WIRELESS SENSOR NETWORK TESTBED:

MEASUREMENT AND ANALYSIS

by

RAHUL PRAMOD SAWANT

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2007

To

my grandfather Tukaram Vagh Sawant

and my maternal uncle Pratap Babaji Kadam

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Qilian Liang for encouraging and motivating me during my research study. I would also like to thank him for his advice in my thesis. I am indebted to Dr. Dan Popa for his guidance, invaluable advice and suggestions he has given me and the keen interest he has shown in my research work. I would like to thank Dr. R Stephen Gibbs for the constant guidance and advice he has given me and for taking time to be on the thesis committee.

I thank my mother Pallavi Pramod Sawant and my father Pramod Tukaram Sawant for believing in me and always encouraging me to reach greater heights. I would have never reached here without their constant support and backing.

I thank Dr. Frank Lewis and Dr. Dan Popa for allowing me to use their Distributed Intelligence & Autonomy Lab's (DIAL) computer, equipment and other resources in UT Arlington's Automation & Robotics Research Institute (ARRI).

Last but not the least I thank my colleagues Dr. Vincenzo Giordano, Prasanna Ballal, Muhammad Faizan Mysorewala and Sankar Gorthi for their much needed suggestions during the course of my thesis.

April 18, 2007

ABSTRACT

WIRELESS SENSOR NETWORK TESTBED:

MEASUREMENT AND ANALYSIS

Publication No. _____

Rahul Pramod Sawant, M.S.

The University of Texas at Arlington, 2007

Supervising Professor: Qilian Liang

Energy conservation is critical in Wireless Sensor Networks. Replacing or recharging batteries is not an option for sensors deployed in hostile environments. Generally communication electronics in the sensor utilizes most energy. This thesis studies the effect of changing the transmission power and baud rate on transmission distance. Using Shannon channel capacity formula and Log – Distance Path Loss Model, transmission distance is shown to be related to transmit power and baud rate. Extensive empirical readings are taken to confirm the above relation. The path loss exponent got as a result of data fitting is within the acceptable range for wireless environment.

Using the equation derived in this thesis, the distance between neighboring motes and traffic density it will be possible for sensors to adjust their transmit power and baud rate so as to use only the required amount of energy to maintain the wireless link to the neighbor and conserve power.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Wireless Sensor Networks (WSN) have been heralded as one of the most important technologies for the 21st century [1]. They are comprised of small, inexpensive sensors with wireless communication capabilities, called motes. These motes are deployed in large numbers and provide unprecedented opportunities for instrumenting and controlling homes, cities and the environment. Other area of application includes the Medical field where these motes can be used to monitor vital parameters of the human body. With the rapid advances in the technology to create micro electro-mechanical systems, the manufacturing of microscopic low power low cost sensors has become a reality. Together with CMOS technology these developments help to bring the vision of potentially dust size computing platforms into reality. These hugely capable motes are shrinking in size every year and are within the physical size of a typical coin. Future platforms will have the potential to fit within a cubic millimeter of volume [2].

These motes are deployed in ad – hoc networks and are powered by limited power supplies. They are sometimes deployed in difficult to reach regions and this makes it difficult to replace the batteries. Hence power conservation becomes an important factor for these motes. One of the main reasons for deploying these motes in

ad-hoc is power conservation. Power is consumed during data processing and RF communication, but communication electronics uses far more power than processing. This thesis is mainly concerned with changing the transmit power and baud rate in the communication protocol of the mote to conserve power during transmission and still have reliable links with neighbors within the transmission range.

1.2 Applications

Smart disposable motes can be deployed in the air, on ground, under water, on mobile platforms, on bodies, machines and inside buildings. A system of networked sensors can detect and track threats (e.g. vehicles, airplanes, people, chemical and biological agents) and can be used for weapons targeting and area denial. Each of the motes have embedded processing capability and have multiple sensors capable of detecting light, temperature, acceleration, magnetic field, chemical and biological agents. They also have onboard storage capability and local positioning knowledge using localization algorithms.

Potential applications of sensor networks include military sensing, physical security, air traffic control, traffic monitoring, video surveillance, industrial automation, robotics, environment monitoring, structure monitoring, patient monitoring and agriculture [1]. The following figure shows some of the applications of the wireless sensors.

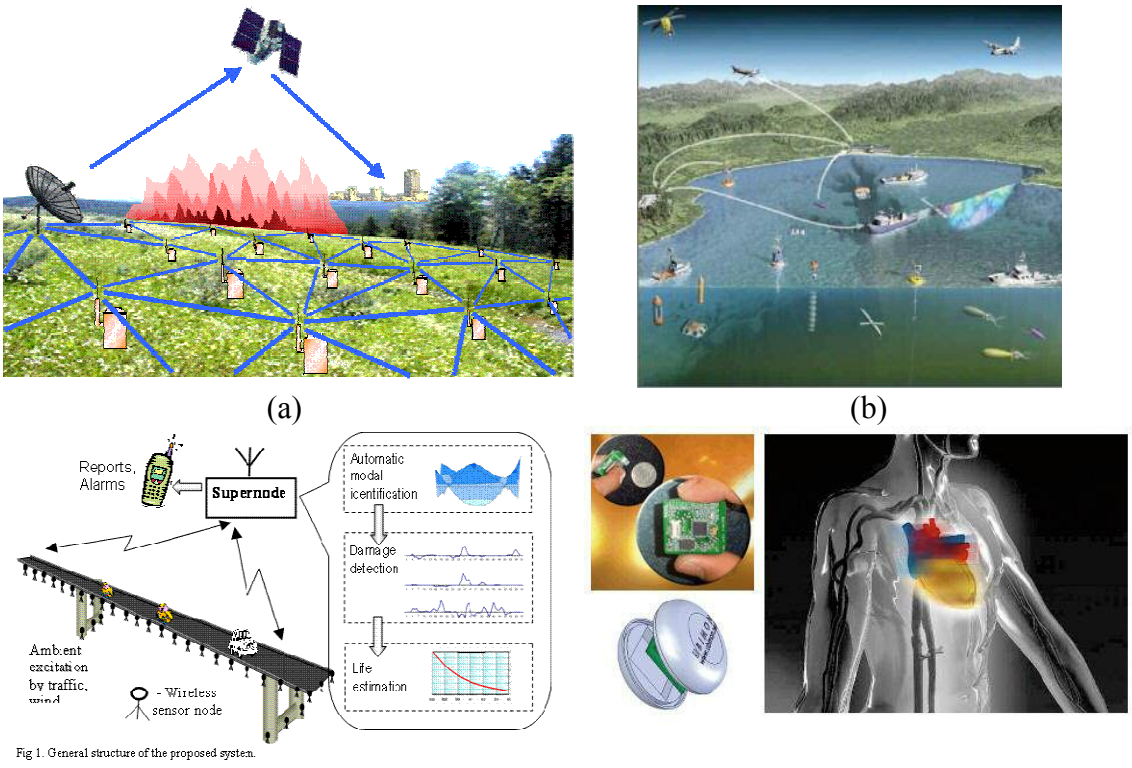

Figure 1.1: Applications of Wireless Sensors in (a) Agriculture, (b) Battlefield, (c) Structure Monitoring and (d) Medical Field [3, 4, 5, 6]

In most of these applications, motes are so embedded in the physical environment that energy conservation is the most important factor as it is difficult to manually replace or recharge the batteries as in the case of motes dropped from an airplane on the battlefield, those attached to the inner parts of the machines or those embedded inside the human organs. Since motes use far less energy for computation than for communication it becomes utmost important to adjust the power level and baud rate during transmission.

1.3 Contribution and Scope of Thesis

Sensor Networks being used in a range of fields from military to agriculture, and deployed in air, on land and under water, each situation presents different challenges to the designer. But one common challenge posed is the energy constraint. These motes expend energy to accomplish their basic task namely sensing, computation and communication. They are deployed in hostile environments and difficult to reach regions hence replacing or recharging these motes is not possible. Even for motes deployed in friendly environment recharging or replacing the batteries is cumbersome as normally motes will be densely populated. Hence energy saving becomes one of the important design criteria. It is seen that communication electronics require far more energy as compared to others and hence a lot of effort goes into designing energy efficient routing algorithms [7, 8, 9], directed diffusion algorithms [10, 11], clustering algorithms [12, 13, 14], data aggregation [15] and MAC protocols [16, 17, 18]. All these algorithms assume fixed transmit power and baud rate.

This thesis is the study of the effect of changing the transmit power and baud rate on transmission distance. It gives an equation to relate distance, transmit power and baud rate for Crossbow's Mica2 motes deployed in indoor environment. This can be incorporated along with the above algorithms and with knowledge of the distance to neighboring mote and the traffic density can be used to adjust the transmit power and baud rate so that packets can be forwarded to the neighboring mote with the least energy and maximum reliability. Adjusting the transmit power the neighboring mote will fall within the transmission range and neither will be out of range nor will be the

transmission range much bigger than the desired distance. This will also help to keep packet exchange to the desired mote and not to any distant mote, which can itself start some other exchange and so will help keep interference to a minimum. This study along with the energy efficient algorithms will further increase energy efficiency in real time when deployed on Mica2 motes.

1.4 Thesis Overview

This thesis is organized as follows: Chapter 2 gives the details of the software and hardware used on the motes and the software used on the computer attached to the base station for receiving and storing the data.

Chapter 3 discusses the transceiver chip TI Chipcon CC1000 used by the motes.

In Chapter 4 the experiment conducted to get the readings of transmission distance at different power level and baud rate is explained.

Chapter 5 details the curve fitting and surface fitting done on the data gathered from the experiment.

Chapter 6 gives a brief summary of the work presented in this thesis.

CHAPTER 2

WIRELESS SENSOR NETWORKS

2.1 Introduction

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations. Each node in the network is equipped with one or more sensors, a transceiver for wireless communication, a small microcontroller and a power source usually a battery. Since each node has a sensor and a transceiver it is called a mote in the field of technology. The size and cost constraints on the motes result in corresponding constraints on resources like power, memory, computational speed, bandwidth [19]. Sensor networks consist of one or more base stations which might have extra computational capability, no power constraint and robust communication ability. It acts as an interface between the sensor nodes and the end user.

Due to the tiny size of the mote, all the components are designed to perform the basic and bare minimum functionality required. Hence low cost low power microcontrollers and transceivers are used on the motes instead of state of the art expensive ones. Each and every component hardware as well as software used on the mote takes extra care that it uses the least amount of energy to carry out its task. Another constraint due to cost, mentioned above is memory. Each mote requires

memory to store the sensed data, to execute the communication algorithms and for the Operating System (OS). This calls for a special purpose OS which use the least memory and is specifically designed for sensor networks. One such OS used is TinyOS and is discussed in the next section. It is developed by University of California, Berkeley (UCB). TinyOS is written in nesC also discussed further in the chapter and is a dialect of the C Programming Language.

Because of the wide ranging applications of sensor networks a lot of research is going on globally in different area of sensor networks. One of the early researchers was the Network Embedded Systems Technology (NEST) research group of (UCB) in co – operation with Intel Research. They designed a range of hardware platforms starting with WeC, Rene, Dot, Mica, Mica2Dot, Mica2, Telos etc [20]. For all the experiments carried out in this thesis Mica2 platform is used and will be discussed in detail later.

The base station mote is equipped with a component, usually a serial interface or a USB interface to connect to the computer to which the sensed or processed data can be transferred and the user commands taken from. Almost all the software available can be used with more or less effort on the computer to collect and store the data received from the motes. General software like C and Java require lot of coding to be done to design user interfaces to read input from users. But one particular software which is much easier to use, learn and less time consuming to develops applications is LabVIEW and is used in the experiments carried out here. One major advantage of LabVIEW is that it is a Graphical Language. Again we will discuss this later.

2.2 TinyOS

TinyOS is a tiny micro – threaded operating system. It is an open source component based operating system especially designed for wireless sensor networks. It is designed to operate in severe memory and power constraints inherent in sensor networks [21]. Sensor networks due to their numerable limitations require a special operating system like this which can do very high concurrency operations and can work with various hardware platforms. Since the motes are designed to be as small as possible they always carry the most relevant hardware components and sensors. Because of this TinyOS provides a high degree of modularity with great efficiency. TinyOS is also capable of handling a lot of operations simultaneously since in sensor networks the motes will be sensing the events, sending packets to the base station, receiving and forwarding packets of other motes and also processing the data all at the same time. Another requirement of sensor networks is the software should be robust and reliable. The motes are densely populated and will be most of the time unattended. So to increase reliability one possible solution is to use redundancy but due to the space and power limitations we cannot do so. Another solution would be to have redundancy across motes but the huge communications cost are a deterrent. TinyOS is designed to overcome these problems. It also handles the less memory problem very well and only takes 178 bytes of memory [22]. TinyOS has an event based model and handles all the tasks associated with an event rapidly without any blocking and when there are no events happening conveniently operates in the sleep mode and conserves energy. It also has an efficient multithreading engine that processes the small amount of data

associated with the hardware events faster while the longer tasks are interrupted. The configuration of the operating system consists of a tiny scheduler and a graph of components. A component has four parts a command handler, an event handler, a fixed – sized frame and a few small tasks. Tasks, commands and handlers execute within the frame. Each component declares the commands it uses and the events it will signal. This makes it easier to have a layered architecture where the upper layer applications can give commands to lower level providers who in turn can signal the completion of the job or the physical occurrence of some event by signaling to the upper layer. Commands and events normally do small amount of work and the bulk of the work is left for the tasks to do. Tasks are generally used to handle low priority work and always run to completion. They are not interrupted by other tasks but can be interrupted by hardware interrupts. Tasks can call low level commands, signal events to the upper layers and can in turn generate more tasks. Since the tasks scheduler is a simple FIFO scheduler they do not have any priority scheme. Also once all the tasks are executed TinyOS goes into sleep mode and only wakes up when some external event takes place. TinyOS does not support dynamic memory allocation but this makes it possible to optimize the application at compile time and know its exact memory requirements. TinyOS also has a very quick response time and propagates events in the time it takes to copy 1.25 bytes of memory and context switches in the time it takes to copy 6 bytes of memory and supports two level scheduling [22]. Lastly TinyOS is also intended to be incorporated into ‘smart – dust’ which are essentially motes with micrometer dimensions.

2.3 nesC

nesC (**N**etwork **E**mbedded **S**ystems **C)** is a programming language designed to build applications for the Tiny Operating System. It is a dialect of the C programming language and supports all mathematical functions and data types supported by the C language. It was developed by NEST research group of UCB in co – operation with Intel Research. nesC does not allow dynamic memory allocation and the calls to other functions and variables is already known, this helps in the whole program analysis and optimizations at compile time which makes it very reliable. This is needed for applications of sensor networks where motes running applications developed in this language will be unmanned for months together. nesC supports the special needs of sensor networks i.e. it is event driven, supports concurrent behavior and component based application design.

In sensor networks processing work only arises when an event occurs like the sensor on the mote detects a change in the environment variable sensed or some packet is received through the communication component. nesC by supporting this helps in reducing power consumption rather than having interactive and batch processing capabilities.

The concurrency model of nesC is simple and allows high concurrent operations and uses very little resources as compared to stack based thread mechanism which consume a large amount of precious memory. Resource contention which occurs because of concurrency is solved by rejecting the request of the new user. Like if a communication packet is already being sent and a request to send another packet comes,

this new request will be rejected with the signaling of an error. But if the need arises the communication component can be designed to provide a queued send interface. As in any concurrent system there is a possibility of deadlock and data races. But in nesC as the whole call – graph is already known at compile time race conditions can be detected and signaled as an error. This can be removed by the programmer by explicitly declaring the code as atomic so that each update to the data by the different components will be executed to completion and the race condition avoided.

A component in nesC provides and uses interfaces. These interfaces are the only point of access to the component [23]. Interfaces are bidirectional in nature and contain commands and events. For all the interfaces provided by the components the commands of those interfaces have to be defined within the component and the events of these interfaces have to be defined by the components which will be using these interfaces. Also for all the interfaces used proper linking has to be defined. This is called wiring in nesC and is similar to function pointers. By grouping the commands and events together in an interface helps in the clean modeling of split – phase operations. For e.g. the 'send' interface designed for TinyOS has the 'send' command and 'sendDone' event of the split – phased packet send. Components using this 'send' interface will be giving a 'send' command and will in turn have to implement the 'sendDone' event through which it will be notified of the packet been sent on the radio. In nesC there are two types of components, modules and configurations. Each are written in separate files the module file name end with a capital 'M' after the component name and the configuration file name end with a capital 'C' after the component name. Each

application also has a top level configuration file which links all the components used and its file name does not end with a 'C'. Modules provide the application code for the interfaces provided by the component and the configurations is used to wire the interfaces used by the component. The commands of the interface being implemented in modules are written with C – like code with a few extra keywords. Like 'call' which is used to call a command used by the component, 'signal' is used to signal an event to the component using the interface, 'command' is used when defining the command during implementation and 'event' is used while defining the event during implementation. It is also possible to create abstract components in nesC, by declaring a component 'abstract' with optional parameters. These abstract components are created at compile – time in configurations. This component model also allows alternate implementations and a flexible hardware/software boundary [23].

2.4 Hardware

2.4.1. Mica2 mote

Mica2 mote is an upgraded version of Rene and was designed again by the NEST group of UCB and manufactured in collaboration with Cross Bow Technology (XBow). They come in three models according to their RF frequency band: the MPR400 (915 MHz), MPR410 (433 MHz) and MPR420 (315 MHz). In this thesis the MPR410 (433 MHz) were used for the experiments.

The Mica2 mote has a high performance, low power 8 – bit microcontroller, the Atmega 128L manufactured by Atmel Corporation. It uses an 8 MHz crystal oscillator.

Mica2 also has 128k bytes Program Flash Memory, 512k bytes Measurement Flash and 4k bytes Configuration EEPROM. It also has a 10 bit Analog to Digital Converter (ADC), DIO, I2C and SPI interfaces.

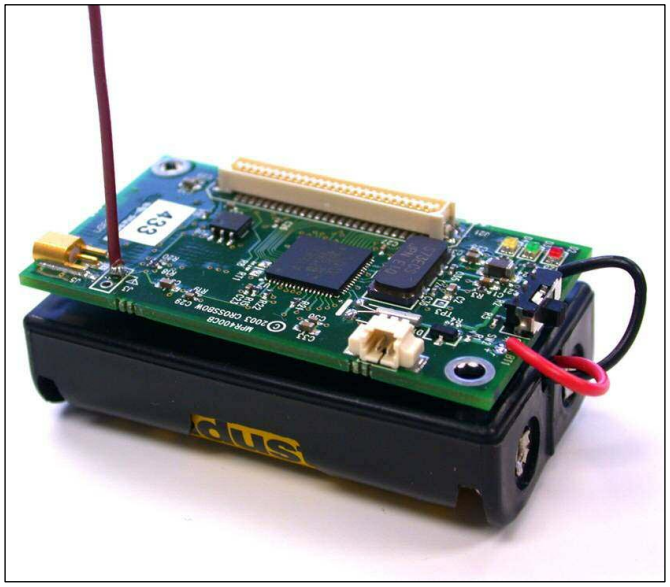

Figure 2.1: Mica2 Mote manufactured by Cross Bow Technology

For the radio communications it uses the CC1000 Chip manufactured by Texas Instruments Chipcon Corporation which works in the 433 MHz band (433.05 – 434.79 MHz). This chip can be programmed to work in any of the 4 channels available in this frequency band. More details about this chip will be discussed in the next chapter.

Because of the small physical size of the mote, the usual antenna chosen is a length of insulated wire called the monopole whip antenna one – quarter wavelength long. This happens to be 6.8 inches for the 433 MHz Mica2 [24].

Mica2 is powered by two AA batteries and requires a voltage between 2.7 – 3.3 Volts for successful operation. It also has three different colored LED's which can

be used for debugging or to signal some event. There is a 51 – pin male Hirose connector onboard the Mica2 which is normally used for connecting the sensor board or for connecting the Mica2 to the programmer board for programming.

2.4.2. MTS310 Sensor Board

The MTS310 Sensor board manufactured by Crossbow Technology is shown below.

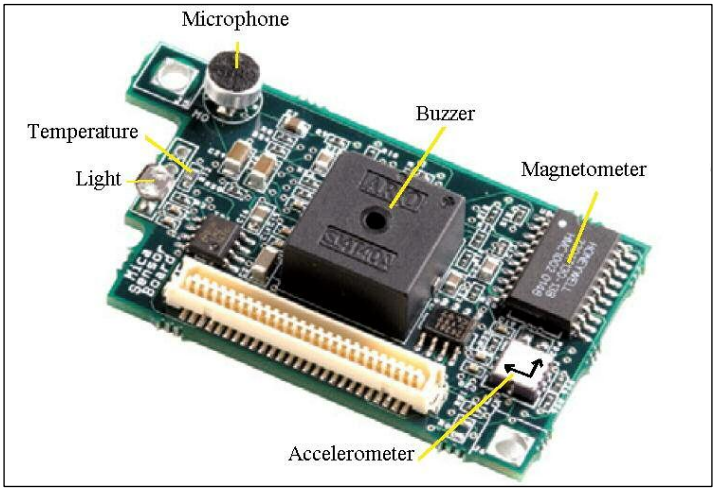

Figure 2.2: MTS310 Sensor Board for Mica2 mote manufactured by Cross Bow Technology

It has the following sensors Light, Temperature, Microphone, Buzzer, 2 – axis Accelerometer and 2 – axis Magnetometer.

2.4.3. MIB510 Programming Board

The applications written on computer using nesC for Mica2 mote are transferred to the mote using a programming board. The programming board used in the experiments is the MIB510 programming board manufactured by Cross Bow Technology. It supplies power to the devices through an optional external power adapter and also has a RS – 232 Mote Serial port and reprogramming port. It has an

onboard in – system processor (ISP) – an Atmega 16L. The code is downloaded from the computer to the ISP through the RS – 232 serial port and the ISP in turn programs the code into the mote. The ISP runs at the fixed baud rate of 115.2 kBaud. The programming board also has two LED's that give the status of the ISP. MIB510 also has a JTAG connector through which the Atmel JTAG pod can be connected for in – circuit debugging. The optional power adapter accepts a 5 – 7 Volts DC and supplies a regulated 3 Volts DC to the Mica2 mote. Following figure shows the MIB510 programming board.

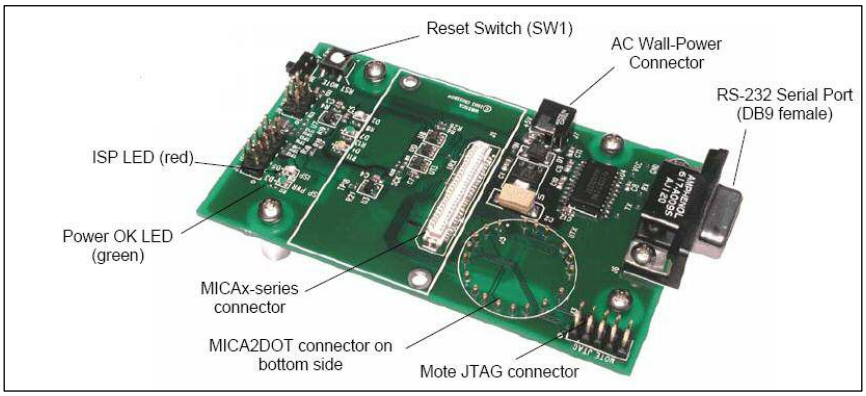

Figure 2.3: Cross Bow Technology's MIB510 Programming Board

2.5 LabVIEW

LabVIEW short for **Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench is a platform and development environment for a visual programming language from National Instruments. The graphical language used is called "G" which is a dataflow language. Execution is determined by the structure of the graphical block diagram on which the programmer connects different function – nodes by drawing wires. The variables pass through the wires and when the node has all the variables it

requires then only it starts with processing of the data and generates the output variables which are then passed to the other nodes or are displayed as output. This also makes parallel execution of the nodes possible. LabVIEW supports polymorphism and the wires that carry the variables automatically change according to the data type of the input. LabVIEW provides script nodes to execute math scripts. These nodes support the Matlab language syntax and the Xmath syntax.

LabVIEW programs are called Virtual Instruments (VIs). Each VI has three components, a block diagram, a front panel and a connector plane. The whole connector plane of one program can be included as a node in the block diagram of another program i.e. a VI can be included as a sub VI in another program. This makes it easier to test each and every sub VI created by the programmer before being implemented in the main program and helps in building robust applications. LabVIEW has a huge library with a large number of functions for data acquisition, signal generation, mathematics, statistics, signal conditioning, analysis, along with numerous graphical interface elements [25]. Thus the programmer can quickly generate the required front panels by just dragging and dropping the controls or indicators from the library onto the front panel window. The front panel severs two purposes when the VI is used as a stand alone application the front panel can be used to get the user input and to display the output and when the VI is used as a sub VI inside another VI the front panel defines the inputs and outputs of the nodes. LabVIEW has an extensive support for accessing instrumentation hardware. Drivers and abstraction layers for many different types of

instruments and buses are included in the library. These are in the form of graphical nodes and offer standard software interfaces to communicate with hardware devices.

In this thesis LabVIEW was used to collect and store the data transmitted by the base station mote over the serial port. Since the LabVIEW library itself provided all the nodes necessary to initialize and read data on the serial port the programming time to build the application was drastically reduced. Further the graphical programming made decoding and converting the data to the required format much easier. A few VIs created for implementations in this thesis are given in Appendix A.

LabVIEW is also helpful for building large applications because as VIs increase in functionality and complexity, the increased visual size of the code encourages the programmer to adopt a modular approach which ensures better organization. This architecture, along with continuous flow of data simplifies debugging and maintenance. LabVIEW has built nice tools for debugging which actually shows the flow of data throughout the entire application. The execution can also be stopped in between during debugging to check the exact value of data. Thus one can quickly identify the error and its position which can be promptly rectified.

CHAPTER 3

CHIPCON'S CC1000 TRANSCEIVER

3.1 Introduction

CC1000 is a true single chip Ultra High Frequency (UHF) transceiver designed, for very low power and very low voltage wireless applications, by Texas Instrument's Chipcon Corporation. The transceiver is build to operate in the Industrial, Scientific and Medical (ISM) bands of 315, 433, 868 and 915MHz. It is mainly designed for use in Short Range Devices (SRD). The operating frequency of CC1000 is programmable and can vary from 300 – 1000 MHz. Since the frequency is programmable frequency hopping protocols can be easily implemented. CC1000 requires a low supply voltage of 2.1 volts to a max of 3.6 volts. It uses the Binary Frequency Shift Keying (BFSK) modulation in the physical layer. The transmit power and the baud rate are completely programmable and their different values were used in the experiments done in this thesis. The transceiver also has an output pin using which the Received Signal Strength Indicator (RSSI) can be measured. There is no need of any external RF switch / IF filter with this chip [26]. Actually very few additional components are required for use along with the chip. It interfaces to the microcontroller using the 3 – wire serial configuration interface (PDATA, PCLK and PALE) and the bi – directional synchronous data signal interface (DIO and DCLK). There are inbuilt registers using which the operating parameters of CC1000 can be programmed. The company has also developed software

"Smart RF Studio" for generating the configuration data. This software is freely available for download from the company's website. By choosing the required parameters in the software the setting of all the configuration parameters are given. There is also a pre – programmed Microsoft Excel file available from the company in which, if one gives the operating frequency, crystal oscillator frequency, frequency separation and specifies if Local Oscillator is on High side or Low Side then all the other correct parameter values are calculated and shown for use in the configuration registers. The company also provides with a development kit for CC1000.

3.2 Register Settings and Parameter Values

CC1000 has twenty two (22) 8 – bit configuration registers which can be set according to the required parameter values. A few important register settings and parameter values are discussed next.

3.2.1 Frequency Parameters

Six registers as shown below are used to control the frequency synthesizer (PLL).

Table 3.1 CC1000 Frequency registers [26]

Address	Byte Name	Description
01h	FREQ_2A	Frequency Register 2A
02h	FREQ_1A	Frequency Register 1A
03h	FREQ_0A	Frequency Register 0A
04h	FREQ_2B	Frequency Register 2B
05h	FREQ_1B	Frequency Register 1B
06h	FREQ_0B	Frequency Register 0B
07h	FSEP1	Frequency Separation Register 1
08h	FSEP0	Frequency Separation Register 0

Frequency registers 2A, 1A and 0A together give the 24 bit frequency control word A and similarly 2B, 1B and 0B give the frequency control word B. 2A and 2B holds the 8 – MSB bits of the respective control word and 0A and 0B hold the 8 – LSB bits of the respective control word. For experiments in this thesis the Frequency Control Word A had a value of 0x580000 decimal – 5767168 and B had a value of 0x57F685 decimal – 5764741. Frequency Control Word A is used to set the local oscillator frequency in receive mode and Frequency Control Word B is used to set the transmitting frequency, f_0 in the transmit mode. The MAIN.F_REG control bit in the MAIN register (not shown) is used to select the frequency word A or B.

Frequency Separation Register 0 and 1 together give the Frequency Separation Control Word. But this word is only 11 bits long. FSEP0 holds the 8 – LSB bits of the word and FSEP1{2:0} i.e. bit #2:0 holds the 3 – MSB bits of the word. FSEP1{7:3} is not used for any parameters. The two registers together hold a value of 0x355 decimal 853.

Transmit and receive frequencies can be calculated using the following formula [26].

$$f_{VCO} = f_{ref} \times \frac{FREQ + (FSEP \times TXDATA) + 8192}{16384} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3.1)}$$

In the above formula f_{VCO} gives the Local Oscillator (LO) frequency in receive mode and the f_0 and f_1 frequency in transmit mode (lower and upper FSK frequency). f_{ref} is the reference frequency calculated using the formula and register settings given below. *FREQ* is decimal value in the frequency control word A or B according to whether the

formula is used to calculate receive frequency or transmit frequency. *FSEP* is the decimal value in the FSEP0 and 1 registers given above. *TXDATA* is 0 or 1 in transmit mode depending on the data bit to be transmitted. In the receive mode *TXDATA* is always 0.

Another register that we need to look at is the PLL register.

Table 3.2 CC1000 PLL register [26]

Address	Byte Name	Description
0Ch	PLL	PLL Control Register

The 8 bits in this register are used to set different parameters. Bit #0 i.e. PLL{0} is used for ALARM_L, PLL{1} for ALARM_H, PLL{2} for ALARM_DISABLE, PLL{6:3} for REFDIV{3:0} and PLL{7} for EXT_FILTER. Of these we require REFDIV which gives the reference divider and is used in the formula below to calculate f_{ref}. The value set for REFDIV is 0xC decimal – 12.

The crystal oscillator connected to CC1000 has a frequency of 14.7456 MHz.

Now to calculate f_{ref} we use the following formula [26]

$$f_{ref} = \frac{f_{xosc}}{REFDIV} \quad \text{……...…….. (3.2)}$$

f_{xosc} is the crystal oscillator frequency of 14.7456 MHz and *REFDIV* as mentioned above is 12. This gives f_{ref} = 1.2288 MHz.

Now that we have got all the required values we can find the value of f_{VCO}.

In receive mode $f_{VCO} = 1.2288 \times 10^6 \times \frac{5767168 + (853 \times 0) + 8192}{16384} = 433.152 \times 10^6 \, Hz$

In transmit mode $f_{VCO} = 1.2288 \times 10^6 \times \frac{5764741 + (853 \times 0) + 8192}{16384} = 432.96997 \times 10^6 Hz$

Thus the LO frequency in the receive mode is 433.152 MHz. In the experiments we have used a high side LO injection. So $f_{VCO} = f_{RF} + f_{IF}$ where f_{RF} is the centre frequency and f_{IF} is the Intermediate frequency. For CC1000 f_{IF} is designed to be 150.0375 KHz. So we get f_{RF} = 433.152 MHz – 150.0375 KHz = 433.0019625 MHz.

The upper FSK transmit frequency is given by $f_1 = f_0 + f_{sep}$. f_0 is the f_{VCO} frequency calculated above for transmit mode i.e. 432.96997 MHz and f_{sep} is the frequency separation calculated using the formula [26] given below.

$$f_{sep} = f_{ref} \times \frac{FSEP}{16384} \quad \text{..... (3.3)}$$

Using the value of f_{ref} and *FSEP* found above we get

$f_{sep} = 1.2288 \times 10^6 \times \frac{853}{16384} = 63975 Hz$ further using this f_{sep} we get

f_1 = 432.96997 MHz + 63975 Hz = 433.03395 MHz. From f_0 and f_1 the centre frequency will be mid way between the two which comes to 433.0019625 MHz and is same as in the receive mode.

The following figure shows the relation between f_{VCO}, f_{IF} and LO frequency.

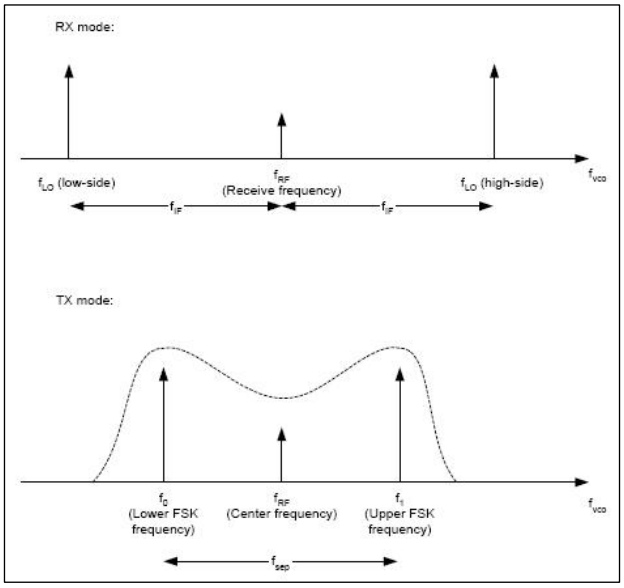

Figure 3.1: Relation between f_{VCO}, f_{IF} and LO frequency [26]

3.2.2 Transmit Power

CC1000 being designed for low power applications offers great flexibility in power management. If the radio is not used it can be set to power down mode by setting some parameters in the MAIN register. During normal operation the transmit power can be varied from -20 dBm to 10 dBm in steps of 1 dB. This transmit power is controlled by the PA_POW register shown in the table below.

Table 3.3 CC1000 PA_POW register [26]

Address	Byte Name	Description
0Bh	PA_POW	PA Output Power Control Register

The PA_POW register has two parts PA_POW{3:0} i.e. bit #3:0 is used to control the output power in low power array and PA_POW{7:4} is used to control the output power in high power array. The value in both these register parts together set the output transmit power according to the following table.

Table 3.4 PA_POW register values and corresponding output power [26]

Output Power (dBm)	PA_POW Register Value (hex)
-20	01
-19	01
-18	02
-17	02
-16	02
-15	03
-14	03
-13	03
-12	04
-11	04

Note: The other output power settings are not shown in the above table as they were not used in the experiments in this thesis.

3.2.3 Baud Rate, Data Format and Oscillator Frequency

Another important register of CC1000 used in the experiments is the MODEM0 register.

Table 3.5 CC1000 MODEM0 register [26]

Address	Byte Name	Description
11h	MODEM0	Modem Control Register 0

This register is divided into four parts MODEM0{7} i.e. bit #7 is not used, MODEM0{6:4} gives the BAUDRATE{2:0}, MODEM0{3:2} gives the DATA_FORMAT{1:0} and MODEM0{1:0} gives the XOSC_FREQ{1:0} which tells CC1000 the frequency range of the crystal oscillator connected. The following table shows the possible bit values of each part of the register and the corresponding parameter values.

Table 3.6 MODEM0 register parts [26]

Register	Part Name	Bit & Parameter Value
MODEM0{7}	-	Not Used
MODEM0{6:4}	BAUDRATE{2:0}	000: 0.6 kBaud 001: 1.2 kBaud 010: 2.4 kBaud 011: 4.8 kBaud 100: 9.6 kBaud 101: 19.2, 38.4 and 76.8 kBaud 110: Not Used 111: Not Used
MODEM0{3:2}	DATA_FORMAT{1:0}	00: NRZ operation 01: Manchester operation 10: Transparent Asynchronous UART operation 11: Not Used
MODEM0{1:0}	XOSC_FREQ{1:0}	Selection of XTAL frequency range 00: 3 MHz – 4 MHz crystal, 3.6864 MHz recommended Used for 76.8 kBaud, 14.7456 MHz 01: 6 MHz – 8 MHz crystal, 7.3728 MHz recommended Used for 38.4 kBaud, 14.7456 MHz 10: 9 MHz – 12 MHz crystal, 11.0592 MHz recommended 11: 12 MHz – 16 MHz crystal, 14.7456 MHz recommended

To set a baud rate of 19.2 kBaud MODEM0{6:4} should have a value of 101 binary and since the crystal oscillator connected to CC1000 has a frequency of 14.7456 MHz, MODEM0{1:0} should have a value of 11 binary. Further to set 38.4 kBaud MODEM0{6:4} should again have a value of 101 binary but this time MODEM0{1:0} should be set to 01 binary in spite of the crystal being of 14.7456 MHz. CC1000 will detect the proper crystal frequency and set the baud rate to 38.4 kBaud. Similarly to get 76.8 kBaud MODEM0{6:4} should be set to 101 binary and this time MODEM0{1:0} should have 00 binary.

In the experiments the DATA_FORMAT i.e. MODEM0{3:2} was always set to 01 binary. Manchester encoding was used on the data. Hence the actual bit rate used on the channel will be half of the baud rate. For 19.2 kBaud we get 9.6 kbps, for 38.4 kBaud – 19.2 kbps and for 76.8 kBaud – 38.4 kbps.

3.2.4 RSSI

CC1000 has an output pin on which one can measure the Received Signal Strength Indicator (RSSI) value. Actually this particular pin is assigned two tasks either it can output RSSI value or can give a 10.7 MHz IF output buffer which can then be connected to a separate external demodulator. On the Mica2 mote this particular pin is used to read the RSSI value. To do this a particular register as shown below should be set so that CC1000 outputs RSSI value on the required pin instead of IF buffer output.

Table 3.7 CC1000 FRONT_END register [26]

Address	Byte Name	Description
0Ah	FRONT_END	Front End Control Register

There are five parts in this register FRONT_END{0} i.e. bit # 0 is used as XOSC_BYPASS, FRONT_END{2:1} gives IF_RSSI{1:0}, FRONT_END{4:3} gives LNA_CURRENT{1:0}, FRONT_END{5} gives BUF_CURRENT and FRONT_END{7:6} is not used. Of these we are only interested in IF_RSSI{1:0} and the possible values of this part are shown below.

Table 3.8 FRONT_END register part # 2 [26]

Register	Part Name	Bit & Parameter Value
FRONT_END{2:1}	IF_RSSI{1:0}	Control of IF_RSSI pin 00: Internal IF and demodulator, RSSI inactive 01: RSSI active, RSSI/IF pin is analog RSSI output 10: External IF and demodulator, RSSI/IF pin is mixer o/p. Internal IF in power down mode. 11: Not Used

So to get RSSI, bit # 2 – 1 of the FRONT_END register of CC1000 should be set to 01 binary. The RSSI output on this pin is an analog voltage value. To convert this to dBm we need to use the following formula [26].

$$RSSI(dBm) = (-51.3 \times V_{RSSI}) - 49.2 \quad \text{..... (3.4)}$$

Here V_{RSSI} is the output voltage on RSSI/IF pin of CC1000. On Mica2 this pin is connected to ADC pin of microcontroller. The microcontroller just gives a 10 – bit value which then needs to be processed using the formula [24] given below to get the analog value of V_{RSSI} which when used in formula 3.4 above gives RSSI in dBm.

$$V_{RSSI} = \frac{V_{batt} \times ADC\_Counts}{1024} \quad \text{..... (3.5)}$$

In the above formula ADC_Counts is the 10 – bit value of ADC that we get from the microcontroller and V_{batt} is the remaining battery voltage of Mica2. The battery voltage is measured using a component defined in TinyOS and will be explained while discussing the details of the experiment conducted.

CHAPTER 4

EXPERIMENT DETAILS

4.1 Introduction

This thesis deals with the energy conservation in Wireless Sensor Networks created using Mica2 motes. In sensor networks, sensors are densely deployed and normally don't require huge power to transmit at large distances. Data is transmitted to the far away base station using multihop routing [7, 8] and clustering protocols [12, 13, 14]. Localization algorithms [27, 28, 29] are also available which can be run on sensor networks. These give the distance between the neighboring motes. It is known that in all wireless channels there is power loss called path loss i.e. the received power is less than the transmitted power. This path loss is inversely proportional to the distance. So once we know the distance we can find the power required at the transmitter for successful reception of data. All the radios have a fixed distance up to which it can transmit at a certain power level and a certain Bit Error Rate (BER). This distance is called the transmission distance.

Another important factor that affects the transmission distance is the antenna sensitivity. Baud rate of the data affects the receiver antenna sensitivity. This is because at the higher baud rate, there is less energy and a fewer number of actual radio waves in each bit of information transmitted. The lower baud rates have more energy per bit and more actual radio waves per bit transmitted. Correctly receiving each bit of information

requires the radio receiver establish a waveform pattern similar to the transmitted waveform. The more radio waves received per bit makes it much easier to establish that waveform. Thus, lower baud rates mean more energy per bit; better receive signals and longer transmission distances [30]. Since in sensor networks motes mostly sense the environment, the frequency of this sensing or the duty cycle is very low. Most of the time the mote is idle. So operating at a lower baud rate which takes a little bit more time to transmit is not a problem. The advantage is by keeping the transmit power constant and only decreasing the baud rate we can increase the transmission distance of the radio.

Thus the two important factors that affect the transmission distance are transmission power and baud rate. The radio on Mica2 motes has a capability to transmit at different transmit power levels and baud rates as explained in chapter 3. The multihop routing and clustering protocols implemented on Mica2 motes does not take both these factors into account for power conservation. These only rely on finding the best path to conserve energy and always keep the transmit power and baud rate at a fixed level. Even when the neighboring mote is very close the power and baud rate are never changed. This unnecessarily wastes precious power when transmitting at short distances. Also if the duty cycle of transmission is low we can use lower baud rate and lower transmit power and increase the transmission distance.

There is another advantage to using this strategy. By lowering the transmit power only the required mote and a few close ones hear the packet transmission. Motes

far away do not hear anything and are free to do their own transmission with some other motes. This is explained in the following figures.

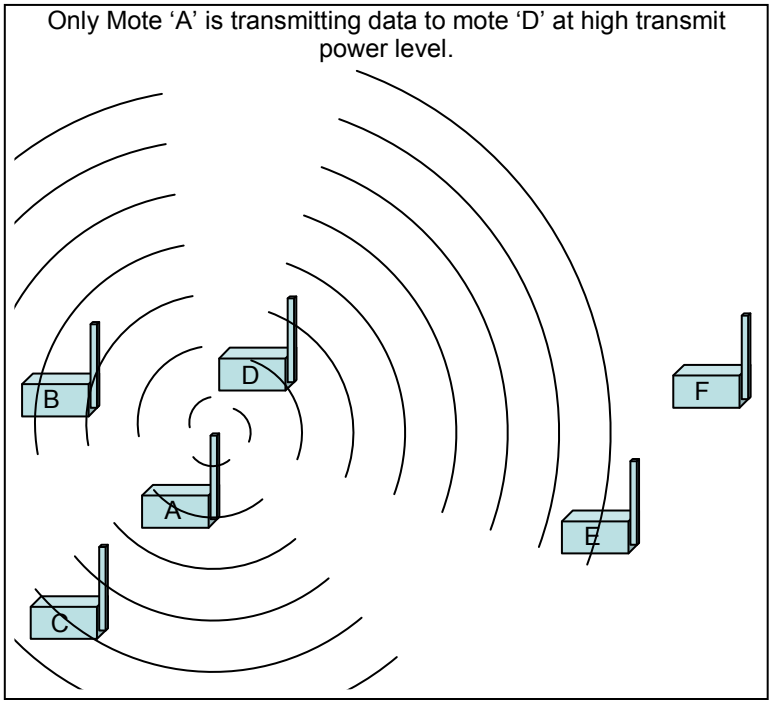

Figure 4.1: Transmission between motes at high transmit power

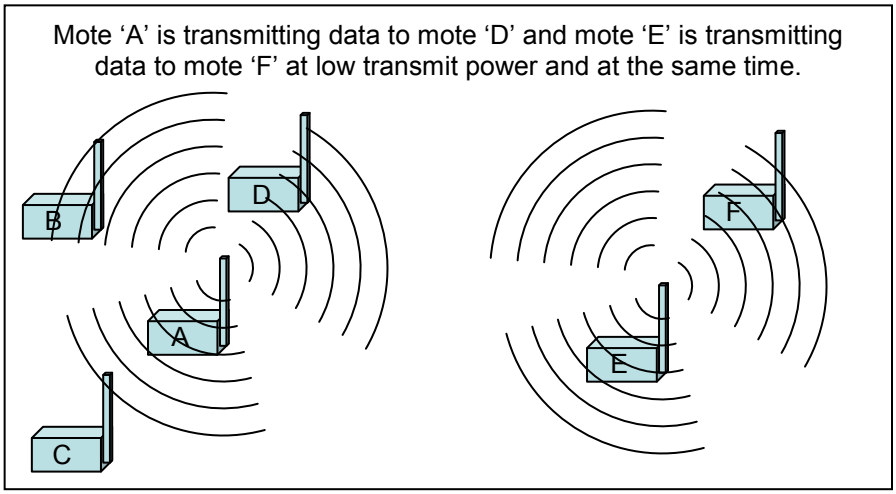

Figure 4.2: Transmission between motes at low transmit power

In figure 4.1 above mote ‘A’ is transmitting data to mote ‘D’ at high power hence apart from mote ‘D’ hearing the conversation mote ‘E’ also hears the conversation. Ultimately mote ‘E’ just drops the packet got from mote ‘A’ as it is not intended for mote ‘E’. During the time mote ‘A’ is transmitting the data, mote ‘E’ cannot transmit data to mote ‘F’ as it will find the channel busy. On the other hand as shown in figure 4.2 if mote ‘A’ is using low power to transmit data to mote ‘D’, mote ‘D’ clearly hears the conversation but mote ‘E’ being a little far cannot hear the conversation. The channel around mote ‘E’ will be idle and so mote ‘E’ can begin transmitting data to mote ‘F’ at the same time as mote ‘A’ is transmitting data to mote ‘D’.

Thus our aim in this thesis is to gather empirical data, using Mica2 motes, about transmission distance at different transmit power levels and baud rate. Then analyze this data and find a relation between these three quantities i.e. transmit power, baud rate and distance.

4.2 Experiment Outline

For the experiment one of the motes acts as the transmitter and the other acts as the receiver. For the transmitter mote a certain power level and baud rate is set from the possible options available. Since the wireless channel varies a lot randomly with respect to time there will be some packet loss even when the receiver mote is within the transmission distance. Hence from a fixed distance the transmitter mote transmits a large number of packets each with a sequence number. The receiver mote already

knows the total number of packets it should receive and once it receives all the packets it transmits a summary packet to the computer attached to it. If for a long time the receiver does not receive any packet it assumes it cannot receive any more packets and sends the required data about number of packets received till then to the computer. The distance between the transmitter and receiver is increased in increments of 5 inches and the transmitter again starts transmitting the packets. Once it is found that the receiver is not receiving any packets at all we know that the receiver is out of the transmission range. To be fully sure, the experiment is carried at a few more distances. When it is absolutely sure the receiver is out of range the transmit power or baud rate on the transmitter mote are changed to a different value and the above experiment carried again.

4.3 Requirements and Location

For the hardware the experiment required two Mica2 motes. As said above one of the motes was the transmitter and the other receiver. To program these Mica2 motes a MIB510 parallel programming board was used, it was powered using a power adapter by an external power supply. The transmitter mote was powered by two AA batteries 1.5 volts each. The receiver mote also acted as the base station and after programming was always mounted on the programming board. The programming board was connected to the computer through the RS – 232 serial cable.

For the software part a program was written in nesC which will run on both the Mica2 motes. The transmitter mote program will transmit packets with sequence

numbers and the receiver mote program continuously listen the channel to receive the packets. More about the program and other TinyOS settings are discussed in the next section. There is another program continuously running on the computer which listen the serial port of the computer for any summary TinyOS packets sent by the base station i.e. the receiver mote. This program was written in LabVIEW. Once LabVIEW program receives the packet it extracts the data decodes and processes it as explained in the next section and writes all the data to a text file. A Matlab program is then used to read all the text files and analyze the data.

The experiment was carried out in a big warehouse which was used as a lab with minimal office furniture. The Mica2 motes were placed on the ground and were always in line of sight of each other. The following picture was taken during the experiment.

(a) (b)

Figure 4.3: Experimental Site (a) Front view (b) Top view of motes

4.4 Implementation Details

4.4.1 TinyOS Parameters Setting

TinyOS ver 1.1.15 that was used does not have any program that can be called to change the baud rate. Hence we need to manually change the value in an array as follows. As discussed in chapter 3 CC1000 transceiver on Mica2 mote has 22 registers, these need to be set with the required values during the initialization of the transceiver. One of the registers set the baud rate. TinyOS has a file CC1000Const.h that has an array called CC1K_Params. The values in this array are used to set the register values of CC1000. This CC1000Const.h file along with some other files are used by the TinyOS communication part. The CC1K_Param array is a 6 x 31 array with each row used for different frequency channel. Row # 1 is used for 433.002MHz channel, on which the experiment was conducted. Column # 13 in Row # 1 is used to set the baud rate. The exact value to use here was discussed in chapter 3. Once the value in this array is changed manually, during compilation of the mote program these above mentioned files will be compiled and linked, since the mote program uses the communication part, and the desired baud rate will be set for the transceiver. Of the available baud rates 76.8 kBaud, 38.4 kBaud and 19.2 kBaud was used in the experiment.

To set the transmit power level the program written in nesC used an interface provided by TinyOS called CC1000Control. This CC1000Control interface is implemented in the CC1000ControlM module of CC1000RadioC configuration. The CC1000Control interface has a function SetRFPower that was called to set the desired

power. The experiment was conducted for four power levels -19.5 dBm, -17 dBm, -14 dBm and -11.5 dBm.

4.4.2 Mote Software Details

There is no hardware support on Mica2 for doing floating point operations. So instead of using the software implemented math library which might increase the processing in Mica2, only integer values were read on Mica2. Whatever floating point operations were needed was done in LabVIEW on the computer.

During the experiment care was taken to always start the receiver mote before the transmitter mote so it does not loose any initial packets. Once the motes are started the software on both the motes initializes and then stores their respective remaining battery voltage. This is required for calculating RSSI in dBm and was explained in chapter 3. For this there is a component available in TinyOS, VoltageC. This component gives the battery voltage in millivolts. Now the receiver mote goes in to receive mode and waits for the packets. If it does not receive any packet for 1 minute, the receiver assumes the transmitter is out of range and stops receiving. It then sends a summary packet through the serial port to the computer which has the following data, transmit power level (if no packet is received then zero), number of packets received, number of packets missed, total number of packets (4000 in this experiment), RSSI value (this is the sum of 10 – bit RSSI value of all packets received), its own battery voltage and battery voltage of transmitter (zero if no packet received).

The transmitter mote on taking its remaining battery voltage starts sending packets to the receiver. Each packet contains a sequence number, its battery voltage and

transmission power level. Only when the packet is successfully sent by the radio, the packet with next sequence number sent. Since the radio channel varies with time in a random manner as said in the section above the transmitter mote sends a total of 4000 packets so as to follow Monte Carlo method.

The receiver mote on receiving the packet from the transmitter uses the sequence number in it and keeps a record of the number of packets received and packets missed. Apart from this it also stores the transmit power level and remaining battery voltage of the transmitter contained in the packet. For each packet received TinyOS provides the RSSI value. This is just the 10 – bit ADC value of the RSSI port. To convert this to dBm we need to use the formulas given in chapter 3. This 10 – bit RSSI value of all the packets is summed together to get the average RSSI. The average is found on the computer and not the mote. The receiver on receiving all the 4000 packets or on time out of receive timer sends the summary packet through the serial port to the LabVIEW program running on the computer as mentioned above.

4.4.3 LabVIEW Outline

On starting LabVIEW, the distance between the transmitter and receiver is manually entered through the front panel. On receiving the packet from the base station the labVIEW software extracts the data part and decodes it to get all the desired fields. Next using the receivers remaining battery voltage and sum RSSI value, the average RSSI value in dBm is calculated by the formulas in chapter 3. It also calculates the percentage of packets successfully received. All this data is then stored in a text file.

The data in the text file is read using Matlab to do the analysis explained in the next chapter. For Important Block Diagrams of the LabVIEW program see Appendix A.

4.5 Empirical Data Plots

Following plots were generated using the data gathered from the experiment.

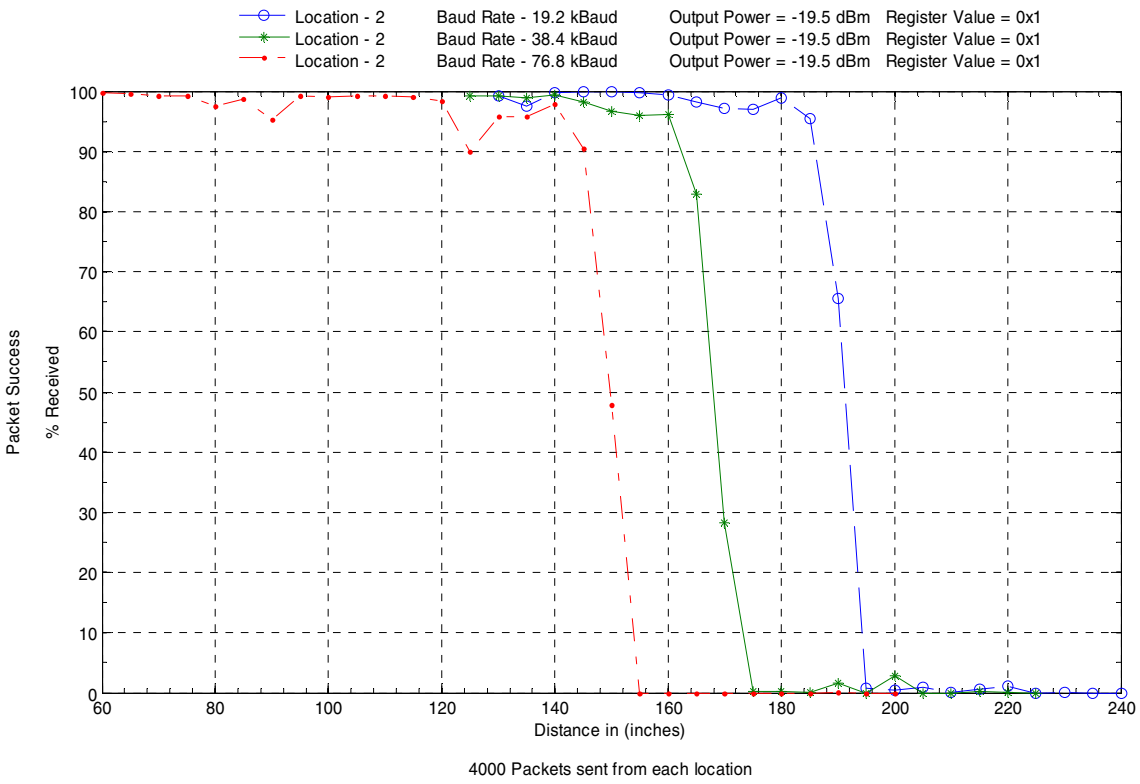

Figure 4.4: Packet success in percentage at power level -19.5 dBm and for 19.2 kBaud, 38.4 kBaud and 76.8 kBaud

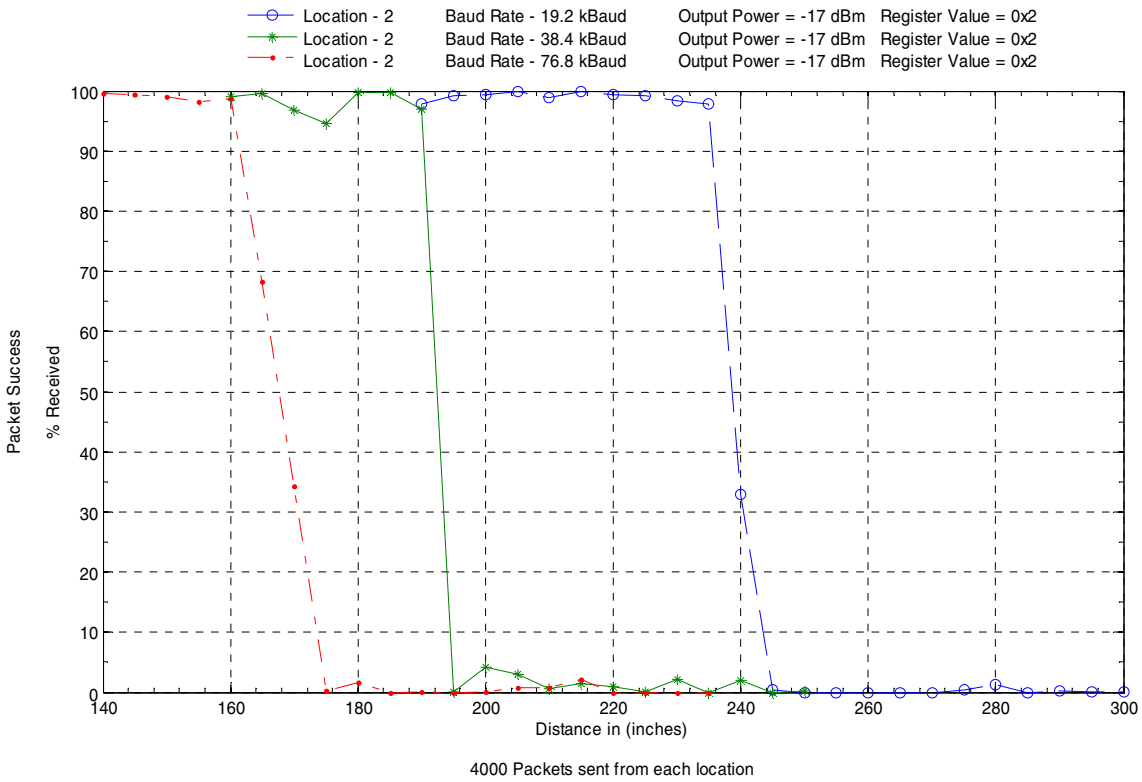

Figure 4.5: Packet success in percentage at power level -17.0 dBm and for 19.2 kBaud, 38.4 kBaud and 76.8 kBaud

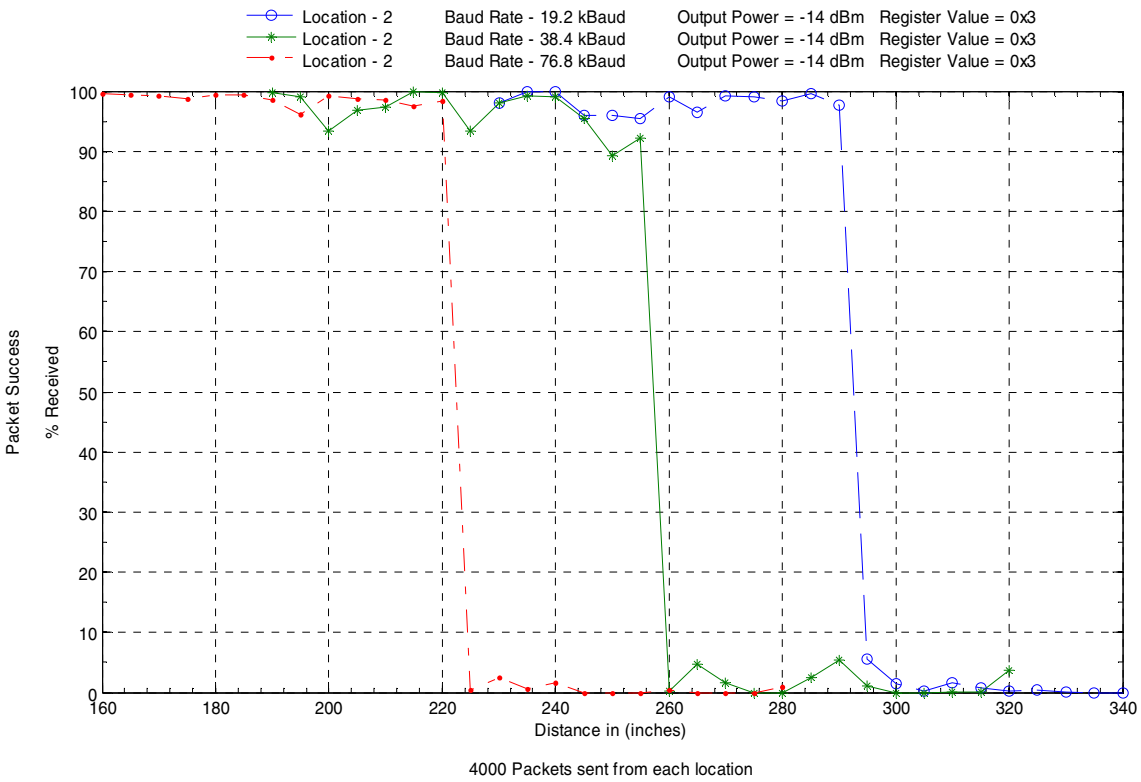

Figure 4.6: Packet success in percentage at power level -14.0 dBm and for 19.2 kBaud, 38.4 kBaud and 76.8 kBaud

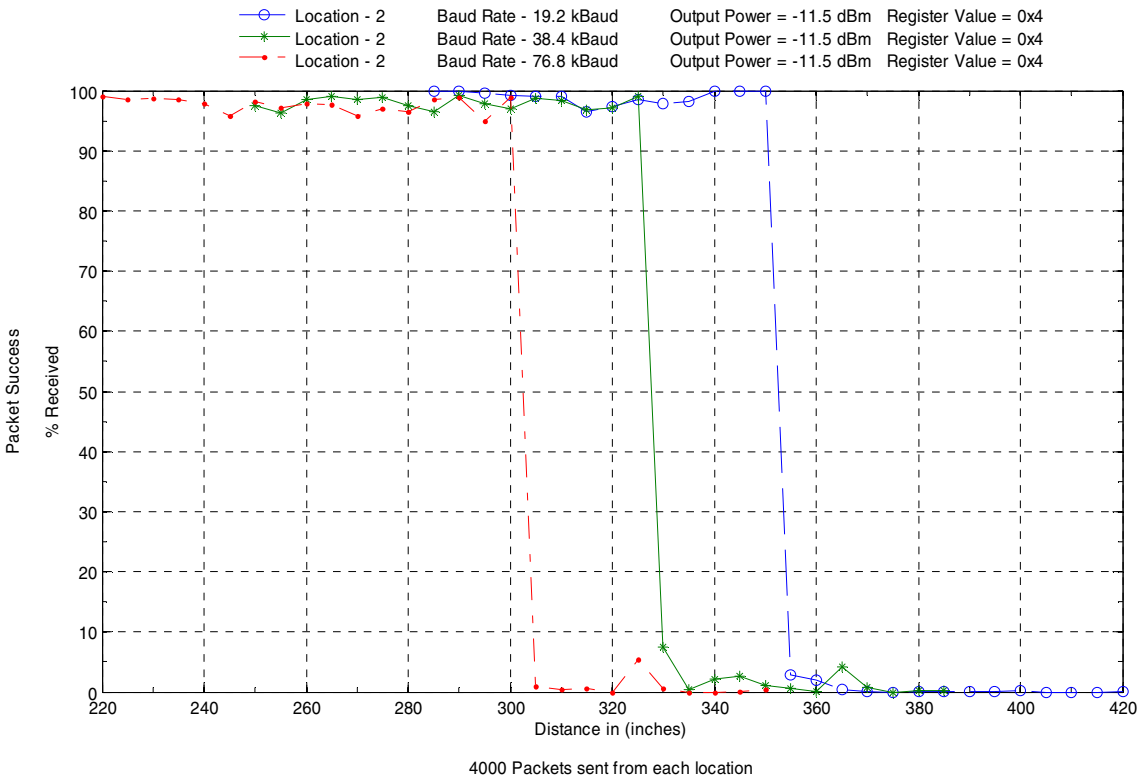

Figure 4.7: Packet success in percentage at power level -11.5 dBm and for 19.2 kBaud, 38.4 kBaud and 76.8 kBaud

From the above figures it is seen that each curve in the plots is initially in the region of 90 – 100 % this is because the transmitter and receiver motes are within the transmission range and the receiver is receiving almost all the packets. A few packets are dropped but this could be because of variations in the channel that occur for a very small amount of time and is normal for any wireless channel. As the distance between the transmitter and receiver is increased, after a certain distance the curve drops drastically and the packets received are in the range of 0 – 5 %. This implies the transmitter is out of range of the receiver mote.

CHAPTER 5

CURVE FITTING & SURFACE FITTING

5.1 Introduction

We have come across a set of data that is thought to be derivable from some underlying function that also fills in the gaps between the data in some way. More precisely, we have a set of function values at some points and wish to find the function that fits this data. The concept of "fits" in the last sentence is in terms of Lagrange's method of least squares where we try to minimize the sum of squared deviations [31]. Mathematically it can be stated using the formula [31] as follows.

$$E(p) = \sum_{i=0}^{N} [p(x_i) - f_i]^2 \quad \text{..... (5.1)}$$

Here E is a functional and assigns a number to p

p is a function such that $p \in P_m$ where $m \leq N$

f_i are the ordinates and

x_i are the (N + 1) abscissas. Using eq. (5.1) we try to minimize E.

5.2 Curve Fitting

The plots in chapter 4 section 4.5 seem to closely follow the Q – function. Each curve in the plots will have a different mean and std. dev. After fitting Q – function to each curve using Method of Least Squares following figures were generated.

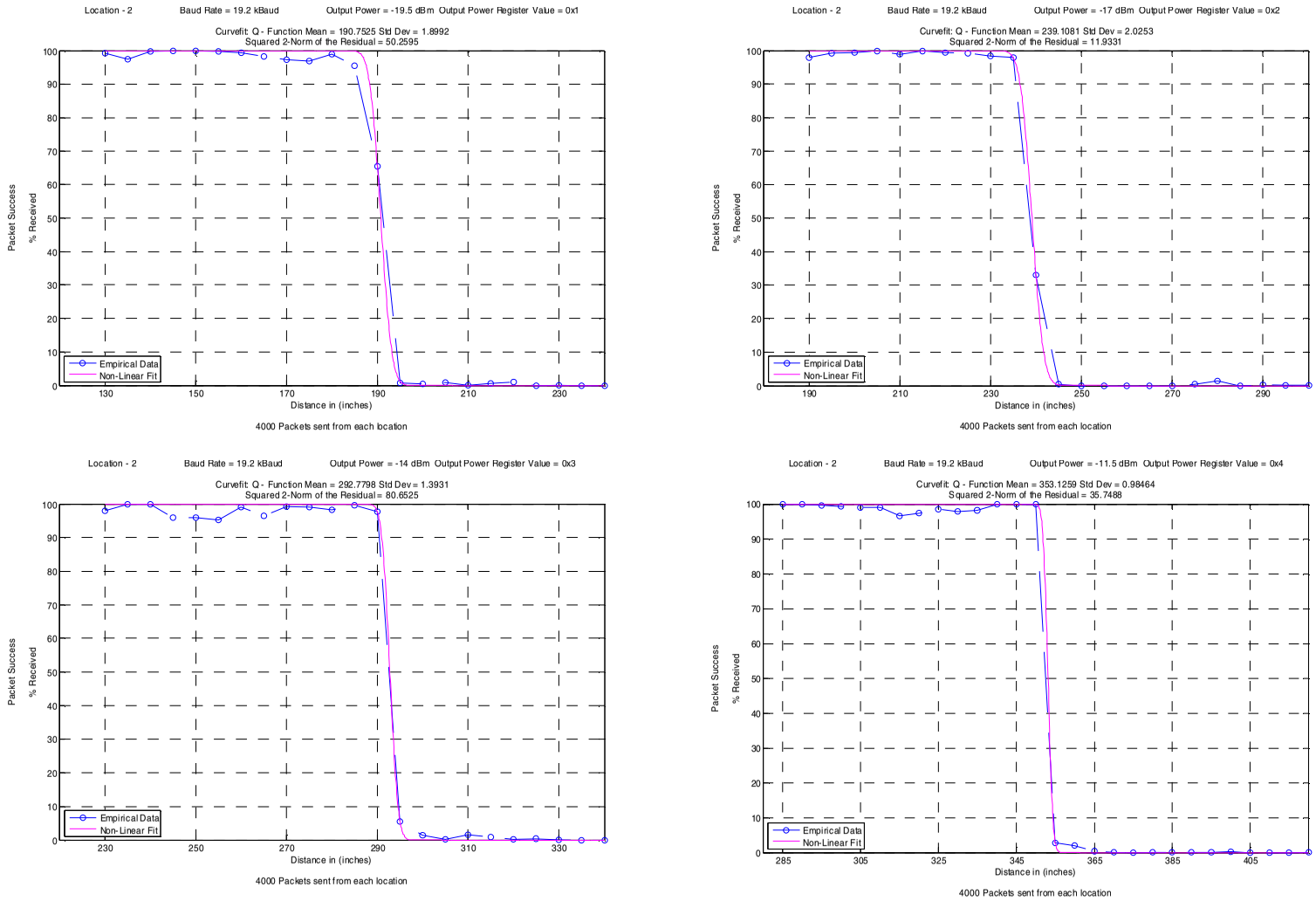

Figure 5.1: Q – function curve fit at 19.2 kBaud and power level of -19.5 dBm, -17 dBm, -14 dBm and -11.5 dBm

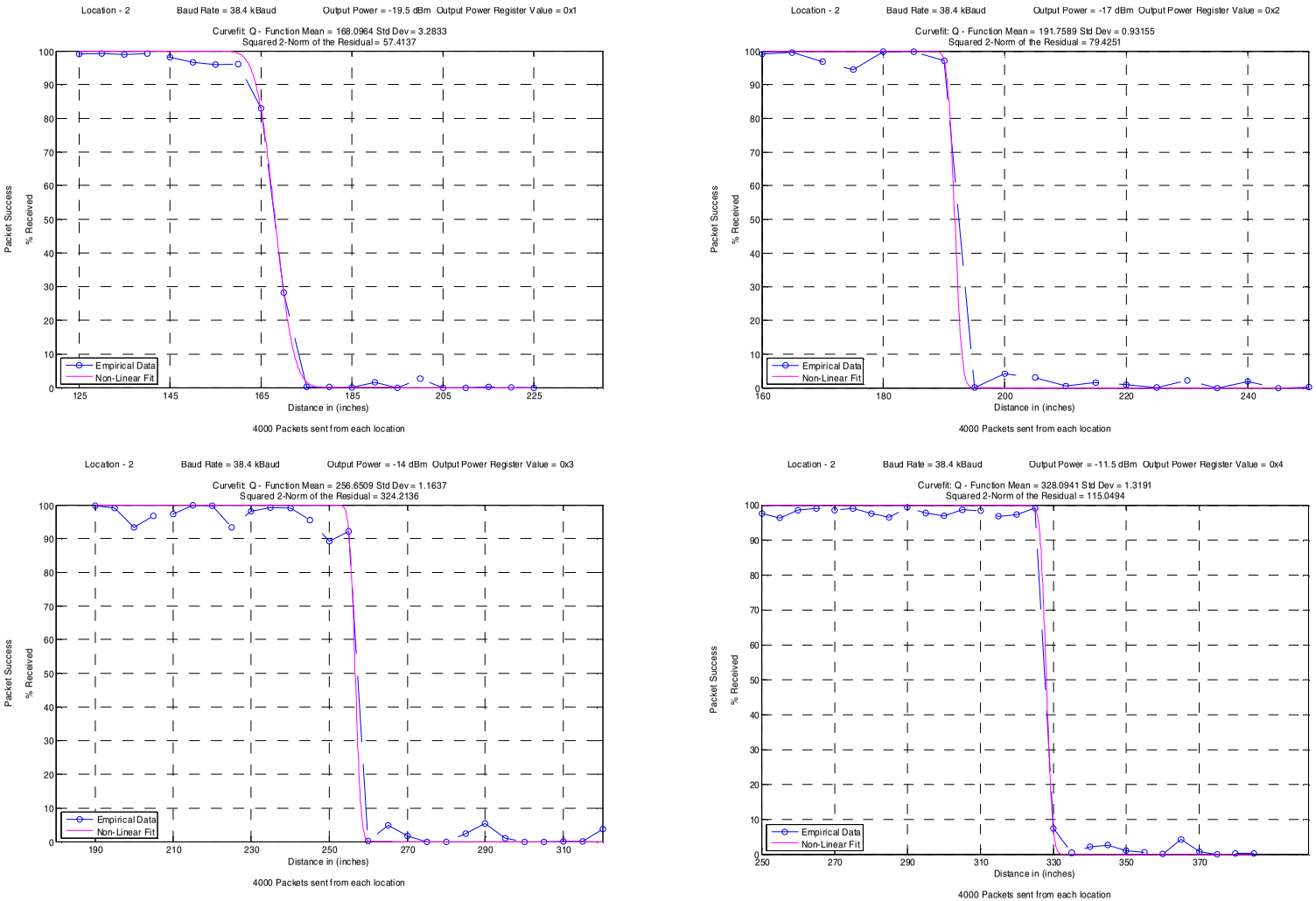

Figure 5.2: Q – function curve fit at 38.4 kBaud and power level of -19.5 dBm, -17 dBm, -14 dBm and -11.5 dBm

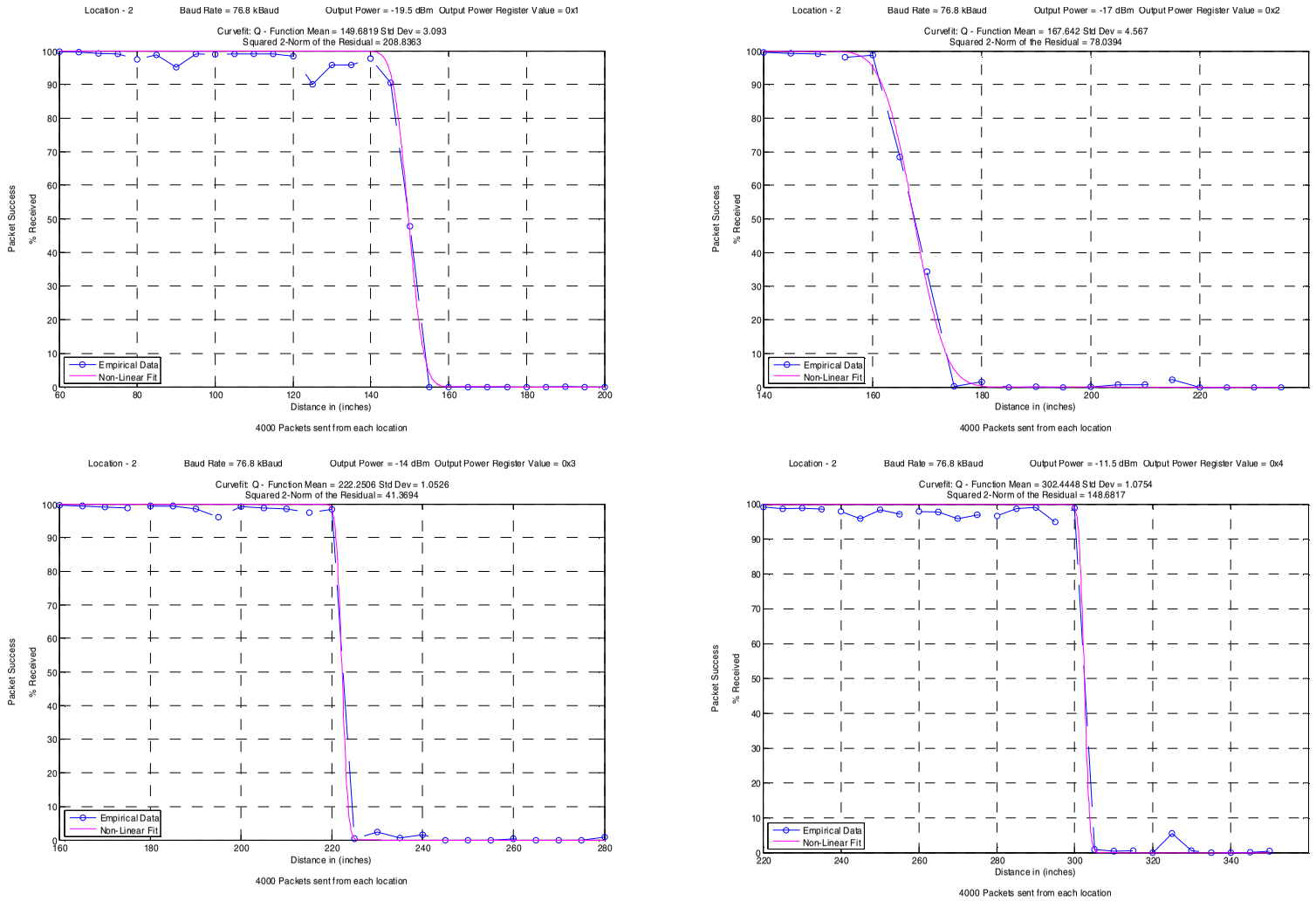

Figure 5.3: Q – function curve fit at 76.8 kBaud and power level of -19.5 dBm, -17 dBm, -14 dBm and -11.5 dBm

An initial value of one was used for std. dev. for curve fit and for mean of distance a value where packet success equals 50 % was used. The following table shows the parameters got from the curve fit in above figures.

Table 5.1 Mean, Std. Dev. and Residual Error of Q – function curves fit above for different baud rate and transmit power

Location Baud Rate	Transmit Power Register Value	Curve Parameters (Inches)		Residual Error
2 76.8 k Baud	-19.5 dBm	Mean =	149.6819	208.8363
	0x 01	Std Dev =	3.093	
	-17 dBm	Mean =	167.642	78.0394
	0x 02	Std Dev =	4.567	
	-14 dBm	Mean =	222.2506	41.3694
	0x 03	Std Dev =	1.0526	
	-11.5 dBm	Mean =	302.4448	148.6817
	0x 04	Std Dev =	1.0754	
2 38.4 k Baud	-19.5 dBm	Mean =	168.09639	57.413704
	0x 01	Std Dev =	3.283256	
	-17 dBm	Mean =	191.75895	79.425085
	0x 02	Std Dev =	0.931551	
	-14 dBm	Mean =	256.65091	324.213592
	0x 03	Std Dev =	1.163718	
	-11.5 dBm	Mean =	328.09406	115.049392
	0x 04	Std Dev =	1.31913	
2 19.2 k Baud	-19.5 dBm	Mean =	190.7525	50.2595
	0x 01	Std Dev =	1.8992	
	-17 dBm	Mean =	239.1081	11.9331
	0x 02	Std Dev =	2.0253	
	-14 dBm	Mean =	292.7798	80.6525
	0x 03	Std Dev =	1.3931	
	-11.5 dBm	Mean =	353.1259	35.7488
	0x 04	Std Dev =	0.98464	

For all the above std. dev. of distance the average = 1.899 inches. The maximum error is 2.668 inches. This is the error in the std. dev. of Q – function and not in the transmission distance. In transmission distance the error will be $2.668/2 = 1.334$ inches. This is because only the distance between mean and monotonically decreasing part of Q – function is included while measuring transmission distance.

All the mean values of distance in table 5.1 are related to baud rate and transmit power. This is three dimensional and we have used the equation mentioned in the next section to do surface fit.

5.3 Surface Fitting

When we consider an arbitrary set of points in the plane, there is not generally an obvious, unique ordering of the points, and this makes surface fitting somewhat difficult [31]. But even here we can use method of least squares to make it a bit easier.

The mean of the distance from curve fitting above is related to channel bandwidth, frequency of operation and SNR apart from transmit power and baud rate. A very important result derived by Shannon [32] is

$$C = B_c \times \log_2\left[1 + \frac{P_r}{(N_0 \times B_c)}\right] \quad \text{.....(5.2)}$$

Here C is the channel capacity in bits per seconds

B_c is the channel bandwidth in Hz

P_r is the received power in watts and

N_0 is the single – sided noise power density in watts / Hz.

$$\therefore \frac{C}{B_c} = \log_2\left[1 + \frac{P_r}{(N_0 \times B_c)}\right]$$

$$\therefore 2^{\frac{C}{B_c}} = 1 + \frac{P_r}{(N_0 \times B_c)}$$

$$\therefore P_r = N_0 \times B_c \times \left(2^{\frac{C}{B_c}} - 1\right)$$

Converting P_r to dBm we get

$$P_r(dBm) = 10 \times \log_{10}\left[1000 \times N_0 \times B_c\left(2^{\frac{C}{B_c}} - 1\right)\right] \quad \text{..... (5.3)}$$

In the wireless channel the received power is calculated using the formula [33] below.

$$P_r(d)[dBm] = P_t[dBm] - PL(d)[dB] \quad \text{.....(5.4)}$$

Here $P_r(d)$ is the received power in dBm at a distance d from transmitter

P_t is the transmit power in dBm and

$PL(d)$ is the path loss in dB at a distance d from transmitter.

For finding the path loss $PL(d)$ we use the Log – Distance Path Loss model [33] where

$$PL[dB] = PL(d_0) + \left\{10 \times n \times \log_{10}\left(\frac{d}{d_0}\right)\right\} \quad \text{.....(5.5)}$$

In this formula $PL[dB]$ is the path loss in dB

$PL(d_0)$ is the path loss at a reference distance d_0

n is the path loss exponent

d is distance in meters at which the path loss is to be calculated and

d_0 is the reference distance also in meters.

Substituting eq. (5.5) in eq. (5.4) we have

$$P_r[dBm] = P_t[dBm] - PL(d_0) - \left\{10 \times n \times \log_{10}\left(\frac{d}{d_0}\right)\right\} \quad \text{.......................................} (5.6)$$

Substituting eq. (5.6) in eq. (5.3)

$$P_t[dBm] - PL(d_0) - \left\{10 \times n \times \log_{10}\left(\frac{d}{d_0}\right)\right\} = 10 \times \log_{10}\left[1000 \times N_0 \times B_c\left(2^{\frac{C}{B_c}} - 1\right)\right]$$

$$\therefore 10 \times n \times \log_{10}\left(\frac{d}{d_0}\right) = P_t - PL(d_0) - 10 \times \log_{10}\left[1000 \times N_0 \times B_c\left(2^{\frac{C}{B_c}} - 1\right)\right]$$

$$\therefore \log_{10}\left(\frac{d}{d_0}\right) = \frac{P_t - PL(d_0) - 10 \times \log_{10}\left[1000 \times N_0 \times B_c\left(2^{\frac{C}{B_c}} - 1\right)\right]}{10 \times n}$$

$$\therefore d = d_0 \times 10^{\frac{P_t - PL(d_0) - 10 \times \log_{10}\left[1000 \times N_0 \times B_c\left(2^{\frac{C}{B_c}} - 1\right)\right]}{10 \times n}} \quad \text{...} (5.7)$$

Thus we relate the transmission distance with transmit power, baud rate, SNR and channel bandwidth. We use this equation in the surface fit using method of least squares.

In the above equation $PL(d_0)$ is calculated using the Friis free space path loss formula [33] given below.

$$PL(d_0) = -10 \times \log_{10}\left[\frac{G_t \times G_r \times \lambda^2}{(4\times\pi)^2 \times {d_0}^2}\right] \quad \text{..... (5.8)}$$

Here G_t and G_r are the antenna gains of transmitter and receiver respectively

λ is the wavelength and

d_0 is the reference distance.

We assume G_t and G_r both to be one i.e. the antenna gains are unity. Thus

$$PL(d_0) = -10 \times \log_{10}\left[\frac{\lambda^2}{(4\times\pi)^2 \times {d_0}^2}\right]$$

Centre frequency of the motes is 433.002 MHz as given in chapter 3 section 3.2.1 from which we get λ and the distance d_0 it is said should be in the far – field region of the transmitting antenna. This far – field region or Fraunhofer region of a transmitting antenna is defined as the region beyond the far – field distance d_f which is related to the antenna dimension and wavelength. It is given as follows [33]

$$d_f = \frac{2 \times D^2}{\lambda} \quad \text{.....(5.9)}$$

Where d_f is the far – field distance

D is the largest physical linear dimension of the antenna and

λ is the wavelength.

The Mica2 motes have a monopole whip antenna which is $\lambda/4$ meters high. Using 433.002 MHz frequency we get $\lambda = 0.6928$ meters = 27.2767 inches and the whip antenna height = 0.1732 meters = 6.8 inches. So D = 0.1732 meters. Putting these values in eq. (5.9) we get

$d_f = 0.086116$ meters = 3.3903 inches

Additional criteria to be in the far – field region are d_f must satisfy

$d_f \rangle\rangle D$ i.e. $d_f \rangle\rangle 6.8$ inches and

$d_f \rangle\rangle \lambda$ i.e. $d_f \rangle\rangle 27.2767$ inches.

So to be in the far – field region we choose our distance d_0 to be

100 inches = 2.54 meters and from the λ value got above we calculate $PL(d_0)$ and use it in eq. (5.7). The N_0 in eq. (5.7) is calculated using $N_0 = k \times T$. k is the Boltzmann constant = $1.3806503 \times 10^{-23}$ and T is the room temperature in degree kelvin = 300^0 k. The channel bandwidth for 433 MHz Mica2 mote, $B_c = 620kHz$ [34]. Using all these values in eq. (5.7) we try to find the best values for n the path loss exponent such that the error, between the distances calculated using eq. (5.7) and the values of distance we got from our experimental data, is minimized. To start with, a value of one is assumed for n. After doing the surface fit we get the following graphs.

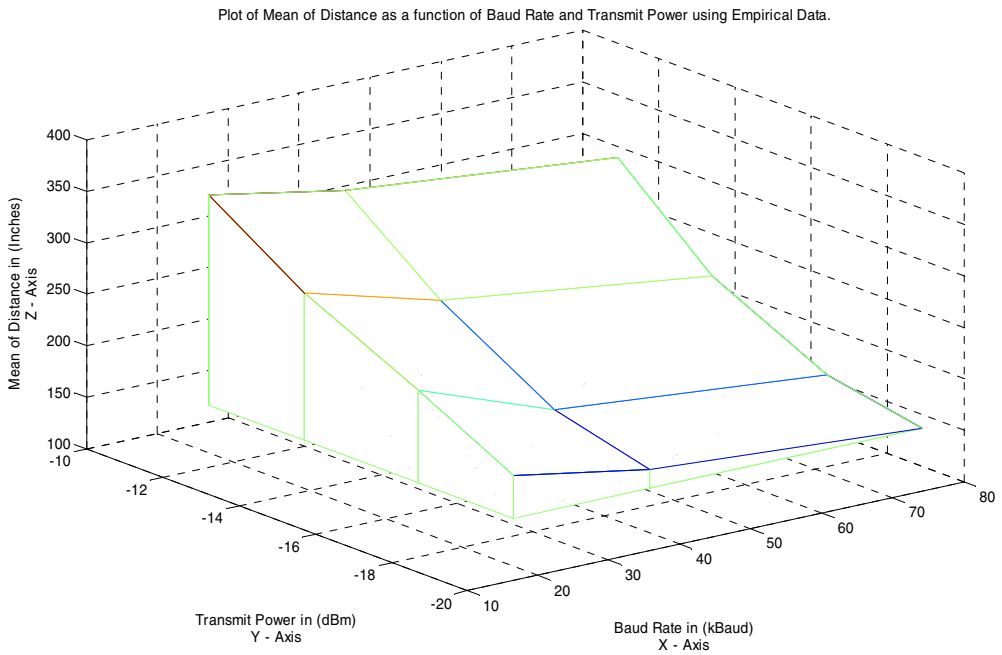

Figure 5.4: Mean of Distance vs Transmit Power and Baud Rate using Empirical Data

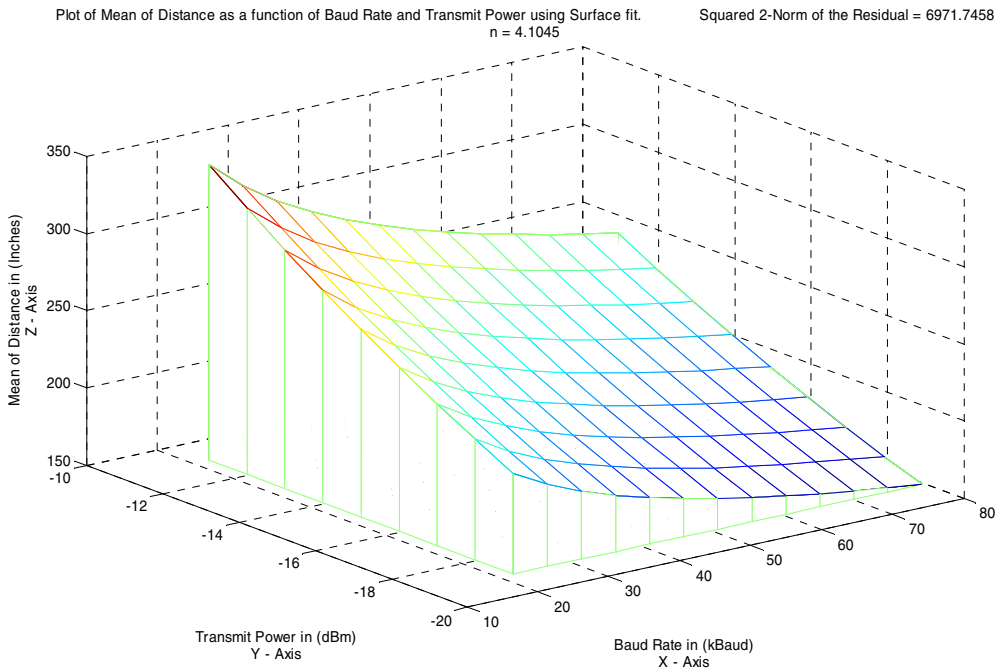

Figure 5.5: Mean of Distance vs Transmit Power and Baud Rate using eq. (5.7) derived from Shannon theorem and Path Loss Model

The optimized value of path loss exponent n got from surface fit is n = 4.01. The following graph compares distances got using both empirical data and eq. (5.7) and also shows the surface got using eq. (5.7).

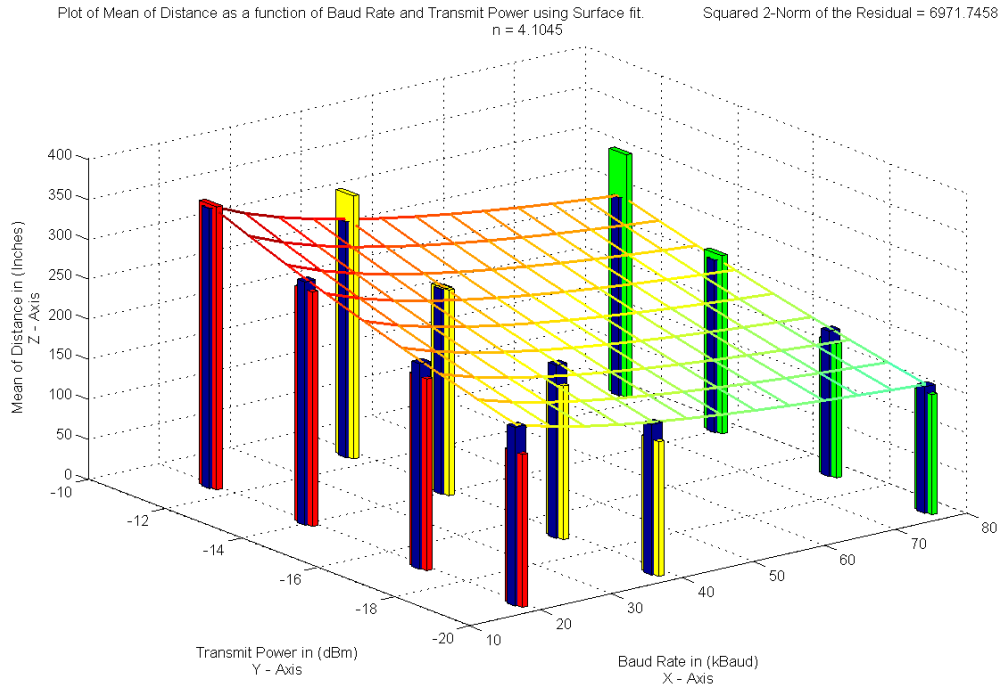

Figure 5.6: Comparison of Mean of Distance vs Transmit Power and Baud Rate using Empirical Data and Surface Fit Data

In the plot above the red, yellow and green bars show the distances got using the empirical data at baud rates of 19.2, 38.4 and 76.8 kBaud respectively. The blue bars show the distance got using eq. (5.7). So the difference between the two is the error. The above plot also shows the surface we get using eq. (5.7).

For comparison purpose the table below shows the empirical distance, distance obtained using eq. (5.7), the absolute error between the two and the percentage error.

Table 5.2 Summary of mean of distance and absolute error

Empirical Data				
		Baud Rate		
		19.2 kbaud	38.4 kbaud	78.6 kbaud
Transmit Power	- 19.5 dBm	190.7526	168.0964	149.6819
	-17 dBm	239.1081	191.7589	167.6420
	-14 dBm	292.7798	256.6509	222.2506
	-11.5 dBm	353.1259	328.0941	302.4448

Shannon Formula Surface Fit - n = 4.01				
		Baud Rate		
		19.2 kbaud	38.4 kbaud	78.6 kbaud
Transmit Power	- 19.5 dBm	223.0249	187.8755	157.8458
	-17 dBm	256.6032	216.1618	181.6109
	-14 dBm	303.6356	255.7817	214.8980
	-11.5 dBm	349.3505	294.2918	247.2528

Absolute Error				
		Baud Rate		
		19.2 kbaud	38.4 kbaud	78.6 kbaud
Transmit Power	- 19.5 dBm	32.2723	19.7791	8.1639
	-17 dBm	17.4952	24.4028	13.9689
	-14 dBm	10.8558	0.8692	7.3526
	-11.5 dBm	3.7754	33.8022	55.1921

Max Error - 55.1921 **Min Error -** 0.8692

Percentage Error				
		Baud Rate		
		19.2 kbaud	38.4 kbaud	78.6 kbaud
Transmit Power	- 19.5 dBm	16.9184	11.7665	5.4542
	-17 dBm	7.3168	12.7258	8.3326
	-14 dBm	3.7078	0.3387	3.3083
	-11.5 dBm	1.0691	10.3026	18.2486

Note: All distances in Inches.

CHAPTER 6

POTENTIAL APPLICATIONS, CONCLUSIONS AND FUTURE WORK

6.1 Potential Applications

In TinyOS 1.1.15 there is a multihop routing protocol MultiHopRouter that uses the MultiHopLEPSM (Multi Hop Link Estimation Parent Selection Module). In this algorithm a parent is chosen from the available neighboring motes by finding the neighbor, to which there is a good quality link. This link quality is the bidirectional quality and is found by keeping a track of the number of packets missed and number of packets received. The parent mote chosen this way is used to forward all the packets sent to the base station. This algorithm doesn't have any facility to adjust the transmit power according to the link quality or the distance to the parent mote.

Localization algorithms like those mentioned in [27, 28] give the distance between the motes in a sensor network. They can be implemented on Mica2 motes and need to be linked to the above mentioned MultiHopLEPSM protocol. Using the distance to the neighboring mote got from the localization algorithm and the equation found in this thesis correct transmit power and baud rate can be set while sending Route packets from MultiHopLEPSM protocol to the neighboring motes. Now a parent mote can be selected which has the best link and the corresponding transmit power and baud rate for the parent is also known. So while forwarding packets correct transmit power and baud rate can be set so as to curb wasteful energy. Thus the study in this thesis makes use of

the data obtained by localization and routing algorithms and aids in energy conservation.

6.2 Conclusions

The key challenge in Wireless Sensor Networks being power conservation, the study in this thesis will support multi – hop routing and clustering algorithms as mentioned in the section above in increasing the power saving in each sensor node. Using Shannon theorem and Log – Distance Path Loss Model the transmission distance is shown to be related to transmit power and baud rate. The extensive empirical data obtained confirms to this relation of transmission distance. The optimized value of the path loss exponent obtained as a result of surface fitting is close to that obtained in other wireless environment. This further bolsters the correctness of the relation. Using this relation along with the knowledge of localization and traffic density between the neighboring mote it will be possible to adjust the transmit power and / or the baud rate to maintain the wireless links between the neighboring motes without loosing the connection and without incurring huge energy costs. All this helps in creating sensor node network in an ad – hoc manner.

Apart from the advantage of power saving, the correct transmit power level will decrease the level of interference in the network since less motes in the surrounding will hear the conversation. There is also an increase in the network capacity as the packet transmissions are confined only to the small local area and other motes out of the area are free to carry out their own transmissions with some other motes within their area.

6.3 Future Work

The experiment done for this thesis was performed in an indoor environment and in Line of Sight. It remains to be seen what will be the effect when performed in an outdoor environment and also for Non Line of Sight condition. Will the relation between transmission distance, transmit power and baud rate still hold and if so what will be the change in the path loss exponent.

Mica2 motes working in the 433 MHz frequency band were used in the experiment. Some new sensors have come in the market that work in the 2.4 GHz frequency band. It will be interesting to see the result on these motes.

While performing the experiment it was found the position of the antenna on Mica2 was very important. Only when the antenna was in vertical position did the mote receive the packets and not when the antenna was fallen down in horizontal position. Care was taken that the antenna was always in vertical position. It looks encouraging to see the results with some different antenna.

APPENDIX A

IMPORTANT LABVIEW BLOCK DIAGRAM

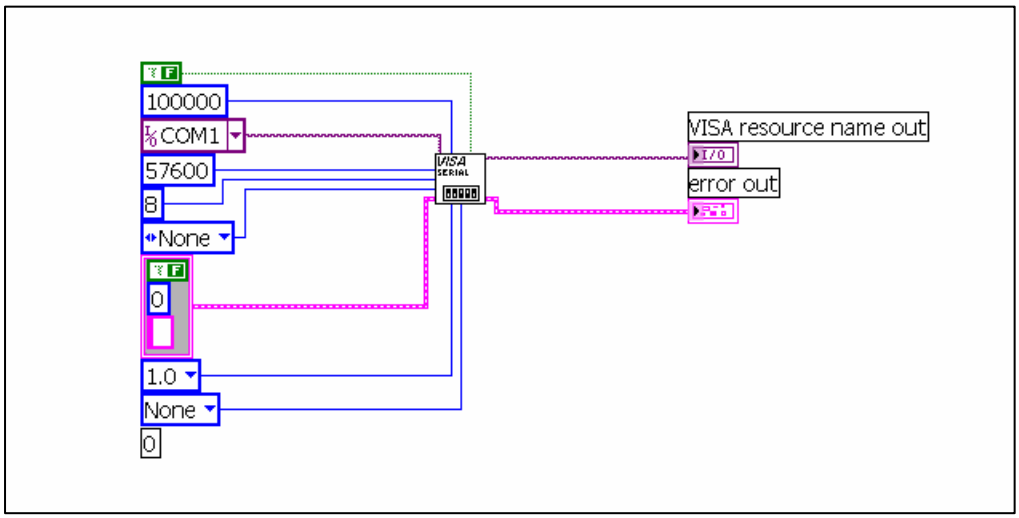

Figure A.1: Block Diagram to Initialize the Serial Port

The Block Diagram in figure A.1 is used to initialize the serial port of the computer. It also inputs the parameters like baud rate, COM port number, data bits, parity, flow control etc that are required during initialization.

Block Diagram in the figure below A.2 captures the packet on the serial port that starts with hexadecimal ‘7E’ and ends with ‘7E’. It outputs all the bytes between these two including the ‘7E’ at both ends. It drops packets that does not confirm to this structure.

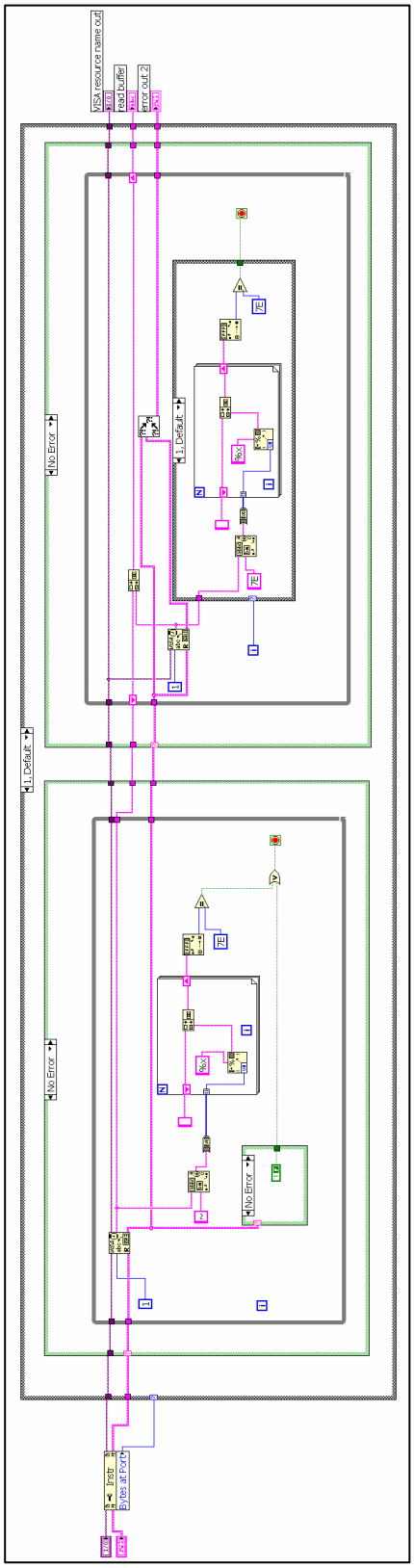

Figure A.2: Block Diagram captures TOSMsg from Serial Port

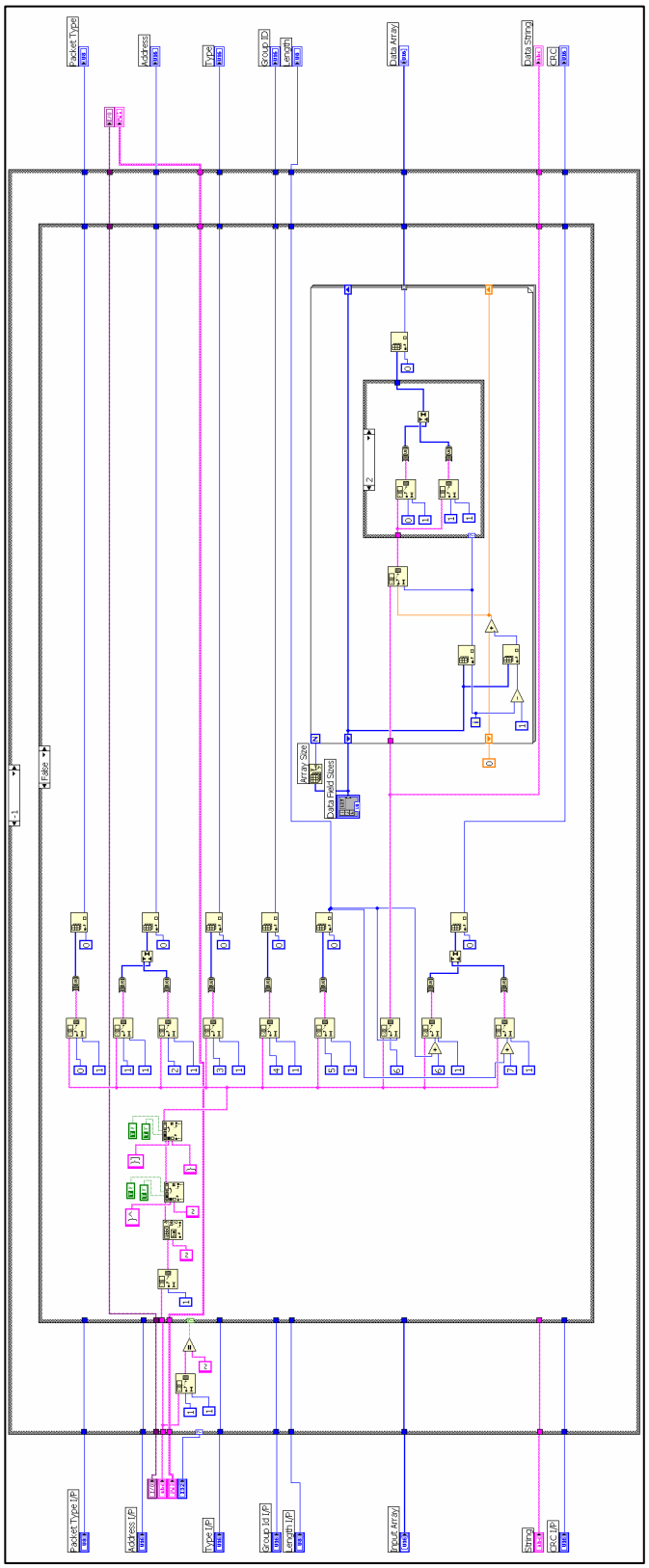

Figure A.3: Block Diagram to decipher the TOSMsg and Data Fields

The program of Block Diagram in figure A.3 takes the packet outputted by the Block Diagram of figure A.2 and deciphers it. It takes an array as input in which the user needs to specify the size in bytes of each field in the data part of TOSMsg. This is required because the program will accumulate the bytes from the data part for each field and convert them to its original value.

The above Block Diagrams are a modified version, of those developed by Mr. Sankar B. Gorthi for his thesis, so that they can work for our experiments. These above programs are intended to be used in some other main program as shown in the Block Diagram below in figure A.4. In this program the blocks with number 1, 2 and 3 in the lower right corner are actually the Block Diagram shown in the above figures. They are used here as sub VI's. This program below is a part of the main program that controls reading of the serial port of the computer. Once it receives the data fields it starts the processing that is required for the data.

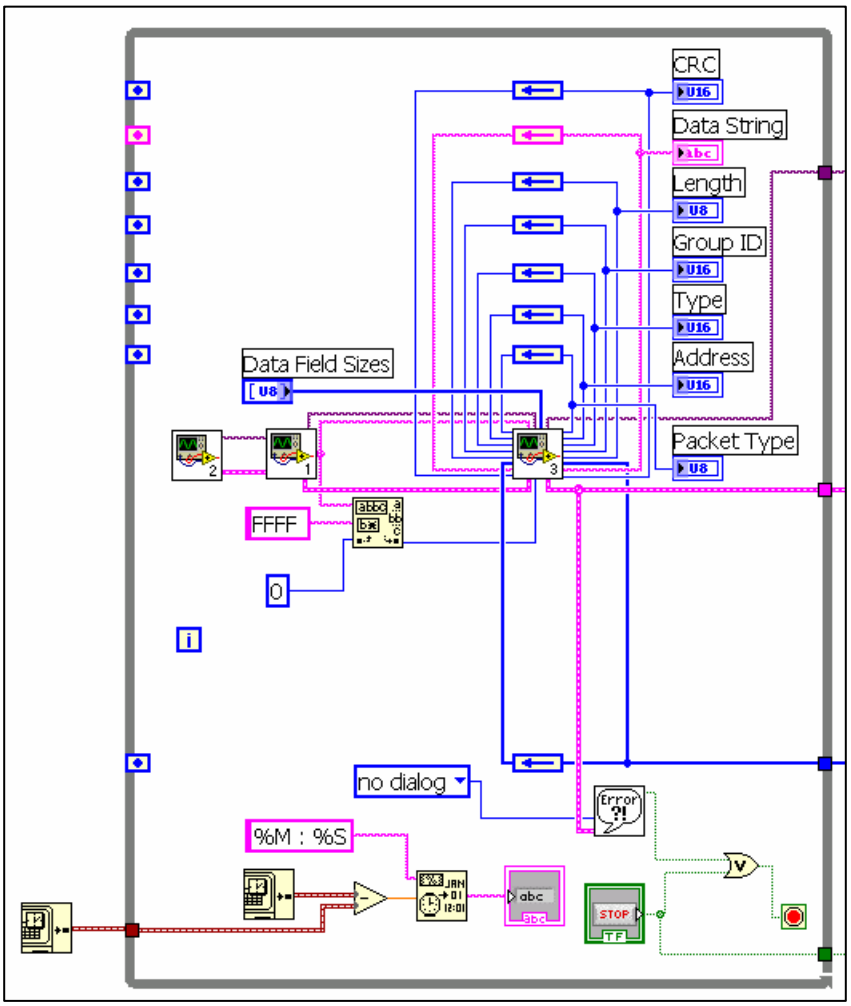

Figure A.4: Part of the Block Diagram that receives user input on data field sizes and outputs the data array

The following figure A.5 shows the Front Panel of the main LabVIEW program.

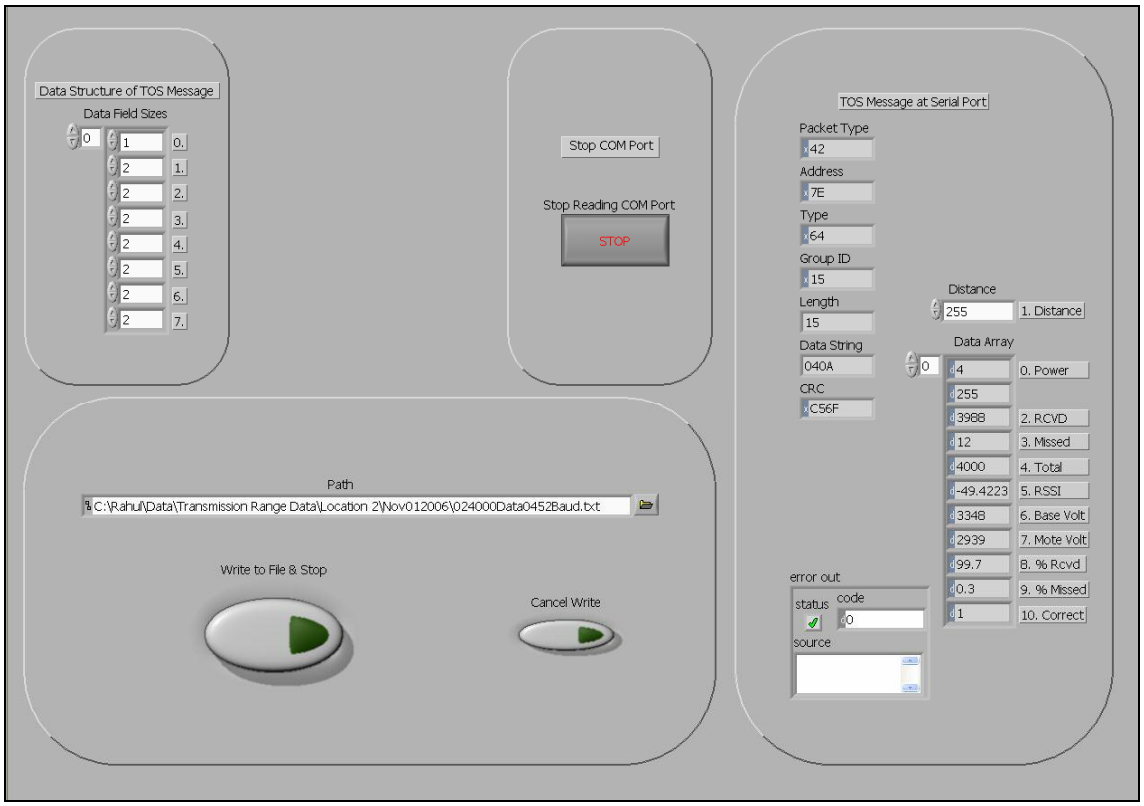

Figure A.5: Front Panel of the main LabVIEW program

REFERENCES

[1] C. Chong and S. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges" Proc. of the IEEE Vol. 91, No. 8, August 2003 Pages: 1247 – 1256.

[2] Alec Lik Chuen Woo, "A Holistic Approach to Multihop Routing in Sensor Networks" December 2004.

[3] http://www.ece.ncsu.edu/wireless/wsn.html

[4] R. Piquepaille and Princeton University, http://blogs.zdnet.com/emergingtech/index.php?cat=4&paged=3

[5] E. Sazonov, K. Janoyan and R. Jha, http://www.intelligent-systems.info/wisan/SMT2004/Sazonov-SMT2004.htm

[6] A. Preece, http://dsonline.computer.org/portal/site/dsonline/.../.../.../...article.xsl

[7] A. Woo, T. Tong and D. Culler, "Taming the Underlining Challenges of Reliable Multihop Routing in Sensor Networks" Proc. of 1st Intl. conf. on Embedded Networked Sensor Systems SenSys November 2003.

[8] C. Perkins and E. Royer, "Ad hoc On – Demand Distance – Vector Routing". Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications, 1999.

[9] D. Braginsky, D. Estrin, "Rumor Routing Algorithm For Sensor Networks" Proc. of 1st Intl. Workshop on WSN & App., September 2002, Pages: 22 – 31.

[10] C. Intanagonwiwat, R Govindan, D Estrin, J Heidemann, F Silva, "Directed Diffusion for Wireless Sensor Networking" IEEE/ACM Trans. on Networking, Vol. 11, No. 1, February 2003 Pages: 2 – 16.

[11] J. Kulik, W. Rabiner and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks" Proc. 5th Annu. ACM/IEEE Int. Conf. Mobile Computing and Networking (MobiCom 1999), Pages: 174 – 185.

[12] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy – Efficient Routing Protocols for Wireless Microsensor Networks" Proc. of 33rd Hawaii Int. Conf. System Sciences (HICSS), Maui, HI, January 2000.

[13] _____, "An Application – Specific Protocol Architecture for Wireless Microsensor Networks" IEEE Trans. Wireless Comm., Vol. 1, No. 4, October 2002 Pages: 660 – 670.

[14] A. Wang, W. Heinzelman, and A. Chandrakasan, "Energy – Scalable Protocols for Battery – Operated Microsensor Networks" Proc. 1999 IEEE Workshop Signal Processing Systems (SiPS 1999), October 1999, Pages: 483–492.

[15] B. Krishnamachari, D. Estrin, S. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks", Proc. of 22nd Intl. Conf. on Distributed Computing Systems, July 2002, Pages: 575 – 578.

[16] J. Polastre, J. Hill and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks" Proc. of 2nd Intl. Conf. on Embedded Networked Sensor Systems (Sensys 2004), November 2004, Pages: 95 – 107.

[17] W. Ye, J. Heidemann, D Estrin, "An Energy Efficient MAC protocol for Wireless Sensor Networks" Proc. IEEE Infocom 2002, Vol. 3, Pages: 1567 – 1576.

[18] Q. Ren, Q. Liang, "A Contention Based Energy – Efficient MAC Protocol for Wireless Sensor Networks", IEEE Wireless Communications and Networking Conf., Vol. 2, April 2006, Pages: 1154 – 1159.

[19] "Wireless Sensor Networks" http://en.wikipedia.org/wiki/Wireless_sensor_networks

[20] J. Polastre, R. Szewczyk, C. Sharp, D. Culler, "The Mote Revolution: Low Power Wireless Sensor Network Devices" webs.cs.berkeley.edu/papers/hotchips-2004-motes.ppt

[21] "TinyOS" http://en.wikipedia.org/wiki/TinyOS.

[22] J. Hill, R. Szewczyk, A. Woo, S Hollar, D Culler, K Pister, "System Architecture Directions for Sensor Networks", Proc. of ASPLOS, ACM – SIGOPS, Vol. 34, Issue 5, December 2000, Pages: 93 – 104.

[23] D. Gay, P. Levis, R. Behren, M. Welsh, E. Brewer, D. Culler, "The nesC Language: A Holistic Approach to Networked Embedded Systems", Proc. of PLDI, ACM – SIGPLAN, Vol. 38, Issue 5, May 2003, Pages: 1 – 11.

[24] "MPR / MIB User's Manual", Rev. A September 2005, Crossbow Technology, http://www.xbow.com/Support/wUserManuals.aspx.

[25] "LabVIEW", http://en.wikipedia.org/wiki/Labview

[26] "CC1000 Single Chip Very Low Power RF Transceiver Datasheet", Chipcon Corporation, http://www.chipcon.com/files/CC1000_Data_Sheet_2_3.pdf

[27] P. Ballal, V. Giordano, P. Dang, S. Gorthi, J. Mireles, F. Lewis, "A LabVIEW based test – bed with off – the – shelf components for research in mobile sensor networks", Proc. of IEEE Intl Symposium on Intelligent Control, October 2006, Pages: 112 – 118.

[28] P. Dang, F. Lewis, D. Popa, "Dynamic Localization of Air – Ground Wireless Sensor Networks", Proc. of IEEE 14th Medi. Conf. on Control and Automation, June 2006, Pages: 1 – 7.

[29] H. Wang, K. Yao, G. Pottie, D. Estrin, "Entropy – based Sensor Selection Heuristic for Target Localization", Proc. of 3rd Intl. Symposium on Info. Processing in Sensor Networks, April 2004, Pages: 36 – 45.

[30] "Wireless Instrumentation: Factors Affecting Transmission Distance", Accutech Tech Note # 215, Accutech Instrumentation Solutions, http://www.savewithaccutech.com/tech-center/technical-articles.asp

[31] P. Lancaster, K. Šalkauskas, "Curve and Surface Fitting An Introduction", Academic Press, 1986.

[32] J. Proakis, "Digital Communication", Fourth Edition, McGraw – Hill, 2001.

[33] T. Rappaport, "Wireless Communications: Principles and Practice", Second Edition, Pearson Education, 2004.

[34] J. Jeong, S. Kim, "DOT3 Radio Stack", www.cs.berkeley.edu/~jaein/presentations/JaeinJeong_ChipconRadioStack.ppt

BIOGRAPHICAL INFORMATION

Rahul P. Sawant received his Bachelors of Engineering degree in Electronics from Mumbai University, India in 1999. He then worked for Electromed Equipment Company and then for Sawant Medical Systems Pvt. Ltd. Besides he has completed a Post Graduate Diploma in Software Technology from National Centre for Software Technology, Mumbai, India in 2003. He started his studies for Masters of Electrical Engineering at University of Texas at Arlington in Fall 2004. Due to his interest in wireless communication he started working on wireless sensor networks in Wireless Communication Networks Lab under Dr. Qilian Liang. He was a Graduate Teaching Assistant for graduate Telecommunication course in UTA. His current research interest is in the field of Wireless Telecommunication.