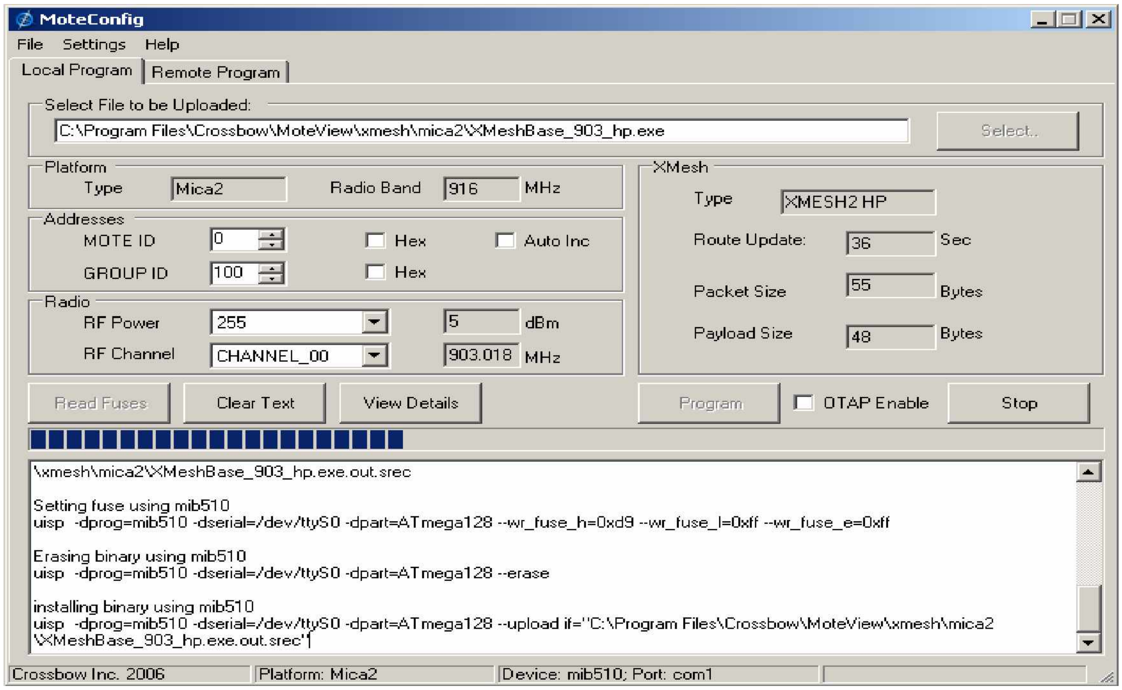
These default parameters can be overwritten by the user by specifying the desired **MOTE ID**, G**ROUP ID**, **RF Power**, and **RF Channel.**

**NOTE:** Remote nodes must be programmed with a non-zero Mote ID.

Press the **Program** button to download the selected firmware and configuration into the mote, as shown in Figure.

When programming is complete, the “Upload SUCCESSFUL!” message is printed in the status box as shown in Figure.

The **Stop** button can be used to cancel a firmware download in progress.



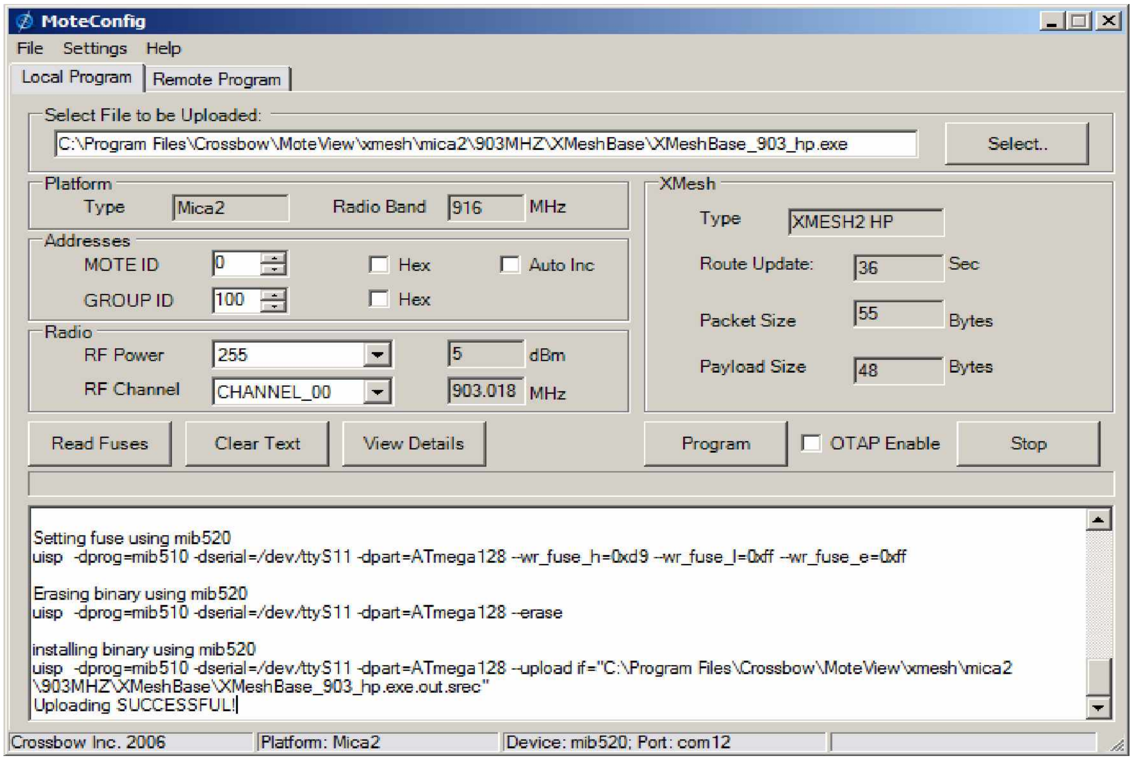
Select File to be Uploaded: C:\Program Files\Crossbow\MoteView\xmesh\mica2\XMeshBase_903_hp.exe

Type: Mica2
Radio Band: 916 MHz
MOTE ID: 0
GROUP ID: 100
RF Power: 255 | 5 dBm
RF Channel: CHANNEL_00 | 903.018 MHz
Type: XMESH2 HP
Route Update: 36 Sec
Packet Size: 55 Bytes
Payload Size: 48 Bytes

```
\xmesh\mica2\XMeshBase_903_hp.exe.out.srec

Setting fuse using mib510
uisp  -dprog=mib510 -dserial=/dev/ttyS0 -dpart=ATmega128 --wr_fuse_h=0xd9 --wr_fuse_l=0xff --wr_fuse_e=0xff

Erasing binary using mib510
uisp  -dprog=mib510 -dserial=/dev/ttyS0 -dpart=ATmega128 --erase

installing binary using mib510
uisp  -dprog=mib510 -dserial=/dev/ttyS0 -dpart=ATmega128 --upload if="C:\Program Files\Crossbow\MoteView\xmesh\mica2
\XMeshBase_903_hp.exe.out.srec"
```

Crossbow Inc. 2006 | Platform: Mica2 | Device: mib510; Port: com1

Figure 3.6 MoteConfig programming in progress



Select File to be Uploaded: C:\Program Files\Crossbow\MoteView\xmesh\mica2\903MHZ\XMeshBase\XMeshBase_903_hp.exe

Type: Mica2
Radio Band: 916 MHz
MOTE ID: 0
GROUP ID: 100
RF Power: 255 | 5 dBm
RF Channel: CHANNEL_00 | 903.018 MHz
Type: XMESH2 HP
Route Update: 36 Sec
Packet Size: 55 Bytes
Payload Size: 48 Bytes

```

Setting fuse using mib520
uisp  -dprog=mib510 -dserial=/dev/ttyS11 -dpart=ATmega128 --wr_fuse_h=0xd9 --wr_fuse_l=0xff --wr_fuse_e=0xff

Erasing binary using mib520
uisp  -dprog=mib510 -dserial=/dev/ttyS11 -dpart=ATmega128 --erase

installing binary using mib520
uisp  -dprog=mib510 -dserial=/dev/ttyS11 -dpart=ATmega128 --upload if="C:\Program Files\Crossbow\MoteView\xmesh\mica2
\903MHZ\XMeshBase\XMeshBase_903_hp.exe.out.srec"
Uploading SUCCESSFUL!
```

Crossbow Inc. 2006 | Platform: Mica2 | Device: mib520; Port: com12

Figure 3.7 MoteConfig programming successful

## 3.10 MoteView

A mesh network is a generic name for a class of networked embedded systems that share several characteristics including:

- **Multi-Hop --** the capability of sending messages peer-to-peer to a base station, thereby enabling scalable range extension.
- **Self-Configuring** -- capable of network formation without human intervention.
- **Self-Healing** -- capable of adding and removing network nodes automatically without having to reset the network.
- **Dynamic Routing --** capable of adaptively determining the route based on dynamic network conditions (e.g., link quality, hop-count, gradient, or other metric).

A wireless network deployment is composed of the three distinct software tiers:

- The **Client Tier** provides the user visualization software and graphical interface for managing the network. Crossbow provides free client software called MoteView that bundles software from all three tiers to provide an end-to-end solution.
- The **Server Tier** is an always-on facility that handles translation and buffering of data from the wireless network and provides the bridge between the wireless motes and the internet clients.
- The **Mote Tier**, where XMesh resides, is the software the runs on the cloud of sensor nodes forming a mesh network.

### 3.10.1 MoteView Overview

MoteView is designed to be an interface between a user and a deployed network of wireless sensors. MoteView provides the tools to simplify deployment and monitoring. It also makes it easy to connect to a database, to analyze, and to graph sensor readings.

In the three-part framework for deploying a sensor network system, the first part is the Mote layer or sensor mesh network. The Motes are programmed with XMesh/TinyOS firmware to do a specific task: e.g., microclimate monitoring, asset tracking, intrusion detection, etc. The second layer or Server tier provides data logging and database services. At this layer sensor readings arrive at the base station (e.g., MIB510, MIB520, MIB600, or Stargate) and are stored on a server or Stargate. The third part is the client tier in which software tools provide visualization, monitoring, and analysis tools to display and interpret sensor data. The purpose of this document is to explain the features of MoteView and to provide information on the supported Mote layer applications, Mote platforms, and sensor boards.

## 3.11 Supported Sensor Boards and Mote Platforms

MoteView supports all of Crossbow's sensor and data acquisition boards as well as the MICA2, MICA2DOT, and MICAz processor/radio platforms.

| MOTE PLATFORMS | MODEL NUMBER(S) | RF FREQUENCY BAND(S) |
|---|---|---|
| IRIS | XM2110 | 2400 MHZ TO 2483.5 MHZ |
| | M2110 | 2400 MHZ TO 2483.5 MHZ |
| MICAZ | MPR2400 | 2400 MHZ TO 2483.5 MHZ |
| | MPR2600 | 2400 MHZ TO 2483.5 MHZ |
| MICA2 | MPR400 | 868 MHZ TO 870 MHZ; 903 MHZ TO 928 MHZ |
| | MPR410 | 433.05 TO 434.8 MHZ |
| | MPR600 | 868 MHZ TO 870 MHZ; 903 MHZ TO 928 MHZ |
| MICA2DOT | MPR510 | 868 MHZ TO 870 MHZ; 903 MHZ TO 928 MHZ |
| | MPR520 | 433.05 TO 434.8 MHZ |

Table 3.2 Mote processor/radio (MPR) platforms supported by MoteView

| **Sensor and Data Acquisition Boards** | **Mote Platforms** | | | |
|---|---|---|---|---|
| | **IRIS** | **MICAZ** | **MICA2** | **MICA2DOT** |
| MTS101 | | ✓ | ✓ | |
| MTS300/310 | ✓ | ✓ | ✓ | |
| MTS410 | | ✓ | | |
| MTS400/MTS420 | ✓ | ✓ | ✓ | |
| MTS450 | | ✓ | ✓ | |
| MTS510 | | | | ✓ |
| MDA100 | ✓ | ✓ | ✓ | |
| XBW-DA100 | | ✓ | | |
| MDA300 | ✓ | ✓ | ✓ | |
| MDA320 | ✓ | ✓ | ✓ | |
| XBW-DA325 | | ✓ | | |
| MDA500 | | | | ✓ |

Table 3.3 Sensor (MTS series) and data acquisition boards supported by MoteView and their plug-and-play compatible Mote platforms

## 3.12 Application Quick Start

Once a sensor network is running and MoteView is installed on a PC, minimal configuration is necessary to start collecting data from the sensor network.

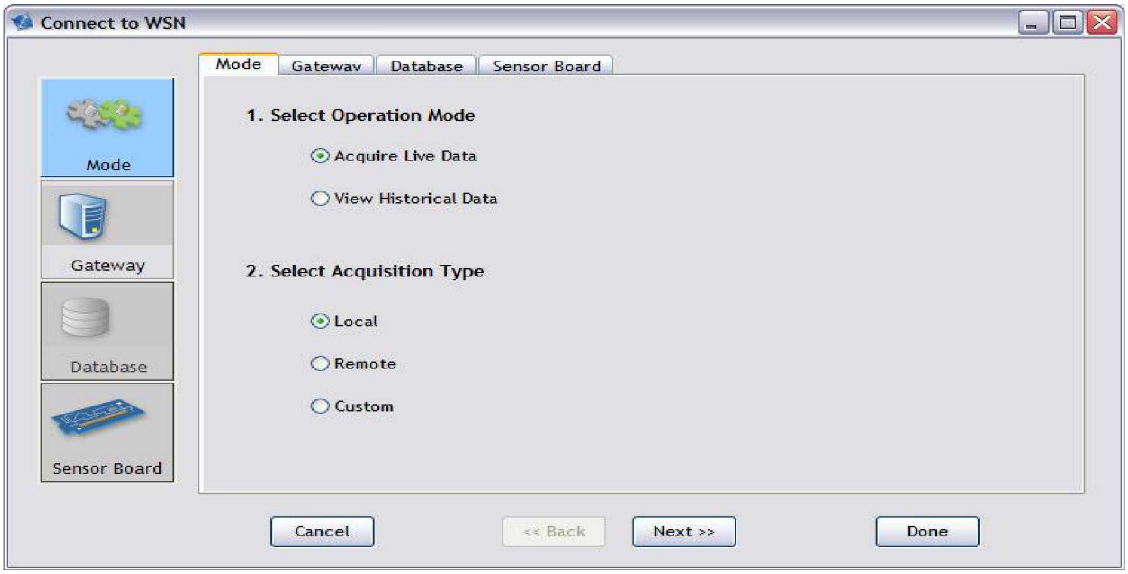
**Verify PostgreSQL Installation**

During the installation of MoteView a static database was included to make it possible to demonstrate MoteView’s features without having to be connected to an active sensor network or a remote server/database. The steps described here also apply to viewing data collected from an active sensor network.

### 3.12.1 Connecting to a Live Sensor Network on your local PC

Use the following steps to access data from a live sensor network connected to your local PC via the MIB510, MIB520 or MIB600 gateway.

1. Click on the Connect to WSN icon, the Connect to WSN Wizard will appear. Select the Mode tab, check on Acquire Live Data as operation mode and Local as acquisition type and click on Next >>.



1. Select Operation Mode

● Acquire Live Data

○ View Historical Data

2. Select Acquisition Type

● Local

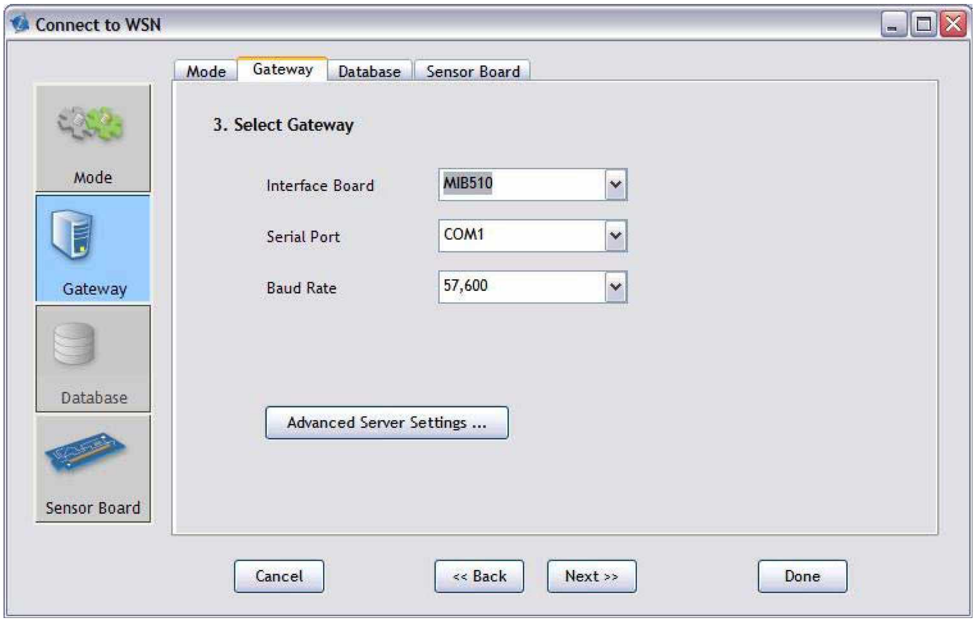
○ Remote

○ Custom

Figure3.8

2. In the Gateway tab, specify the Interface Board type, Port/Host Name etc as described below.

i. If using a MIB510, in the Gateway tab make sure that the MIB510's COM is set to the correct port number and that the baud rate is 57600.

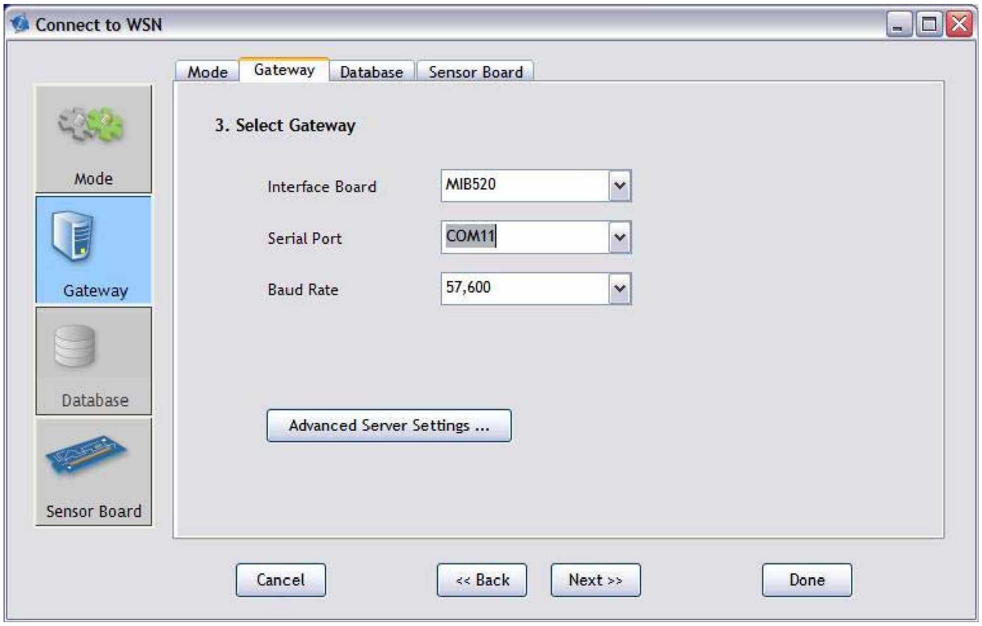


3. Select Gateway

Interface Board: MIB510
Serial Port: COM1
Baud Rate: 57,600

Figure 3.9

ii. If using MIB520, enter the higher of the 2 COM ports installed by the MIB520's driver and set the baud rate to 57600.



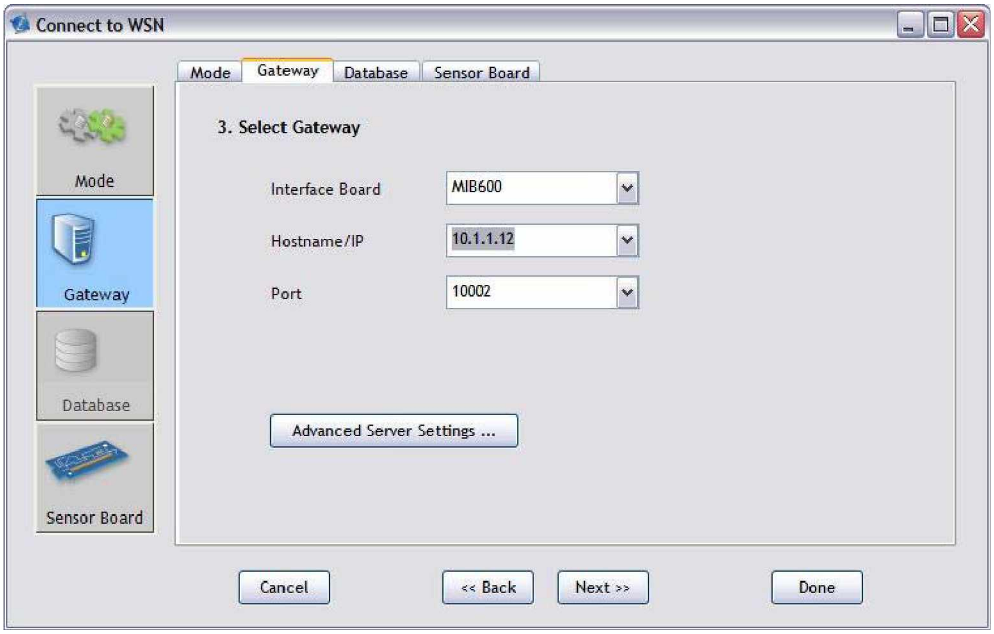
3. Select Gateway

Interface Board: MIB520
Serial Port: COM11
Baud Rate: 57,600

Figure 3.10

**NOTE**: The MIB520 requires the installation of the FTDI FT2232C drivers. Once these drivers are installed, the Device Manager (Start > Control Panel > System > Hardware) will display the MIB520 as two new virtual com ports.

iii. If using a MIB600, select MIB600 from Interface Board dropdown and enter the IP address of the MIB600 in the Hostname text-box. The Port should default to 10002.



3. Select Gateway

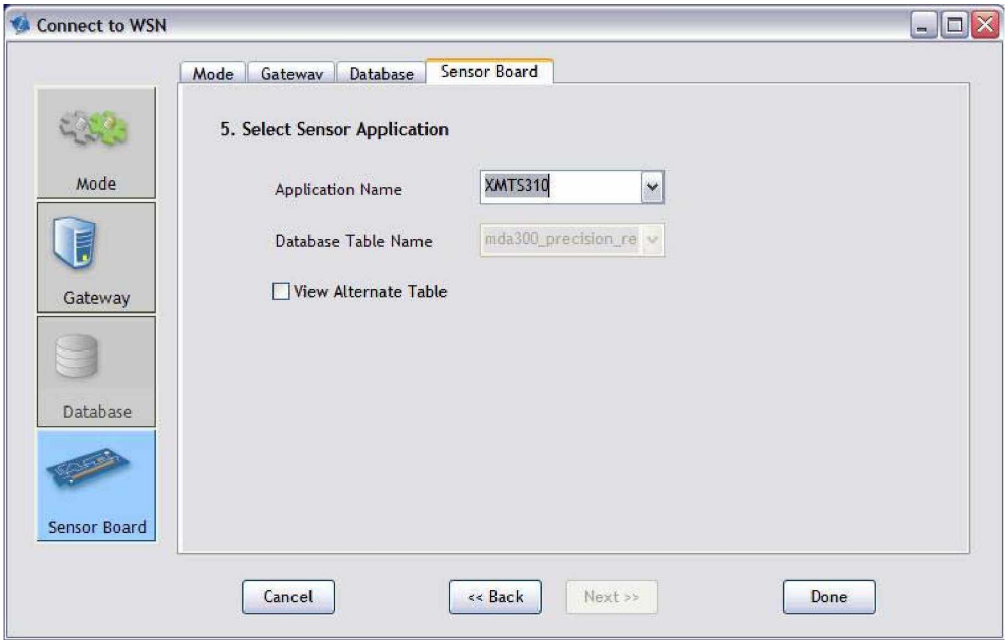
Interface Board: MIB600

Hostname/IP: 10.1.1.12

Port: 10002

Figure 3.11

3. In the Sensor Board tab, uncheck the ”View Alternate Table” checkbox and choose the XMesh Application Name that matches the firmware programmed into the Mote from Application Name dropdown. Click on Done.



5. Select Sensor Application

Application Name XMTS310

Database Table Name mda300_precision_re

☐ View Alternate Table

Figure 3.12

4. If you are not able to receive data, you may also need to check the "LIVE" check box on the main MoteView screen if it has not been previously checked. Use the Server Messages pane at the bottom of your MoteView display to verify that node data is being received by your PC.

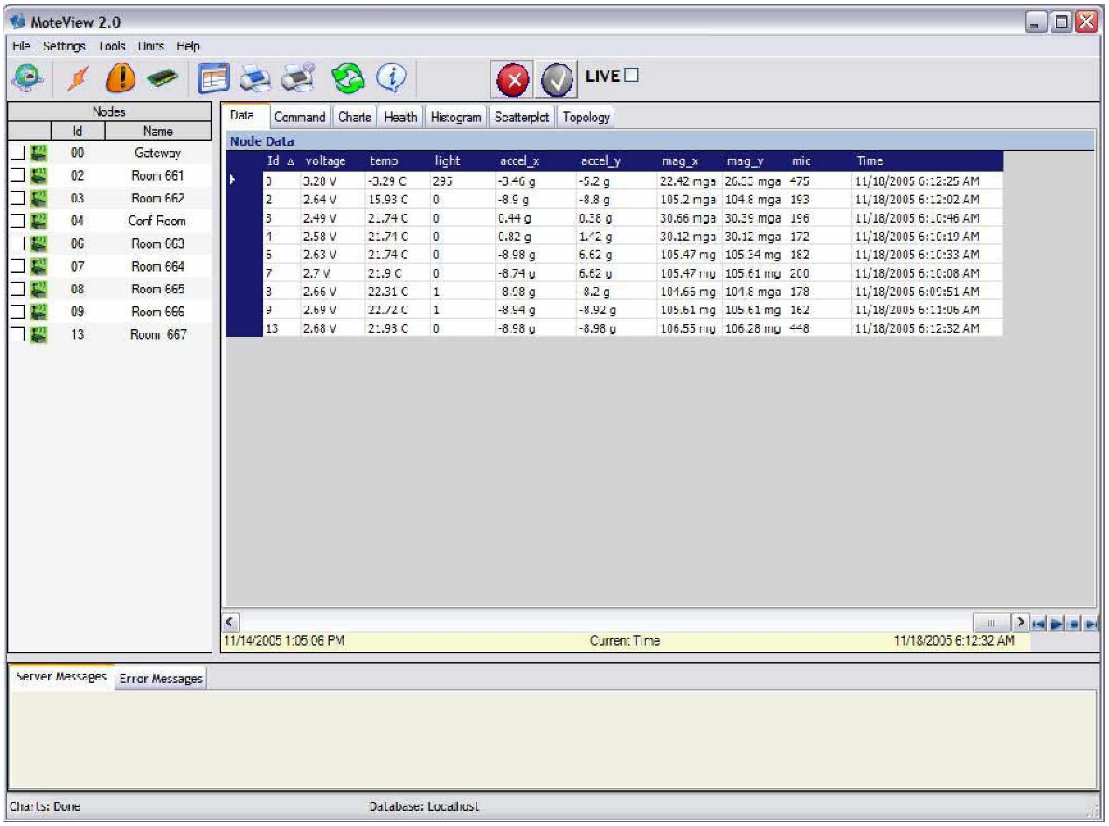
## 3.13 Visualization Tabs

Seven visualization tabs (Data, Command, Charts, Health, Histogram, Scatter plot and Topology) provide different methods of viewing your sensor data.

### 3.13.1 Data

The Data tab displays the latest sensor readings received for each node in the network. The columns include node ID, server timestamp and sensor values from the sensor board firmware packet. The sensor data is automatically converted into standard engineering units.

Left-clicking the column header allows you to sort by node ID, parent, temperature, voltage, last result time, or any other sensor reading. Right-clicking the column header displays a pop-up menu with unit conversions relevant to the sensor.



Nodes

| | Id | Name |
|---|---|---|
| ☐ | 00 | Gateway |
| ☐ | 02 | Room 661 |
| ☐ | 03 | Room 662 |
| ☐ | 04 | Conf Room |
| ☐ | 06 | Room 663 |
| ☐ | 07 | Room 664 |
| ☐ | 08 | Room 665 |
| ☐ | 09 | Room 666 |
| ☐ | 13 | Room 667 |

Node Data

| Id | voltage | temp | light | accel_x | accel_y | mag_x | mag_y | mic | Time |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 3.20 V | -0.29 C | 293 | -3.46 g | -5.2 g | 22.42 mga | 26.33 mga | 475 | 11/10/2005 6:12:25 AM |
| 2 | 2.64 V | 15.93 C | 0 | -8.9 g | -8.8 g | 105.2 mga | 104.8 mga | 193 | 11/18/2005 6:12:02 AM |
| 3 | 2.49 V | 21.74 C | 0 | 0.44 g | 0.38 g | 30.66 mga | 30.39 mga | 196 | 11/18/2005 6:10:46 AM |
| 4 | 2.58 V | 21.71 C | 0 | 0.82 g | 1.42 g | 30.12 mga | 30.12 mga | 172 | 11/18/2005 6:10:19 AM |
| 5 | 2.63 V | 21.74 C | 0 | -8.98 g | 6.62 g | 105.47 mg | 105.34 mg | 182 | 11/18/2005 6:10:33 AM |
| 7 | 2.7 V | 21.9 C | 0 | -8.74 g | 6.62 g | 105.47 mg | 105.61 mg | 200 | 11/18/2005 6:10:08 AM |
| 8 | 2.66 V | 22.31 C | 1 | 8.98 g | 8.2 g | 104.65 mg | 104.8 mga | 178 | 11/18/2005 6:09:51 AM |
| 9 | 2.69 V | 22.72 C | 1 | -8.94 g | -8.92 g | 105.61 mg | 105.61 mg | 162 | 11/18/2005 6:11:06 AM |
| 13 | 2.68 V | 21.93 C | 0 | -8.98 g | -8.98 g | 106.55 mg | 106.28 mg | 448 | 11/18/2005 6:12:32 AM |

11/14/2005 1:05:06 PM Current Time 11/18/2005 6:12:32 AM

Charts: Done Database: Localhost

Figure 3.13 Screenshot of a demo database displayed in the Data tab

## 3.14 Summary of the Process How to get live Data through Sensors

1. Install MoteView and Moteworks.
2. Install FTDI(Future Technology Devices International Limited) drivers for USB.
3. The programming of motes/nodes is done with the help of MoteConfig. We will get two communication ports for e.g COM5, COM6, then select smaller communication port for programming the motes.
4. Go in settings à Interface board MIB520 à COM5 à Apply.
5. The file to be uploaded should be gone through path Program files à Crossbow à MoteView à Xmesh à Micaz à MDA100 à hp.
6. We have to program the base station or gateway also.
7. We should keep same Group ID for all the network and Node ID should be “0” for base and non-zero value for the nodes.
8. Then click program after that program will be uploaded in the nodes, during programming switch off the nodes.

9. Then open MoteView. Connect to WSN.
    a. On the gateway, make sure to choose the higher com port number.
    b. In the sensor board, choose MTS310 as Application name then click on view alternate table. Then choose the xbw_da100_results as the database table name.
    c. Turn ON all the nodes. (Put batteries in each of the sensor node and turn the switch ON).
    d. Click on done. You should be able to view data being logged in from all the nodes.

# Chapter 4

## SENSOR KIT DETAILS (HARDWARE DESCRIPTION)

## 4.1 Introduction

The hardware features of the Mote Processor Radio (MPR) platforms and Mote Interface Boards (MIB) for network base stations and programming interfaces. It is intended for understanding and leveraging Crossbow's Smart Dust hardware design in real-world sensor network, smart RFID, and ubiquitous computing applications.

## 4.2 MPR2400 (MICAz)

### 4.2.1 Product Summary

The MICAz is the latest generation of Motes from Crossbow Technology. The MPR2400 (2400 MHz to 2483.5 MHz band) uses the Chipcon CC2420, IEEE 802.15.4 compliant, ZigBee ready radio frequency transceiver integrated with an Atmega128L micro-controller. The same MICA family, 51 pin I/O connector, and serial flash memory is used; all application software and sensor boards are compatible with the MPR2400.

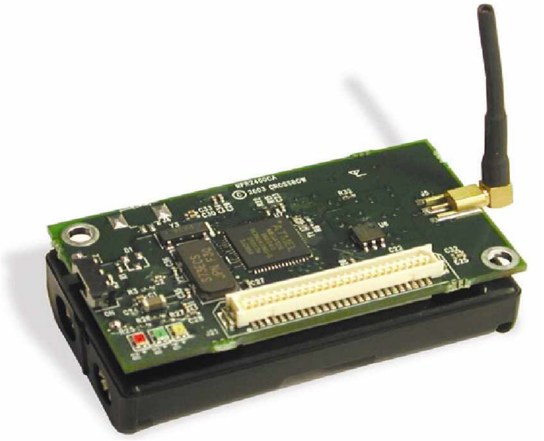


Figure 4.1 Photo of the MPR2400—MICAz with standard antenna

### 4.2.2 Block Diagram and Schematics for the MPR2400 / MICAz

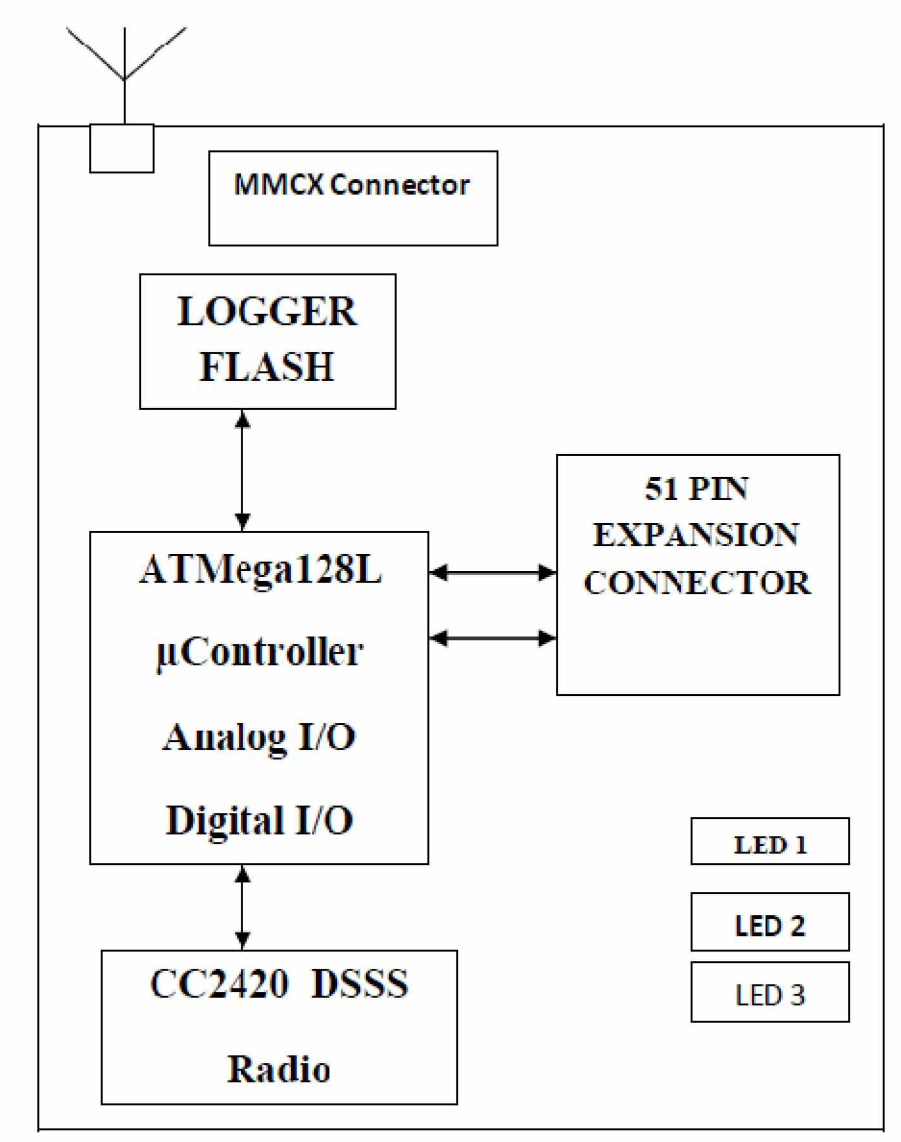


Figure 4.2

### 4.2.3 51-pin Expansion Connector

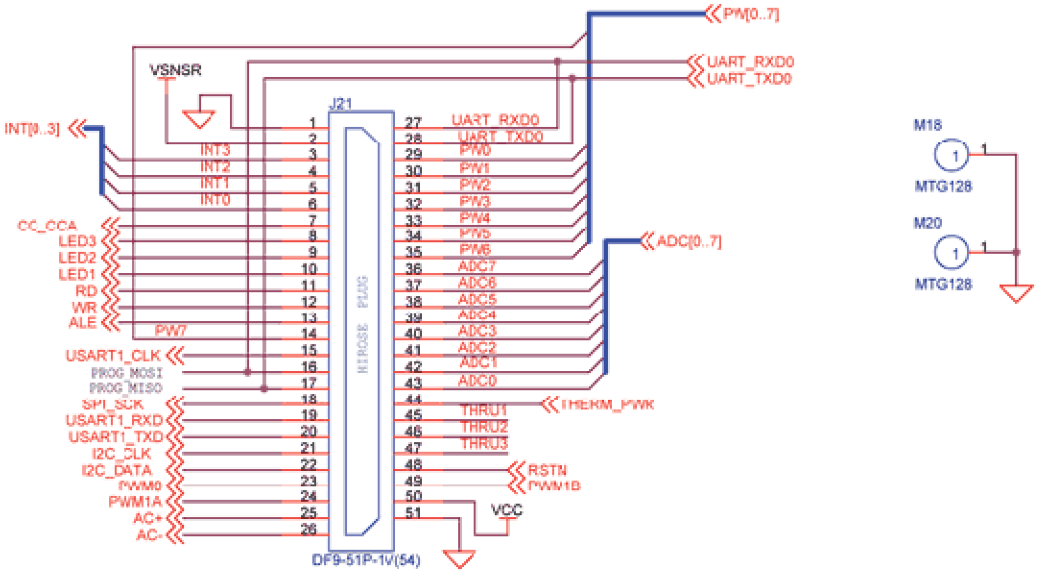


Figure 4.3(a)

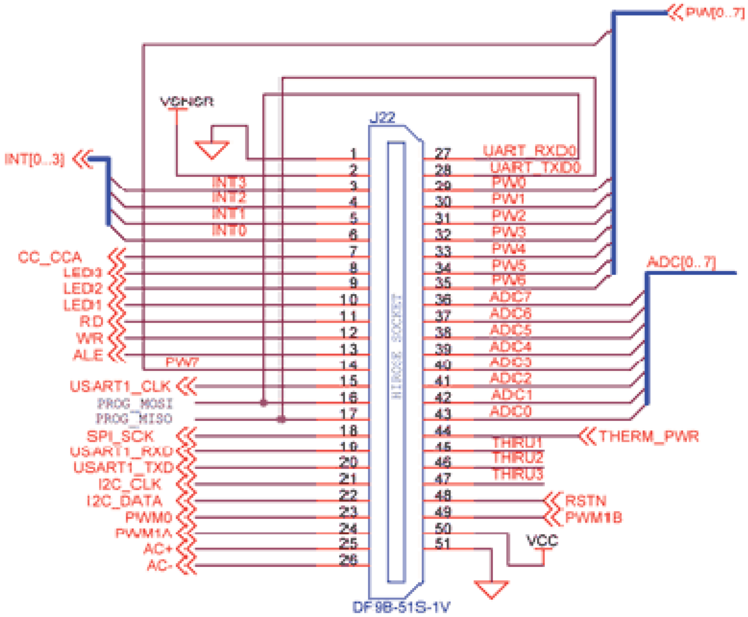


Figure 4.3(b)

## 4.2.4 CC2420 Radio

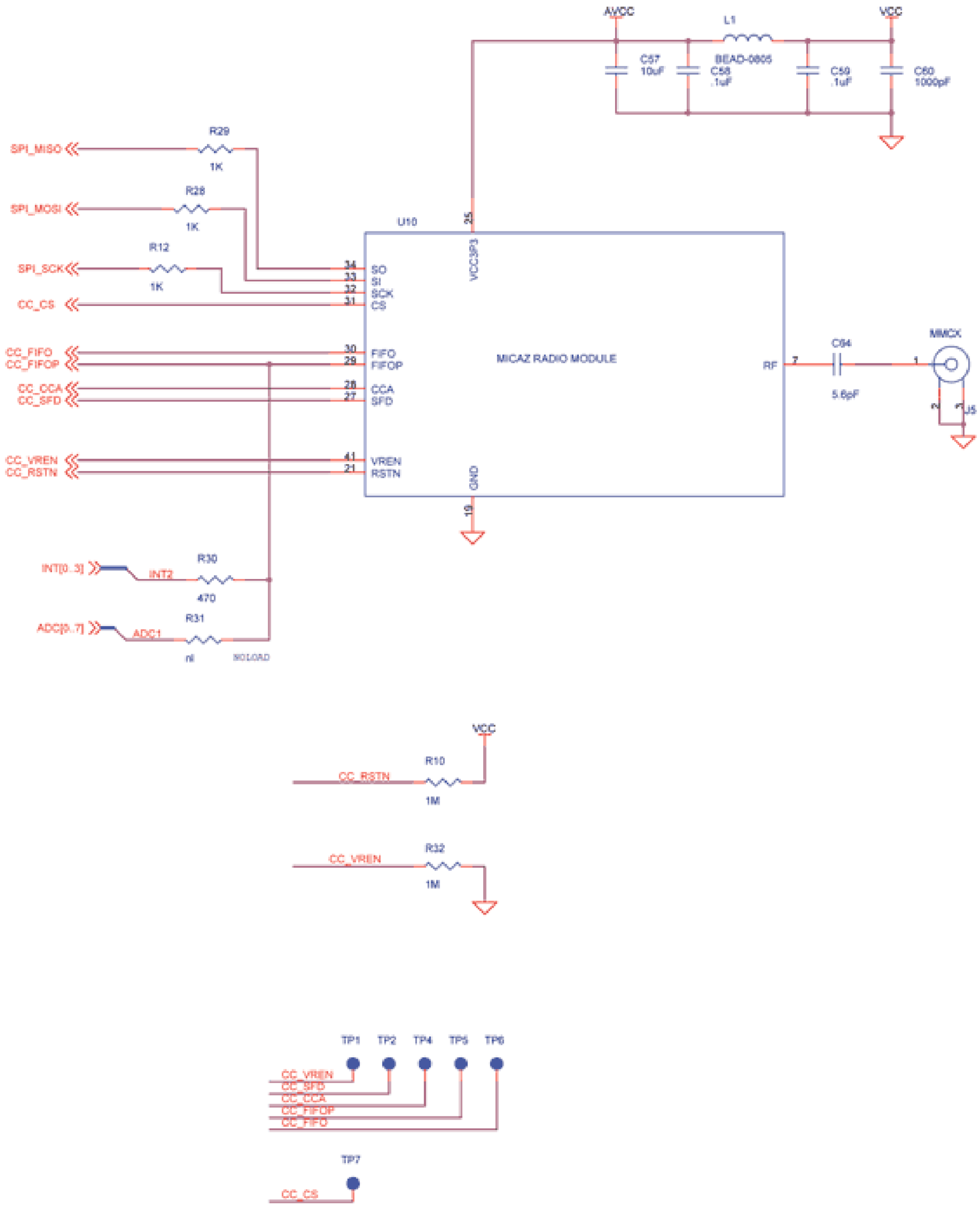


Figure 4.4

## 4.3 Data Acquisition Boards

The MTS series of sensor boards and MDA series of sensor/data acquisition boards are designed to interface with Crossbow's MICA, MICA2, and MICA2DOT family of wireless Motes. There are a variety of sensor boards available, and the sensor boards are specific to the MICA, MICA2 board or the MICA2DOT form factor. The sensor boards allow for a range of different sensing modalities as well as interface to external sensor via prototyping areas or screw terminals.

| Crossbow Part Name | Motes Supported | Sensors and Features |
|---|---|---|
| MTS101CA | MICAz, MICA2, MICA | Light, temperature, prototyping area |
| MTS300CA | IRIS, MICAz, MICA2, MICA | Light, temperature, microphone, and buzzer |
| MTS300CB | | |
| MTS510CA | MICA2DOT | Light, microphone, and 2-axis accelerometer |
| MDA100CA | IRIS, MICAz, MICA2 | Light, temperature, prototyping area |
| MDA100CB | | |
| MDA300CA | IRIS, MICAz, MICA2 | Light, relative humidity, general purpose interface for external sensors |
| MDA500CA | MICA2DOT | Prototyping area |

Table 4.1 Crossbow's Sensor and Data Acquisition Boards

## 4.4 MDA100CA/MDA100CB

MDA100CA and MDA100CB have the same content except for some minor changes. The MDA100 series sensor boards have a precision thermistor, a light sensor/photocell, and general prototyping area. The prototyping area supports connection to all eight channels of the Mote's analog to digital converter (ADC0–7), both USART serial ports and the I2C digital communications bus. The prototyping area also has 45 unconnected holes that are used for breadboard of circuitry.

### 4.4.1 Thermistor

The thermistor, sensor is a highly accurate and highly stable sensor element. With proper calibration, an accuracy of 0.2 °C can be achieved. The thermistor's resistance varies with temperature. The resistance vs. temperature graph is non-linear. The sensor is connected to the analog-digital converter channel number 1 (ADC1) thru a basic resistor divider circuit. The sensor is connected to the analog-digital converter channel number 1 (ADC1) through a basic resistor divider circuit.

| **Temperature (°C)** | **Resistance (Ohms)** |
|---|---|
| -40 | 239,800 |
| -20 | 78,910 |
| 0 | 29,940 |
| 25 | 10,000 |
| 40 | 5592 |
| 60 | 2760 |

Table 4.2 Resistance vs. Temperature

### 4.4.2 Conversion to Engineering Units

The Mote's ADC output can be converted to Kelvin using the following approximation over 0 to 50 °C:

$$1/T(K) = a + b \times \ln(R_{thr}) + c \times [\ln(R_{thr})]^3$$

where:

$R_{thr}$ = R1(ADC_FS-ADC)/ADC

a = 0.001010024

b = 0.000242127

c = 0.000000146

R1 = 10 kΩ

ADC_FS = 1023, and

ADC = output value from Mote's ADC measurement.

### 4.4.3 Light Sensor

The light sensor is a simple CdSe photocell. The maximum sensitivity of the photocell is at the light wavelength of 690 nm. Typical on resistance, while exposed to light, is 2 kΩ. In order to use the light sensor, digital control signal PW1 must be turned on. The output of the sensor is connected to the analog-digital converter channel 1 (ADC1). When there is light, the nominal circuit output is near VCC or full-scale, and when it is dark the nominal output is near GND or zero.

### 4.4.4 Prototyping Area

The prototyping area is a series of solder holes and connection points for connecting other sensors and devices to the Mote.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | GND | GND | GND | VCC | VCC | VCC |
| 2 | OPEN | OPEN | USART1_CK | INT3 | ADC2 | PW0 |
| 3 | OPEN | OPEN | UART0_RX | INT2$^+$ | ADC1$^+$ | PW1$^+$ |
| 4 | OPEN | OPEN | UART0_TX | INT1 | ADC0$^+$ | PW2 |
| 5 | OPEN | OPEN | SPI_SCK | INT0 | THERM_PWR | PW3 |
| 6 | OPEN | OPEN | USART1_RX | BAT_MON | THRU1 | PW4 |
| 7 | OPEN | OPEN | USART1_TX | LED3 | THRU2 | PW5 |
| 8 | OPEN | OPEN | I2C_CLK | LED2 | THRU3 | PW6 |
| 9 | OPEN | OPEN | I2C_DATA | LED1 | RSTN | ADC7 |
| 10 | OPEN | OPEN | PWM0 | RD | PWM1B | ADC6 |
| 11 | OPEN | OPEN | PWM1A | WR | OPEN | ADC5 |
| 12 | OPEN | OPEN | AC+ | ALE | OPEN | ADC4 |
| 13 | OPEN | OPEN | AC- | PW7 | OPEN | ADC3 |
| 14 | GND | GND | GND | VCC | VCC | VCC |
| 15 | OPEN | OPEN | OPEN | OPEN | OPEN | OPEN |
| 16 | OPEN | OPEN | OPEN | OPEN | OPEN | OPEN |
| 17 | OPEN | OPEN | OPEN | OPEN | OPEN | OPEN |

Table 4.3 Connection Table for MDA100. Use the photo (top view) below the table to locate the pins.

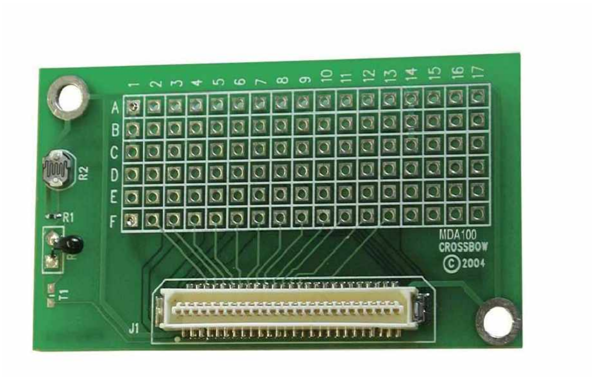


Figure 4.5 MDA100CB

**WARNING:** Never connect signals that are greater than VCC (3V typical) or less than 0 V to any of the holes that connect to the Mote Processor Radio board. It is okay to connect different voltages to the non-connected holes. However, be careful. If a voltage out of the range of 0 to Vcc should reach the Mote Processor Radio Board damage will occur.

## 4.5 MIB520 USB Interface Board

The MIB520 provides USB connectivity to the IRIS and MICA family of Motes for communication and in-system programming. It supplies power to the devices through USB bus. MIB520CB has a male connector while MIB520CA has female connector.

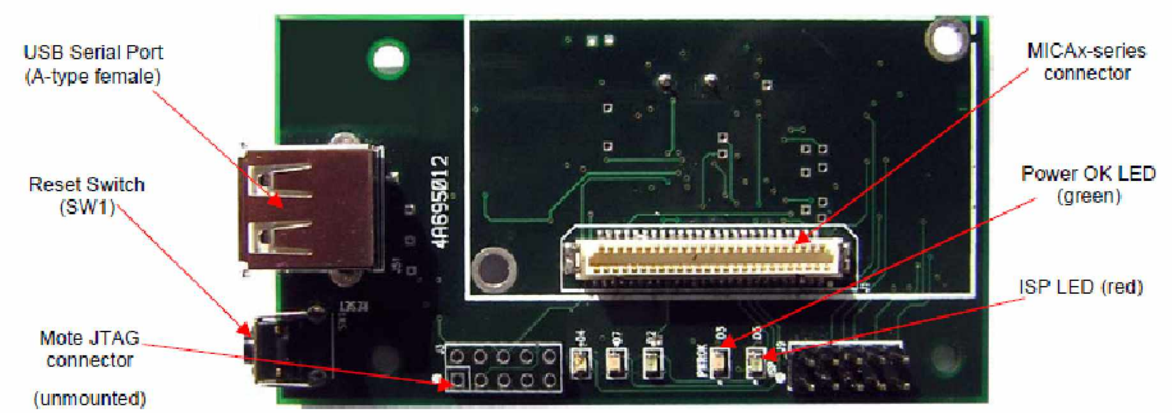


Figure 4.6 Photo of top view of an MIB520CA

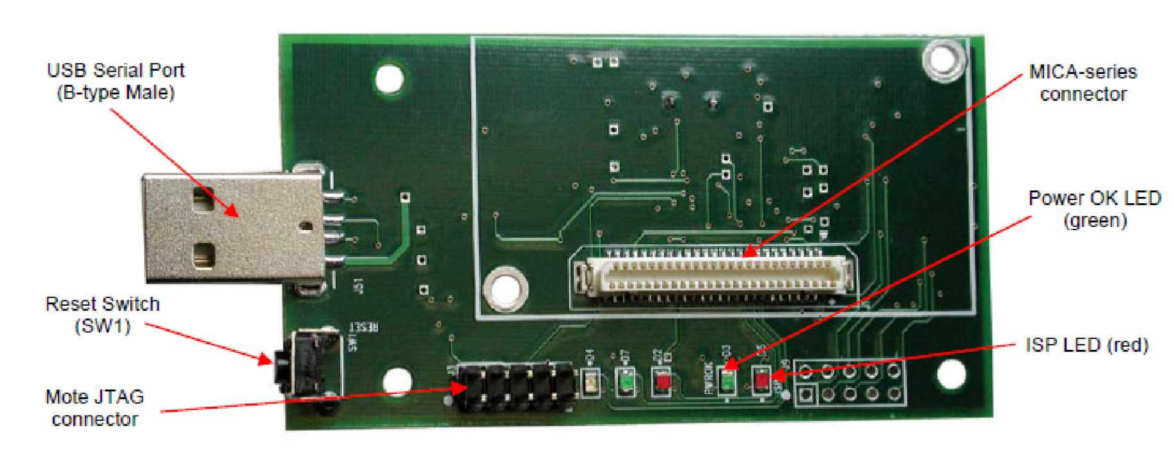


Figure 4.7 Photo of top view of an MIB520CB

### 4.5.1 ISP (In System Processor)

The MIB520 has an on-board in-system processor (ISP) an Atmega16L located at U14 to program the Motes. Code is downloaded to the ISP through the USB port. Next the ISP programs the code into the Mote.

### 4.5.2 Mote Programming Using the MIB520

Programming the Motes requires having MoteWorks/TinyOS installed in your host PC. The IRIS, MICAz and MICA2 Motes connect to the MIB520 for UISP programming from USB connected host PC.

## 4.6 MIB520 Use

### 4.6.1 Install FTDI USB Virtual COM Port Drivers

MIB520 uses FTDI FT2232C to use USB port as virtual COM port. Hence we need to install FT2232C VCP drivers.

- When you plug a MIB520 into your PC for the first time, the Windows detects and reports it as a new hardware. Please select “Install from a list or specific location

(Advanced)” and browse to “MIB520 Drivers” folder of the WSN Kit CDROM. Install shield wizard will guide you through the installation process.

- When the drivers are installed, you will see two serial ports added under the Control Panel à System à Hardware à Device Manager à Port. Make a note of the assigned COM port numbers.
- The two virtual serial ports for MIB520 are $Com_n$ and $Com_{(n+1)}$; $Com_n$ is for Mote programming and $Com_{(n+1)}$ is for Mote communication.

### 4.6.2 Reset

The “RESET” push button switch resets both the ISP and Mote processors. It also resets the monitoring software which runs on the host PC.

### 4.6.3 JTAG

The MIB520 has a connector, J3 which connects to an Atmel JTAG pod for in-circuit debugging. This connector will supply power to the JTAG pod; no external power supply is required for the pod.

**WARNING**: The MIB520 also has JTAG and ISP connectors for the ISP processor. These are for factory use only.

### 4.6.4 Power

The MIB520 is powered by the USB bus of the host.

**WARNING**: When programming an IRIS/MICAz/MICA2 with the MIB520, turn off the battery switch.

### 4.6.5 USB Interface

The MIB520 offers two separate ports: one dedicated to in-system Mote programming and a second for data communication over USB.

# Chapter 5

## FUNDAMENTALS OF CLUSTER BASED ROUTING

## 5.1 Introduction

Hierarchical cluster-based routing scheme is suitable for habitat and environmental monitoring applications. The routing scheme is based on the fact that the energy consumed to send a message to a distant node is far greater than the energy needed for a short range transmission. We extend the LEACH protocol by using a head-set instead of a cluster head. In other words, during each election, a head-set that consists of several nodes is selected. The members of a head-set are responsible for transmitting messages to the distant base station. At one time, only one member of the head-set is active and the remaining head-set members are in sleep mode. The task of transmission to the base station is uniformly distributed among all the head-set members.

First, we describe a few terms that are used in defining our protocol. A clusterhead is a sensor node that transmits an aggregated sensor data to the distant base station. Non-cluster heads are sensor nodes that transmit the collected data to their cluster head. Each cluster has a head-set that consists of several virtual cluster heads; however, only one head-set member is active at one time. Iteration consists of two stages: an election phase and a data transfer phase. In an election phase, the head-sets are chosen for the pre-determined number of clusters. In the data transfer phase, the members of head-set transmit aggregated data to the base station. Each data transfer phase consists of several epochs. Each member of a head-set becomes a cluster head once during an epoch. A round consists of several iterations. In one round, each sensor node becomes a member of head-set for one time.

## 5.2 States of a sensor node

The damaged or malfunctioning sensor states are not considered. Each sensor node joins the network as a candidate. At the start of each iteration, a fixed number of sensor nodes are chosen as cluster heads; these chosen cluster heads acquire the active state. By the end of election phase, a few nodes are selected as members of the head-sets; these nodes acquire associate state. At the end of an election phase, one member of a head-set is in active state and the remaining head-set members are in associate state.

In an epoch of a data transfer stage, the active sensor node transmits a frame to the base station and goes into the passive associate state. Moreover, the associate, which is the next in the schedule to transmit to the base station, acquires the active state. During an epoch, the head-set members are distributed as follows: one member is in active state, a few members are in associate state, and a few members are in passive associate state.

During the transmission of the last frame of an epoch, one member is active and the remaining members are passive associates; there is no member in an associate state. Then, at the start of the next epoch, all the head-set members become associate and one of them is chosen to acquire the active state. At the end of an iteration, all the head-set members acquire the non-candidate state. The members in non-candidate state are not eligible to become a member of an head-set. At the start of a new round, all non-candidate sensor nodes acquire candidate state; a new round starts when all the nodes acquire non-candidate state.

## 5.3 Election Phase

In the proposed model, the number of clusters, k, are pre-determined for the wireless sensor network. At the start, a set of cluster heads are chosen on random basis. These cluster heads send a short range advertisement broadcast message. The sensor nodes receive the advertisements and choose their cluster heads based on the signal strengths of the advertisement messages. Each sensor node sends an acknowledgment message to its cluster head. Moreover, for each iteration, the cluster heads choose a set of associates based on the signal analysis of the acknowledgments.

A head-set consists of a cluster head and the associates. The head-set, which is responsible to send messages to the base station, is chosen for one iteration of a round. In an epoch of an iteration, each member of the headset becomes a cluster head. All the head-set members share the same time slot to transmit their frames. Based on uniform rotation, a schedule is created for the head-set members for their frame transmissions; only the active cluster head transmits a frame to the base station. Moreover, a schedule is created for the data acquisition and data transfer time intervals for the sensor nodes that are not members of the head-set.

## 5.4 Data Transfer Phase

Once clusters, head-sets, and TDMA-based schedules are formed, data transmission begins. The non-cluster head nodes collect the sensor data and transmit the data to the cluster head, in their allotted timer slots. The cluster-head node must keep its radio turned on to receive the data from the nodes in the cluster. The associate members of the head-set remain in the sleep mode and do not receive any messages. After, some pre-determined time interval, the next associate becomes a cluster head and the current cluster head becomes a passive head-set member. At the end of an epoch, all the head-set members have become a cluster head for once. There can be several epochs in an iteration. At the end of an iteration, the head-set members become non-candidate members and a new head-set is chosen for the next iteration. Finally, at the end of a round, all the nodes have become non-candidate members. At this stage, a new round is started and all the nodes become candidate members.

## 5.5 Quantitative Analysis

In this section, we describe a radio communication model that is used in the quantitative analysis of our protocol. The energy dissipation, number of frames, time for message transfer, and the optimum number of clusters are analytically determined.

### 5.5.1 Radio Communication Model

We use a radio model, where for a shorter distance transmission, such as within clusters, the energy consumed by a transmit amplifier is proportional to $r^2$. However, for a longer distance transmission, such as from a cluster head to the base station, the energy consumed is proportional to $r^4$. Using the given radio model, the energy consumed to transmit an l-bit message for a longer distance, d, is given by:

$$E_T = lE_e + l\varepsilon_l d^4 \tag{5.1}$$

Similarly, the energy consumed to transmit an l-bit message for a shorter distance is given by:

$$E_T = lE_e + l\varepsilon_s d^2 \tag{5.2}$$

Moreover, the energy consumed to receive the l-bit message is given by:

$$E_R = lE_e + lE_{BF} \tag{5.3}$$

Equation 5.3 includes the cost of beam forming approach that reduces energy consumption. The constants used in the radio model are given in Table 5.1.

| Description | Symbol | Value |
|---|---|---|
| Energy consumed by the amplifier to transmit at a shorter distance | $\epsilon_s$ | 10 pJ/bit/m$^2$ |
| Energy consumed by the amplifier to transmit at a longer distance | $\epsilon_l$ | 0.0013 pJ/bit/m$^4$ |
| Energy consumed in the electronics circuit to transmit or receive the signal | $E_e$ | 50 nJ/bit |
| Energy consumed for beam forming | $E_{BF}$ | 5 nJ/bit |

Table 5.1 Sample parameter values of the radio communication model used in our quantitative analysis.

### 5.5.2 Election Phase

For a sensor network of n nodes, the optimal number of clusters is given as k. All nodes are assumed to be at the same energy level at the beginning. The amount of consumed energy is same for all the clusters. At the start of the election phase, the base station randomly selects a given number of cluster heads. First, the cluster heads broadcast messages to all the sensors in their neighborhood. Second, the sensors receive messages from one or more cluster heads and choose their cluster head using the received signal strength. Third, the sensors transmit their decision to their corresponding cluster heads. Fourth, the cluster heads receive messages from their sensor nodes and remember their corresponding nodes. For each cluster, the corresponding cluster head chooses a set of m associates, based on signal analysis.

For uniformly distributed clusters, each cluster contains n/k nodes. Using Equation 5.2 and Equation 5.3, the energy consumed by a cluster head is estimated as follows:

$$E_{CH-elec} = \left\{ lE_e + l\varepsilon_s d^2 \right\} + \left\{ \left(\frac{n}{k} - 1\right) l (E_e + E_{BF}) \right\} \tag{5.4}$$

The first part of Equation 5.4 represents the energy consumed to transmit the advertisement message; this energy consumption is based on a shorter distance energy dissipation model. The second part of Equation 5.4 represents the energy consumed to receive $(\frac{n}{k} - 1)$ messages from the sensor nodes of the same cluster.

Using Equation 5.2 and Equation 5.3, the energy consumed by non-cluster head sensor nodes is

estimated as follows:

$$E_{non-CH-elec} = \left\{klE_e + klE_{BF}\right\} + \left\{lE_e + l\varepsilon_s d^2\right\} \tag{5.5}$$

The first part of Equation 5.5 shows the energy consumed to receive messages from k cluster heads; it is assumed that a sensor node receives messages from all the cluster heads. The second part of Equation 5.5 shows the energy consumed to transmit the decision to the corresponding cluster head.

### 5.5.3 Data Transfer Phase

During data transfer phase, the nodes transmit messages to their cluster head and cluster heads transmit an aggregated messages to a distant base station. The energy consumed by a cluster head is as follows:

$$E_{CH/frame} = \left\{lE_e + l\varepsilon_l d^4\right\} + \left\{(\frac{n}{k} - m)l(E_e + E_{BF})\right\} \tag{5.6}$$

The first part of Equation 5.6 shows the energy consumed to transmit a message to the distant base station. The second part of Equation 5.6 shows the energy consumed to receive messages from the remaining $(\frac{n}{k} - m)$ nodes that are not part of the head-set.

The energy, $E_{\text{non-CH/frame}}$, consumed by a non-cluster head node to transmit the sensor data to the cluster head is given below:

$$E_{non-CH/frame} = lE_e + l\varepsilon_s d^2 \tag{5.7}$$

For circular clusters with a uniform distribution of sensor nodes and a network diameter of M, the average value of $d^2$ is given as: $E[d^2] = (\frac{M^2}{2\Pi k})$. Equation 5.7 can be simplified as follows:

$$E_{non-CH/frame} = lE_e + l\varepsilon_s \frac{M^2}{2\Pi k} \tag{5.8}$$

In first iteration, $N_f$ data frames are transmitted. The frames transmitted by each cluster are $N_f$ /k. The $N_f$ /k frames are uniformly divided among n/k nodes of the cluster. Each cluster head frame transmission needs $(\frac{n}{k} - m)$ non-cluster head frames. For simplification of equations, the fractions $f_1$ and $f_2$ are given as below:

$$f_1 = \left( \frac{1}{\frac{n}{k} - m + 1} \right) \frac{1}{k} \tag{5.9}$$

$$f_2 = \left( \frac{\frac{n}{k} - m}{\frac{n}{k} - m + 1} \right) \frac{1}{k} \tag{5.10}$$

The energy consumptions in a data transfer stage of each cluster are as follows:

$$E_{CH-data} = f_1 N_f E_{CH/frame} \tag{5.11}$$

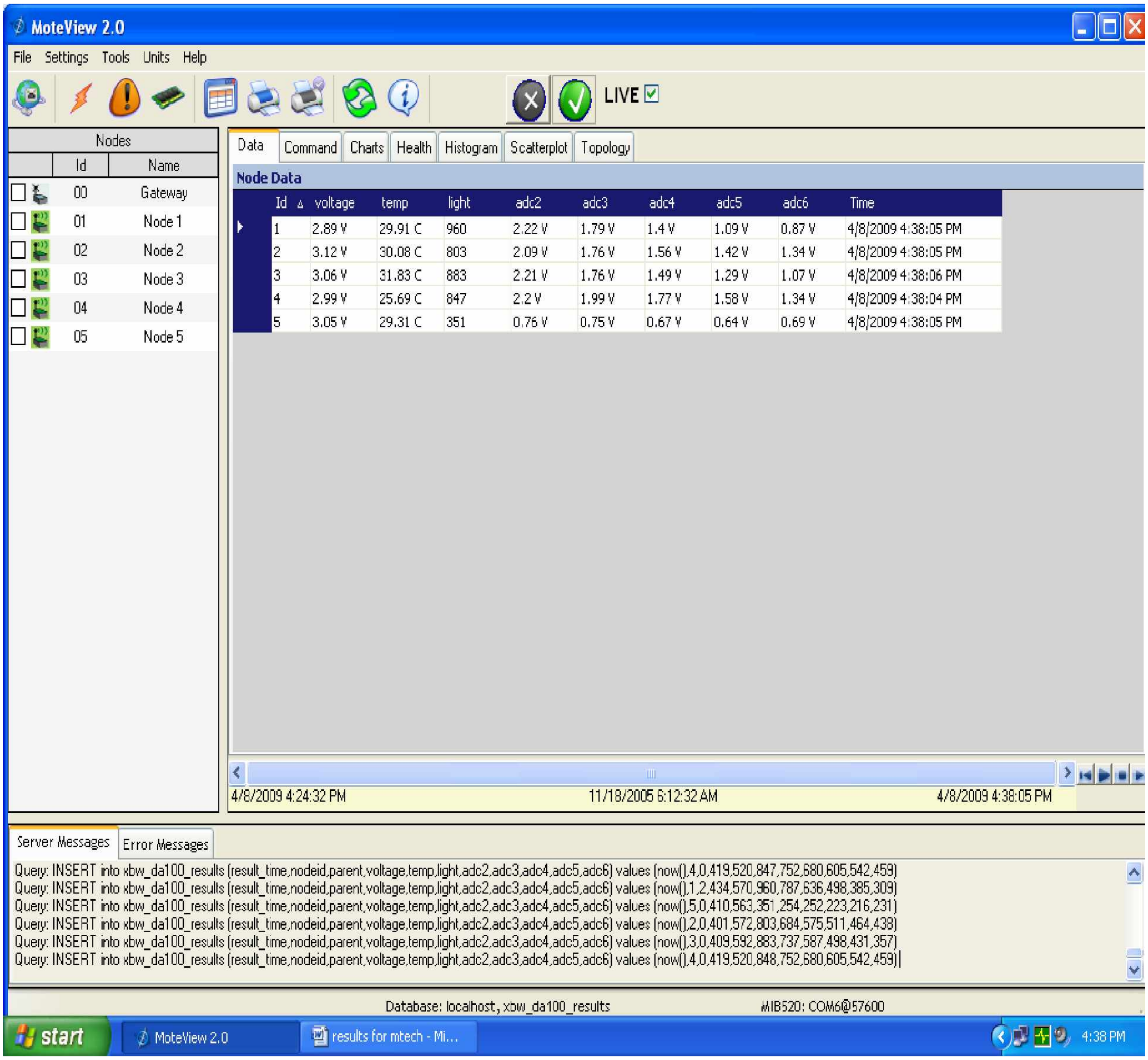
$$E_{non-CH-data} = f_2 N_f E_{non-CH/frame} \tag{5.12}$$

# Chapter 6

## RESULTS AND DISCUSSION

# RESULTS

1. Result showing the temperature and light measurements of the surrounding environment near the nodes. The results are obtained for five different nodes placed at different locations by using Crossbow sensor kit.



| Nodes | | |
|---|---|---|
| | Id | Name |
| ☐ | 00 | Gateway |
| ☐ | 01 | Node 1 |
| ☐ | 02 | Node 2 |
| ☐ | 03 | Node 3 |
| ☐ | 04 | Node 4 |
| ☐ | 05 | Node 5 |

Node Data

| Id | voltage | temp | light | adc2 | adc3 | adc4 | adc5 | adc6 | Time |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2.89 V | 29.91 C | 960 | 2.22 V | 1.79 V | 1.4 V | 1.09 V | 0.87 V | 4/8/2009 4:38:05 PM |
| 2 | 3.12 V | 30.08 C | 803 | 2.09 V | 1.76 V | 1.56 V | 1.42 V | 1.34 V | 4/8/2009 4:38:05 PM |
| 3 | 3.06 V | 31.83 C | 883 | 2.21 V | 1.76 V | 1.49 V | 1.29 V | 1.07 V | 4/8/2009 4:38:06 PM |
| 4 | 2.99 V | 25.69 C | 847 | 2.2 V | 1.99 V | 1.77 V | 1.58 V | 1.34 V | 4/8/2009 4:38:04 PM |
| 5 | 3.05 V | 29.31 C | 351 | 0.76 V | 0.75 V | 0.67 V | 0.64 V | 0.69 V | 4/8/2009 4:38:05 PM |

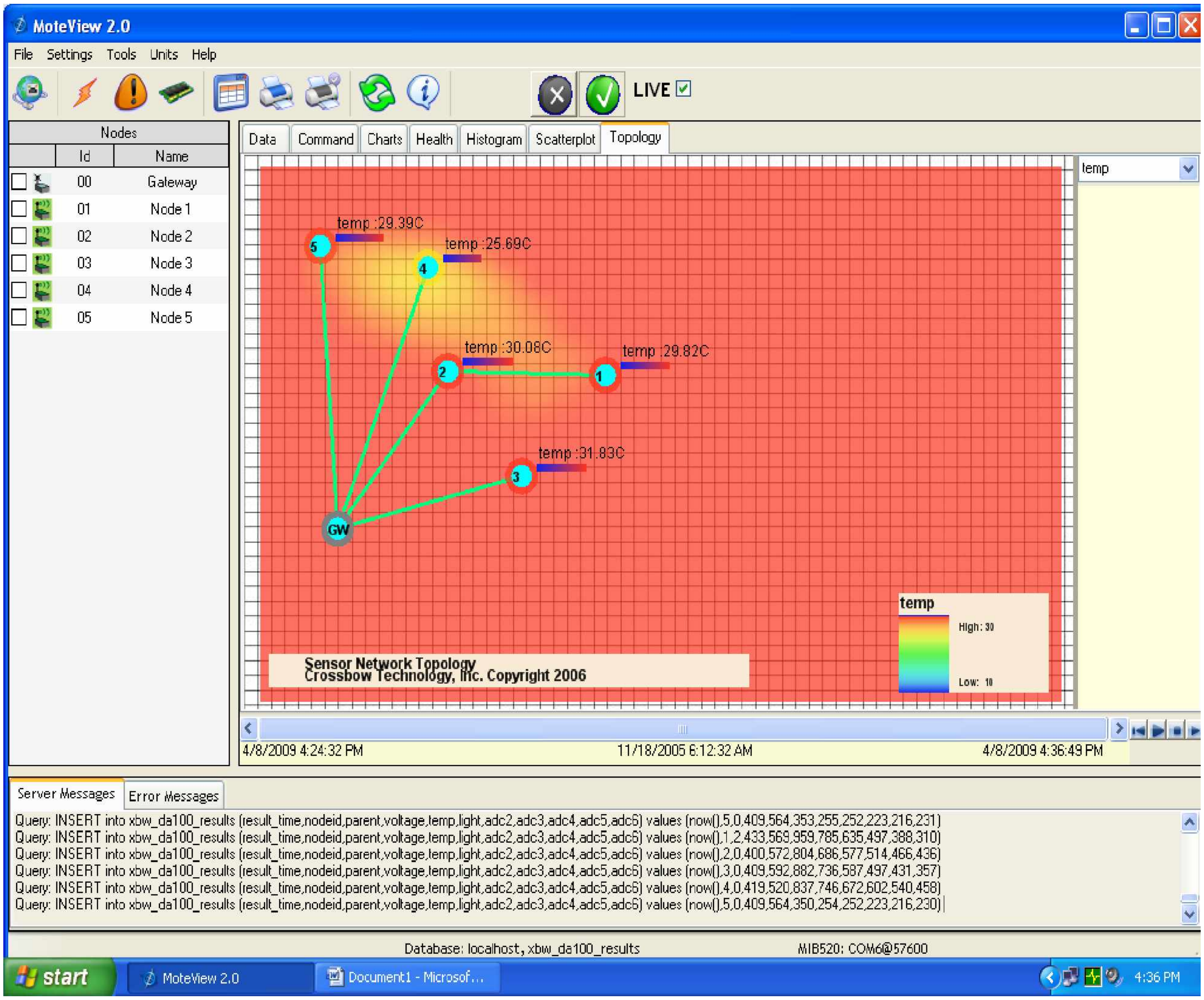
4/8/2009 4:24:32 PM 11/18/2005 6:12:32 AM 4/8/2009 4:38:05 PM

```
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),4,0,419,520,847,752,680,605,542,459)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),1,2,434,570,960,787,636,498,385,309)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),5,0,410,563,351,254,252,223,216,231)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),2,0,401,572,803,684,575,511,464,438)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),3,0,409,592,883,737,587,498,431,357)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),4,0,419,520,848,752,680,605,542,459)|
```

Database: localhost, xbw_da100_results MIB520: COM6@57600

Figure 6.1 Showing temperature and light measurement

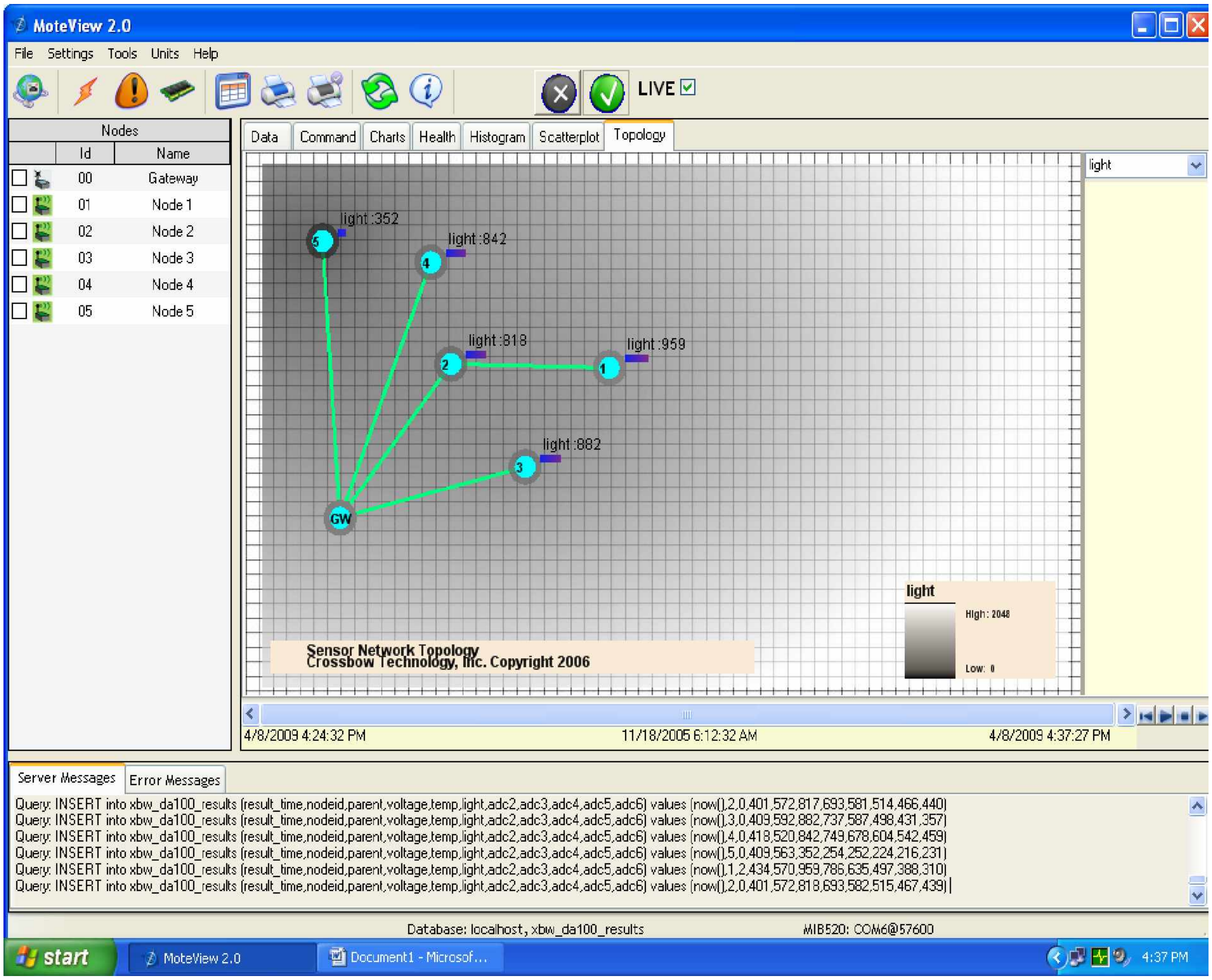
2. Result showing the graphical (topology) representation of temperature measurement of five different nodes at different locations.



```
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),5,0,409,564,353,255,252,223,216,231)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),1,2,433,569,959,785,635,497,388,310)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),2,0,400,572,804,686,577,514,466,436)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),3,0,409,592,882,736,587,497,431,357)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),4,0,419,520,837,746,672,602,540,458)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),5,0,409,564,350,254,252,223,216,230)
```

Figure 6.2 Showing temperature readings in topology form

3. Result showing the graphical (topology) representation of light measurement of five different nodes at different locations.

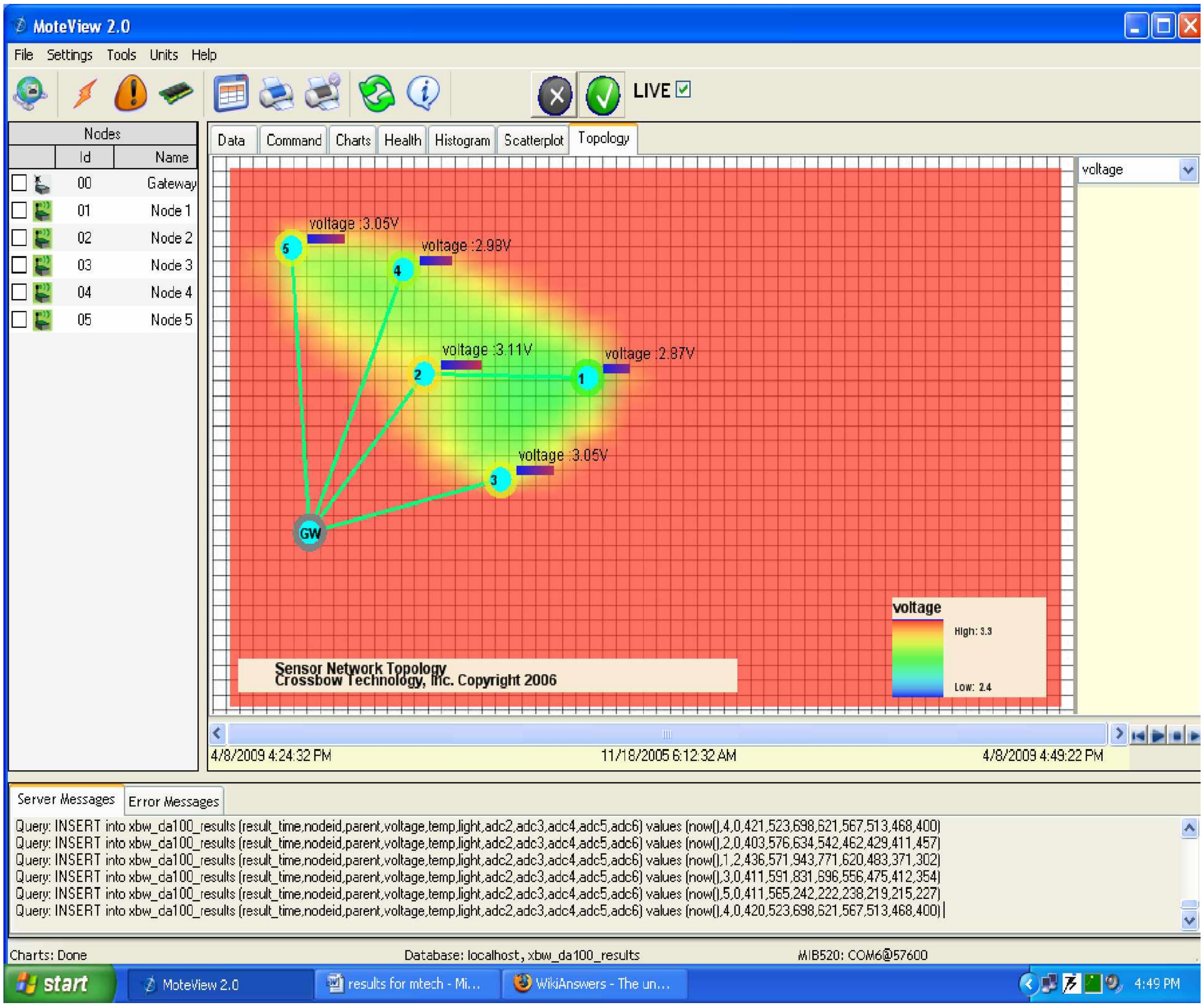


| Id | Name |
|---|---|
| 00 | Gateway |
| 01 | Node 1 |
| 02 | Node 2 |
| 03 | Node 3 |
| 04 | Node 4 |
| 05 | Node 5 |

```
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),2,0,401,572,817,693,581,514,466,440)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),3,0,409,592,882,737,587,498,431,357)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),4,0,418,520,842,749,678,604,542,459)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),5,0,409,563,352,254,252,224,216,231)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),1,2,434,570,959,786,635,497,388,310)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),2,0,401,572,818,693,582,515,467,439)
```

Figure 6.3 Showing light readings in topology form

4. Result showing the graphical (topology) representation of voltage measurement of five different nodes at different locations.



| | Nodes | |
|---|---|---|
| | Id | Name |
| ☐ | 00 | Gateway |
| ☐ | 01 | Node 1 |
| ☐ | 02 | Node 2 |
| ☐ | 03 | Node 3 |
| ☐ | 04 | Node 4 |
| ☐ | 05 | Node 5 |

```
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),4,0,421,523,698,621,567,513,468,400)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),2,0,403,576,634,542,462,429,411,457)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),1,2,436,571,943,771,620,483,371,302)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),3,0,411,591,831,696,556,475,412,354)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),5,0,411,565,242,222,238,219,215,227)
Query: INSERT into xbw_da100_results (result_time,nodeid,parent,voltage,temp,light,adc2,adc3,adc4,adc5,adc6) values (now(),4,0,420,523,698,621,567,513,468,400)
```

Charts: Done

Database: localhost, xbw_da100_results

MIB520: COM6@57600

Figure 6.4 Showing voltage readings in topology form

5. The graph that shows the variation in optimum number of clusters with respect to the head-set size, where the base station is at distance=150m and the number of nodes n=1000. The head-set size can be varied between 1 and 6. As the graph shows, the head-set size cannot be greater than 6. Moreover, for a given head-set size, the maximum number of clusters can also be determined from the graph.

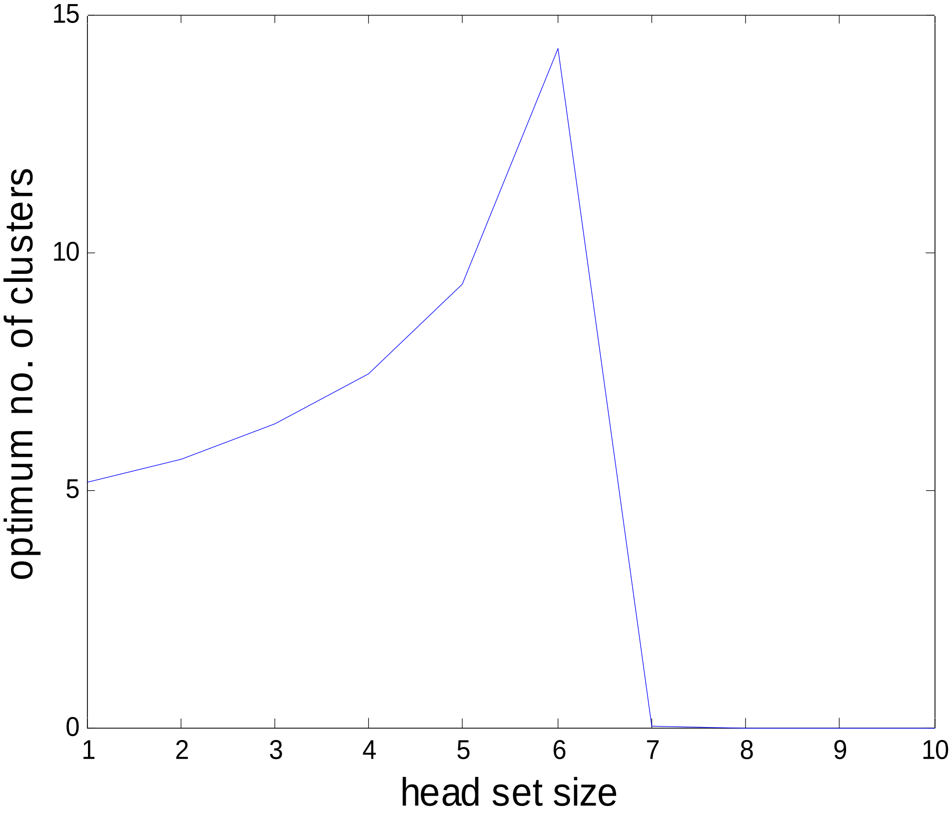


Figure 6.5 Optimum number of cluster Vs Head set size

6. Graph shows the variation in maximum cluster size with respect to distance from the base station and the head-set size. As the graph shows, bigger cluster sizes can be managed for larger values of head-set sizes. However, when the head-set size is small, only small number of clusters are possible. Moreover, when the distance from the base station is increased, more energy is spent for a distant transmission. As a result, for the same head-set size, the maximum number of clusters decreases when the distance to the base station increases.

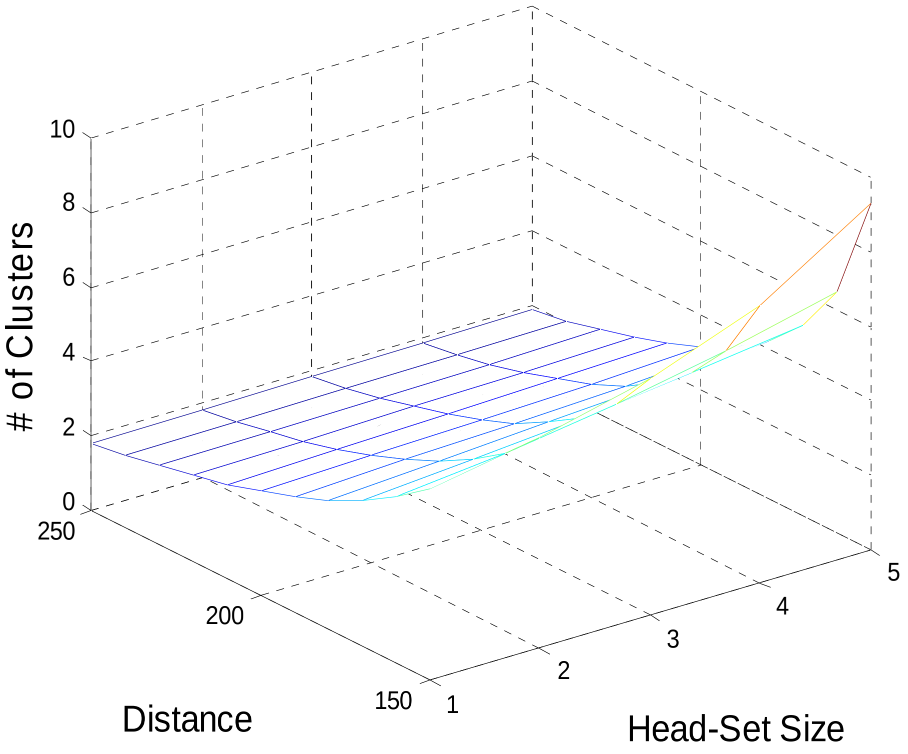


Figure 6.6 Variation in maximum cluster size with respect to distance from the base station and the head-set size.

7. Graph shows the energy consumption with respect to the number of clusters. As expected, the energy consumption is reduced when the number of clusters are increased. However, the rate of reduction in energy consumption is reduced for higher cluster sizes. Moreover, the energy consumption is lower when head-set size is 3 as compared to head-set size of 1.

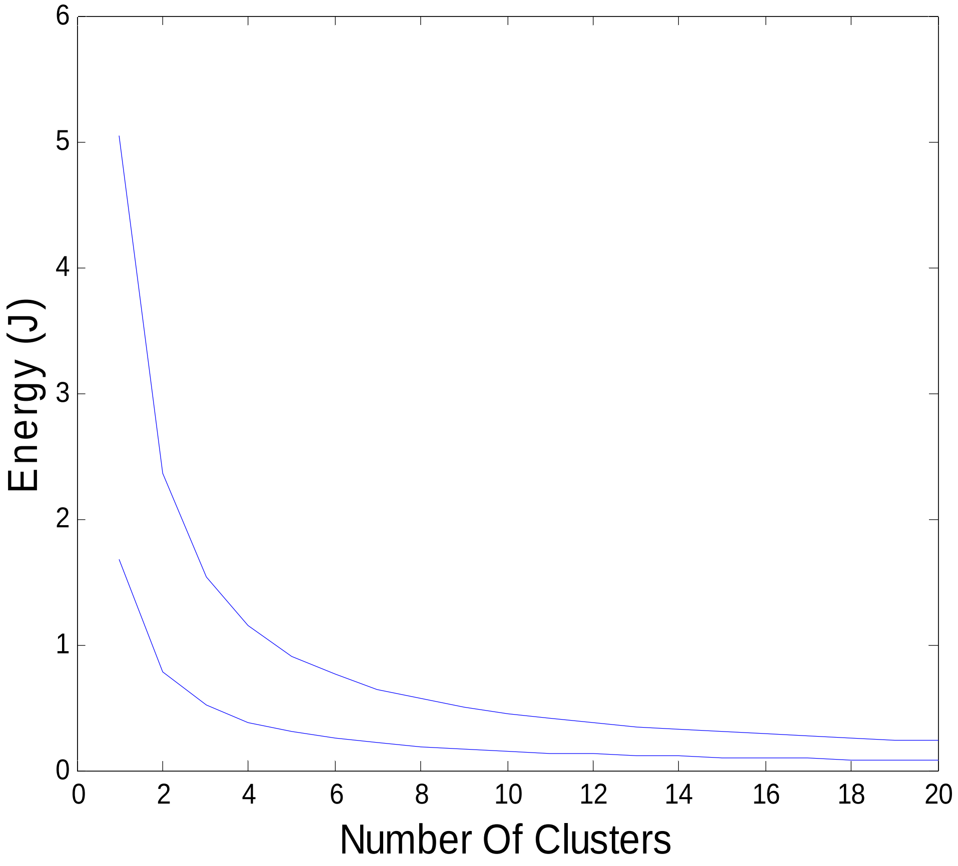


Figure 6.7 The energy consumption with respect to the number of clusters

8. Graph shows the variation in the energy consumed per node with respect to the number of clusters and network diameter. The x-axis and y-axis represent the number of clusters and the energy consumed in one round, respectively. In a round, the number of frames transmitted by one node is 20. The graphs show that energy consumption is reduced when the number of clusters are increased. For the simulated network of 1000 nodes, graphs shows that the optimum range of clusters lies between 20 and 60. When the number of clusters are below the optimum range, for example 10, the data collecting sensor nodes have to send data to the distant cluster heads. On the other hand, when the number of clusters are greater than optimum range, there will be more transmissions to the distant base station. Moreover, the energy consumption is lower for the higher head-set size. In the given graphs, the energy consumed is approximately three times less when headset size is 3 as compared to LEACH, where head-set size is 1.

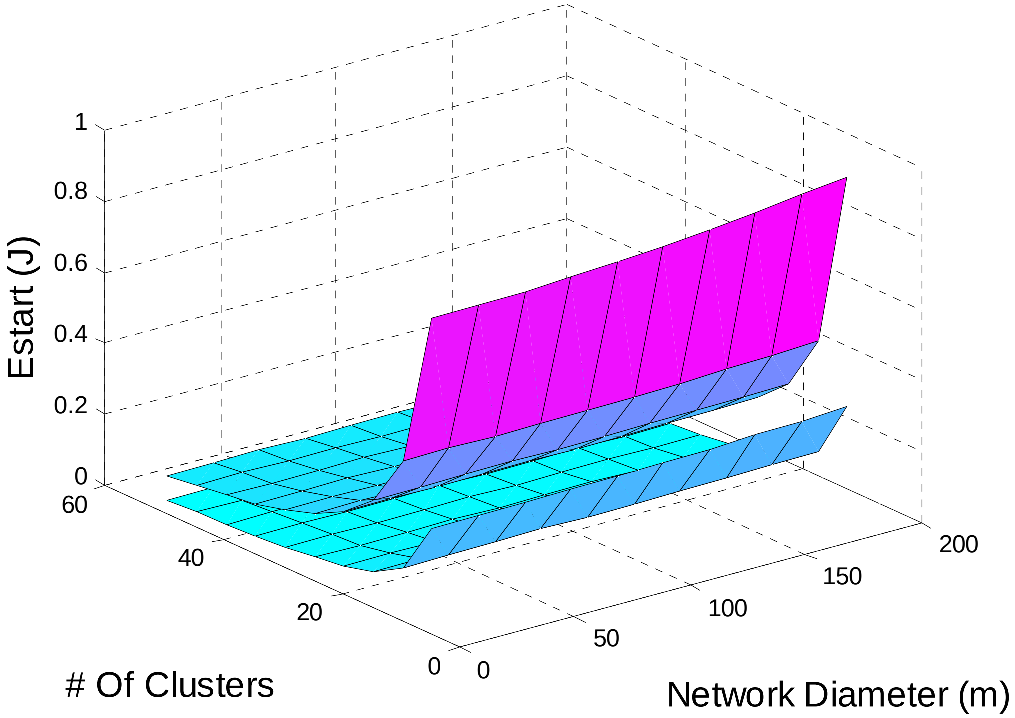


Figure 6.8 Energy consumed per round with respect to number of clusters

9. Graph shows the variation in the energy consumed per round with respect to head-set size and network diameter. The x-axis, y-axis, and z-axis represent the network diameter, the head-set size, and the energy consumed in one round, respectively. The number of data frames in one iteration is $N_f$ =10,000 and the number of clusters k = 50. As expected, the graph shows that energy consumption is reduced when the head-set size is increased. Moreover, this protocol provides a more systematic approach of reducing the energy consumption. If more nodes are added in LEACH, all the nodes are treated alike and these extra nodes will also be used in collecting the sensor data. However, in our approach, the number of sensor nodes for data collection remains unchanged and the number of control and management nodes can be adjusted.

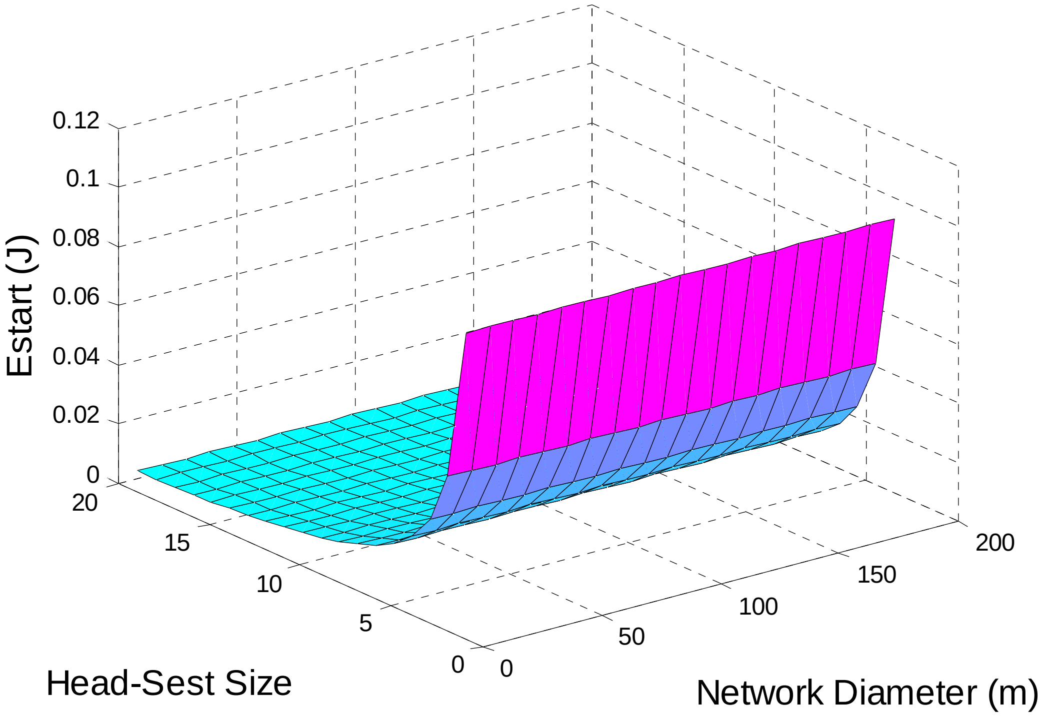


Figure 6.9 Energy consumed per round with respect to head-set size and network diameter.

10. In this section, average time to complete one iteration such that every node becomes a member of head-set is estimated. In other words, an average time for one iteration in each round is estimated. Moreover, frames transmitted in each iteration are also evaluated.

The graph shows the variation in time to complete one iteration with respect to cluster diameter and head-set size. The x-axis, y-axis, and z-axis represent the cluster diameter, head-set size, and time to complete one iteration, respectively. The head-set size is given as a percentage of cluster size. The start energy, Estart is fixed for all the cases. The start energy can be used for the longest period of time when the head-set size is 50% of the cluster size. When the headset size is less than 50% of the cluster size, there are fewer transmissions in each iteration but there are more iterations to complete the round. However, when the head-set size is greater than 50% of the cluster size, there are more transmissions in each iteration, although there are fewer iterations.

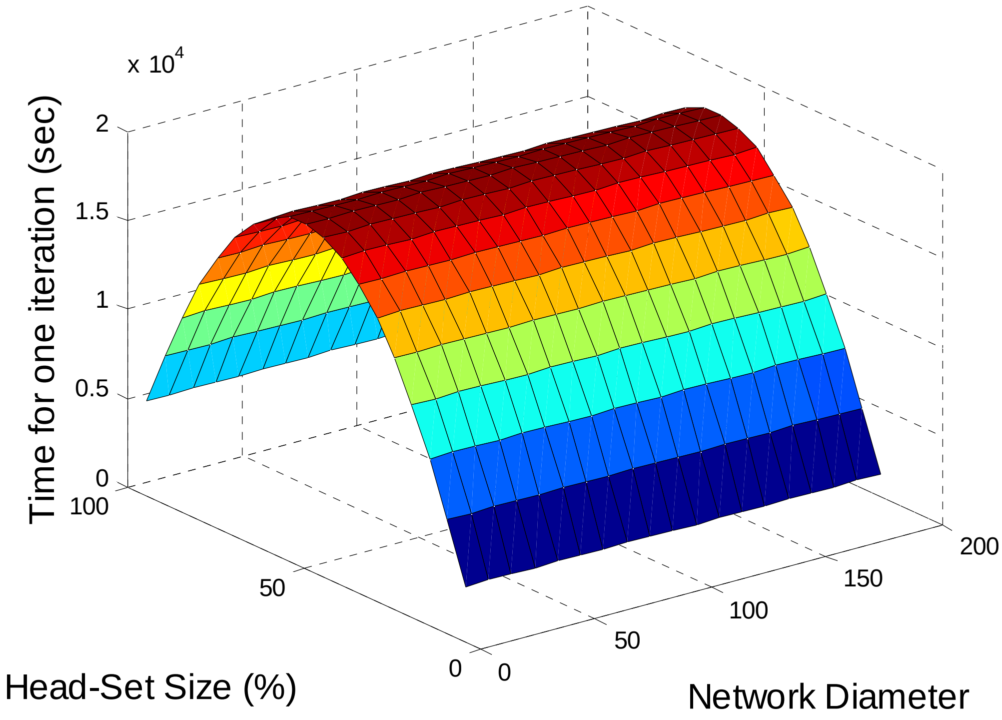


Figure 6.10 The time for iteration with respect to cluster diameter and the head-set size

# Chapter 7

# CONCLUSION AND FUTURE WORK

# CONCLUSION AND FUTURE WORK

Unlike other networks, WSNs are designed for specific applications. Applications include, but are not limited to, environmental monitoring, industrial machine monitoring, surveillance systems, and military target tracking. Each application differs in features and requirements. The measurement of temperature and light parameters through Crossbow Sensor Kit by using MoteView and MoteConfig environment has been done. The sensors or nodes are placed at different locations and the environmental parameters of that locations are measured. TinyOS is a very extensive and complex system. It has many applications and tools that need to be studied before one can fully understand the entire system. The results of our quantitative analysis of the proposed hierarchical cluster-based routing protocol indicate that the energy consumption can be systematically decreased by including more sensors in a head-set. For the same number of data collecting sensor nodes, the number of control and management nodes can be adjusted according to the network environment. In future work, the variation in the head-set size for different network conditions will be investigated. This work will be extended to incorporate non-uniform cluster distributions.

# REFERENCES:

[1] F.L. Lewis , “Wireless Sensor Networks paper”.

[2] S. Bandyopadhyay, E. J. Coyle, “An energy efficient hierarchical clustering algorithm for wireless sensor networks, IEEE Conference on Computer Communication (INFOCOM)”.

[3] www.xbow.com

[4] Mike Horton, John Suh, “A Vision for Wireless Sensor Networks”, IEEE 2005.

[5] D. Braginsky, D. Estrin, “ Rumor routing algorithm for sensor networks”, In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, New York, USA, 2002.

[6] F. Zhao, Leonidas Guibas, “Wireless Sensor Networks, An Information Processing Approach”, Morgan Kaufmann Publisher, 2004.

[7] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy efficient communication protocol for wireless microsensor networks”. In Proceedings of the Hawaii International Conference on System Sciences, January 2000.

[8] J. Hill, M. Horton, R. Kling, L. Krishnamurthy, “The Platforms Enabling Wireless Sensor Networks,” in review for CACM, January 2004.

[9] C.H. See, R.A. Abd-Alhameed, Y.F. Hu and K.V. Horoshenkov, “Wireless Sensor Transmission Range Measurement within the Ground level”.

[10] C.S Raghavendra, Krishna M. Sivalingam, Taieb Znati, “Wireless Sensor Networks”.

[11] Edgar H. Callaway, “Wireless Sensor Networks Architectures & Protocols”.

[12] Kazem Sohraby, Daniel Minoli, “Wireless Sensor Networks, Technology, Protocols and Applications”.

[13] Holger Karl, Andreas Willig, “Protocols and Architecture for Wireless Sensor Networks”.

[14] Behrouz A. Forouzan , “Data Communication and Networking”.