KMote - Design and Implementation of a Low Cost, Low Power Hardware Platform for Wireless Sensor Networks

by
Naveen Madabhushi

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

May 2007

KMote - Design and Implementation of a Low Cost, Low Power Hardware Platform for Wireless Sensor Networks

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

Naveen Madabhushi

to the

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY,KANPUR

May 2007

CERTIFICATE

It is certified that the work contained in the thesis entitled "*KMote-Design and Implementation of a low cost, low power hardware platform for wireless sensor networks*" by *Naveen Madabhushi* has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

May, 2007

Dr.Kameswari Chebrolu
Department of Electrical & Engineering,
Indian Institute of Technology,
Kanpur-208016.

May, 2007

Dr.Rajat Moona
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur-208016.

Abstract

Wireless Sensor Networks (WSNs), as the name suggests consist of a network of wireless nodes that have the capability to sense a parameter of interest like temperature, humidity, vibration etc. The sensed parameter is often relayed to a base station through the network formed amongst these nodes. The devices used are typically characterized by low cost, low power and are rugged in operation. They are commonly referred to as motes in the WSN domain. The usage of motes was originally intended/ designed for military applications but soon spread to other civilian applications like habitat Monitoring, environment monitoring, monitoring structural health of railway bridges etc. Advances in hardware and wireless technologies have facilitated rapid development of these tiny motes worldwide but their availability in India is not common.

Our work has been motivated by the need to develop such motes in India. This report describes the design and implementation of a mote, which we call as KMote. *The mote integrates programming, computation, communication, and sensing onto a single system and provides an easy user interface for operating and deploying it. This report discusses the design challenges and hurdles involved in translating a paper design to a working prototype. Simplicity in design, low cost, and low power consumption were the primary considerations that influenced the design of* KMote. *This being our first effort in hardware design, we opted for a simplistic design and chose programming complexity in favour of hardware simplicity. We have implemented a prototype that is based on the telos design and integrates an MSP430 microcontroller, CC2420 radio, 8 MB flash and an on board temperature sensor. We have tested and proved the operation of the microcontroller and the radio for transfer of data over a wireless channel in 2.4 Ghz spectrum (ISM band). We have also carried out outdoor experimental tests to determine the RF communication range of the* KMote. *We have obtained a range nearly double that of the TMote, the most popular commercial mote available in the market. At this range we were operating at an average RSSI of -84 dBm and experienced a packet error rate less than 1%. The bill of material cost of* KMote *is Rs 1625 and is less than one fourth of the selling cost of the Tmote (Rs 7740). Also, the administrative overheads involved and large lead time for procurement of Tmote deters widespread use of motes in India. Therefore, the low cost of development of* KMote *makes it a viable option for fabricating such boards in India.*

We believe that the experience and knowledge gained by developing this mote would be beneficial to improve the existing prototype and aid in the easy availability of such motes to research and educational communities in India, thus contributing to the growth of applications in the WSN domain. We also argue in favour of developing such motes in India due to the significant cost benefits offered by such a development.

Acknowledgments

This thesis work has been a culmination of sustained endeavor and cooperation from various people directly or indirectly involved for the past one year. I would take this opportunity to highlight the contribution from prominent people during the course of my thesis work. The mid course corrections provided by them have facilitated the successful timely completion of the thesis.

At the very outset, I would like to express my sincere gratitude to Dr Bhaskaran Raman who provided me the opportunity to work on this topic. His round the clock availability and guidance at various junctures was of immense help towards successful completion of thesis. I am also grateful to Dr Kameswari Chebrolu and Dr Rajat Moona who have been consistently involved in the supervision of the project. Dr Rajat Moona has been of immense help in the hardware design, fault finding and fixing. His involvement till the terminal stage of hardware fabrication and PCB testing coupled with his rich experience and knowledge was considerably instrumental in the speedy execution of the thesis. In addition, he has always been inspirational ever since the inception of the project. I would also like to thank Dr Harish who has spent a considerable amount of his time in teaching me the nuances of PCB and RF circuitry design.

I am grateful to Nilesh Mishra who was ever ready to help me out with procurement of hardware, soldering of components on the board. This work would be incomplete without his involvement. I also thank Dr Parthasarathy Sensarma who provided me unrestricted access to 4i Lab facilities. I would also like thank Kumar and Renu, M Tech students of Electrical Department who took their time off from their work and taught me soldering of PCB boards.

In addition ,Lt Cdr DY Gokhale, Lt Cdr VP Singh, Lt Cdr DK Singh have provided regular support in various activities related to experimental measurements and brainstorming issues of technical relevance to the thesis.

Last but not the least, I would extend my sincere thanks to my wife Preethi and my family without whose support and co-operation, this work would have been incomplete.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The advances in the hardware and wireless technologies have resulted in inexpensive low power communication devices that can be deployed throughout a physical space, providing dense sensing close to physical phenomena, processing and communicating this information, and coordinating actions with other nodes. Such a deployment can be termed as a Wireless Sensor Network (WSN). To realize such a network, we must address a new collection of challenges. The individual devices in a WSN are inherently resource constrained: they have limited processing speed, storage capacity, battery capacity, and communication bandwidth. These devices have substantial processing capability in the aggregate, but not individually. These individual devices are referred to as motes in WSN domain.

1.1 Introduction to Mote

The term mote refers to a tiny particle and these embedded devices for use in a WSN are essentially tiny in size and hence the name. These nodes have processing and computational capability and generally consist of an RF trans-receiver, memory, on board sensors/actuators and a power source. Some of the requirements of a mote for their use in applications are as under:

- The ability to have some amount of on board processing.
- They should be able to communicate over the air and also have some basic networking capability.
- The motes should be alive for extended periods (up to a few months to a year) of deployment. Therefore they should be able to operate at low power.
- They should have sensors/actuators embedded in them so as to interact with the environment in which they are deployed and communicate the sensed parameter to a central location.

To be able to fulfill the basic requirements as mentioned above, a mote should have the following components:

- Microcontroller

- Radio
- An embedded sensor or the ability to connect to sensors/sensor boards.
- On board memory: RAM, Flash.
- Power source

The usage of motes was originally intended for military applications but soon spread to other civilian applications like habitat monitoring [18, 22], environment monitoring [23, 21], volcano monitoring [25], wildlife tracking [15], industrial monitoring [17], structural monitoring [16]. These experimental deployments have enabled researchers and scientists across the globe to study effects/interactions in nature which was earlier not possible with wired technologies due to hurdles like hostile environment, physical hurdles in reaching a place etc.

1.2 Motivation and Problem Statement

There are a number of commercial off-the shelf (COTS) motes available in the market today but none of them are readily and easily available in India. There has been a similar development at IIT Delhi [3], wherein they developed *Rete*, a mote which operates at 433 Mhz. However, these motes have their sales division in USA and are not readily available in India. Moreover, the COTS motes have to be imported and not all educational institutions in the country have administrative procedures in place that enable them to procure these devices easily. It is therefore important that such devices are easily and readily available to the educational and research communities in India. Each of these motes costs a few thousands of rupees and thus are still costly enough to deter widespread use in educational and research institutes. Hence, the cost of these devices should be low enough to encourage their widespread use. With an increase in awareness about these devices, it is realistic to assume that new applications are developed which cater to the problems specific to India.

We therefore define our problem statement as: to design and implement a low cost, low power hardware platform for use in wireless sensor networks. Therefore, the main goals that we seek to address in this work are the following:

- Design of a low cost, low power hardware platform for use in sensor networks.
- Build a simple working prototype that can be improved upon. The prototype though simple should be able to deliver all the functionalities possible with commercially available state of the art motes.
- The motes developed should be easily available for research and educational institutions.

We have designed and implemented *KMote* which integrates an MSP430 micro-controller, a CC2420 radio chip which operates in the Industrial, Scientific and Medical (ISM) band (2.4 Ghz). An 8 Mb flash for data logging and a temperature sensor have also been embedded into the platform. We have integrated *KMote* with an open source embedded operating system *TinyOS* so that it can be readily programmed for custom applications. The development cost of *KMote* is Rs 1625 and is less than one fourth the cost of the most popular commercially available mote, *Tmote*, which costs Rs 7740. We have tested the building blocks of the mote: microcontroller and radio circuitry for their correct operation. We have also carried out range measurements and have observed ranges nearly twice that of Tmote. The mote is powered by two AA batteries or 3.0 volts power source and consumes about 20 mA during transmission.

1.3 Thesis Organization

The rest of the report is organized as follows. Chapter 2 discusses the evolution of various motes and comparison of commercially available motes. Chapter 3 discusses the design considerations and implementation of *KMote*. In this chapter we discuss the hardware choices available and the reasons for choosing the hardware used in *KMote*. We also discuss the architecture of *KMote* and design issues while translating the paper design to a working model in the section on PCB design. We also discuss our design experiences during the fabrication and integration of the mote with *TinyOS* in this chapter. Chapter 4 discusses the testing methodology of *KMote* and the results obtained during the range and power consumption tests. We finally summarize our work in Chapter 5 and discuss the future scope in Chapter 6. The appendices at the end of the report give detailed schematics of the mote, bill of material required, procedure for integration of mote with *TinyOS* and the procedure to connect an external antenna to the mote to achieve higher ranges.

Chapter 2

Background and Related Work

The usage of motes was conceived and implemented by the Smart Dust [24] project. The Smart Dust project explored the possibility of using motes to form a massive distributed network. The main goal of the project was to explore micro fabrication technology's limitations and study the feasibility of such a platform. Because of the mote's discrete size, substantial functionality, connectivity, and anticipated low cost, Smart Dust was envisaged to facilitate innovative methods of interacting with the environment, and provide more information from physical regions less intrusively. The intended applications for smart dust were:

- Deploy sensor networks rapidly by unmanned aerial vehicles or artillery in battlefield scenarios and track enemy movements.
- Tracking the movements of birds, small animals, and insects and understand their behavioral patterns
- Monitor health of rotating machinery and understand the reasons for high-cycle fatigue.

There have been a number of platforms which were developed on the lines of smart dust motes. The initial devices incorporated small microcontrollers (8 bit, 4 KB flash) and a simple radio (4 Kbps data rate) and their life time was up to a maximum of 2 years. Table 2.1 taken from [19] shows the evolution of mote from 1998. The figure also depicts the relative capabilities of the mote as it evolved.

As can be seen from the table, the main focus in the design of the motes is low power of operation. Almost all the microcontrollers used in various designs operate at low voltages of 2.7 volts to 3.6 volts. Another important factor for the choice of microcontroller is the current drawn during its sleep and wake up time. Low sleep current and low wake up times minimise the duty cycle of a mote and therefore extends the life of a mote and consequent network life time. Telos design uses MSP430 microcontroller which has lowest sleep and wake times as compared to the other microcontrollers operating at low voltages. On-chip RAM also plays a vital role in the processing capability of the mote and MSP430 offers the maximum on-chip RAM in comparison. Another salient factor in the mote's evolution is their usage of radio chips for communication. With the introduction of

Mote Type (Year)	**WeC 1998**	**Rene 1999**	**Rene2 2000**	**Dot 2000**	**Mica 2001**	**Mica2Dot 2002**	**Mica2 2002**	**Telos 2004**
Processor	AT90LS8535		ATMega163		ATMega128			MSP430
Radio Chip	TR1000					CC1000		CC2420
Flash	32 KB				512 KB			128 KB
Sensors	No on-board sensors			Yes	No on-board sensors			Yes
Frequency of Operation	916.5 Mhz					868 Mhz		2.4 Ghz
Transmit Power	0 dBm							
Max Data Rate	10 Kbps				40 Kbps	38.4 Kbps		250 Kbps
Power Consumption	36 mW					42 mW		35 mW
Sleep Power (μW)	45				75			6
wakeup Time (μS)	1000		36		180			6
Interface	IEEE 1284 and RS232							USB

Table 2.1. Evolution of Mote.(Source - Telos:Enabling ultra-low power wireless research,IPSN '05)

802.15.4 standard for wireless sensor networks, the motes too started incorporating chips that use these standards for interoperability of platforms.

A comparison of state-of-the art popular motes is shown in Table 2.2. As seen from the table, Tmote offers the best functionalities compared to the other motes. However, the cost of these motes is quite expensive and are not readily available for educational/research institutions in India. These motes need to be imported and not all educational institutes have administrative procedures in place for importing them. Moreover, the lead time involved in procurement is in the order of a few months and such a large time deters their widespread use. There has been a similar development at Technology business incubation unit of IIT Delhi called *Rete* [3]. Rete is a mote that is developed as part of CpiderNet, a complete sensor network infrastructure consisting of motes, self configuring networking stack and software front-end for data-analysis and network management. Although, Rete has been developed at IIT Delhi, these motes are marketed and sold by a US based company *ElfSys*. It is therefore evident that motes that cater to Indian requirements are not easily and readily available for educational/research institutions. Rete operates at 433 Mhz which is not unlicensed in India. The usage of this frequency is restricted to demonstration of equipment/ for amateur radioists but is not permitted for commercial use.

The design and development of *KMote* is based on Telos design. The cost of research prototype of *KMote* is Rs 1625 which is significantly lower than the cost of *Tmote* which is priced at Rs 7740. Moreover, procurement of these motes incur and additional shipping charges which are the order of the cost of a couple of *KMotes*. We therefore, believe that easy availability of *KMote* coupled with low cost would facilitate growth of WSN applications that cater to specific problems of India.

	MicaZ	**Tmote**	**Tiny Node**
Processor	AT Mega 128	MSP430	MSP430
Radio Chip	CC2420	CC2420	XE 1205
Flash	512KB	8MB	4MB
Sensors	No on board sensors	Humidity Light	Light Temperature
Frequency of Operation	2.4 Ghz	2.4 Ghz	868-870 MHz
Transmit Power	0 dBm	0 dBm	0 to +12 dBm
Max Data Rate	250 Kbps	250 Kbps	152.3 Kbps
Max Range (outdoors)	upto 70m	upto 150m	200m @ 76.8 Kbps
Power Consumption (Max)	28 mA	21mA @ 0dBm	25mA @0dBm 62mA @ +12 dBm
Interface	USB/Serial	USB	Serial
Operating System	TinyOS	TinyOS	TinyOS
Cost (USD)	125	130 (without sensors) 180(with sensors)	180

Table 2.2. Comparison of Popular Motes

Chapter 3

Design Considerations and Implementation

A good design forms the building block for any successful hardware implementation. It is also the most important aspect and involves multiple revisions and evaluations to identify and correct any errors. When we started off with the design of *KMote*, we decided on the following as general guidelines of our design:

- The design should have all the characteristics of a mote: low cost, low power consuming, flexible platform.
- Keep the first design as simple as possible and opt for programming complexity in favour of simplistic hardware. This being our first effort in hardware design, a simpler design is a good starting point to avoid any challenges arising from inefficient hardware design.
- The design should be such that it should have the functionality of the state of the art motes available commercially.

In this chapter we present the design choices involved and considerations thereof. We explain the choice of hardware for our platform and the architecture chosen to implement this platform. We also discuss the design challenges involved in PCB design and explain the choices made. We finally explain the process involved in translating the software design to fabrication of PCB and its integration with TinyOS, the embedded Operating System.

3.1 Hardware Design

In this section we discuss the hardware chosen for KMote and mention the modifications and enhancements carried out on an existing embedded platform design. With the design guidelines in place, the next step was identification of hardware to suit our requirements. We chose to stick to the hardware used in *telosb* [7] as the advantages offered are dual:

- The microcontroller (MSP430) and radio (CC2420) used in that design are readily and easily available.
- The embedded operating system TinyOS has inbuilt support for these chips. This obviated the need for writing new drivers to integrate the platform with TinyOS.

Property	**CC2420**	**XE 1205**
Frequency of Operation	2.4 Ghz	868-870 MHz
Transmit Power	0 dBm	0 to +12 dBm
Receiver Sensitivity	-95 dBm	-121 dBm @ 1.2 Kbps -101 dBm @ 152.3 Kbps
Max Data Rate	250 Kbps	152.3 Kbps
Max Range	upto 150m	200m @ 76.8 Kbps
Current Consumption	21mA @ 0dBm	25mA @0dBm 62mA @ +12 dBm

Table 3.1. Comparison of CC2420 and XE 1205 Radio chips

For the radio chip, we also contemplated the design alternative of using the XE1205 radio chip in place of CC2420. The comparison of these two radios are as shown in Table 3.1. The data is taken from their respective datasheets [1, 14]. The ranges specified in the table are however taken from Tmote and Tiny Node datasheets [12, 10]. Although XE1205 offers significant advantages in terms of better transmit power and receiver sensitivity and hence better range, the main hurdle in choice of that radio was its frequency of transmission. The radio operates in the range of 868-870 MHz and this spectrum is licensed in India. We had started our work with the aim of easy availability of these motes in India for educational and research purposes and hence the usage of a radio which operates in a licensed spectrum would not be an appropriate choice.

Furthermore, it is possible to achieve better ranges using Tmotes with usage of external directional antennas. During our work [20], we carried out in-depth range measurements of Tmotes using different types of external antennas. The term range refers to the physical distance between two motes for 0% packet error and an average Received Signal Strength Indicator (RSSI) of -84 dBm. The observed ranges for tmote sky in a line of sight environment on a road with some foliage on either side of the road/ airstrip is shown in Table 3.2. Since these ranges are good for one hop distance in a practical deployment, we decided to continue with CC2420 in the prototype.

Having decided on the core components of the mote - microcontroller and radio, we looked at the designs of two state of the art commercially available motes - Tiny Node and Tmote. We then chose Tmote that is based in telosb design. Another factor for consideration of Tmote was that we had an experience of working with them [20]. We decided to modify the design of Tmote and come up with a design that is simple enough but also provides the basic functionality of a mote.

The modifications made to Tmote design are enumerated below. A complete and in depth explanation of the design is explained later in section 3.3. Due to

Tx-Rx Dist.(m)	**Average Pkt Error (%) (Std. Dev.)**	**Average RSSI (*dBm*) (Std. Dev.)**
Internal Antenna at Receiver		
Internal-60m	0.18 (1.03)	-81.11 (2.97)
Internal-75m	**1.37 (4.34)**	**-83.74 (3.61)**
Omni-60m	0 (0)	-81.92 (0.49)
Omni-75m	**0 (0)**	**-80.64 (2.47)**
Omni-90m	35.92 (33.42)	-94.91 (1.6)
Sector-210m	**0 (0)**	**-81.92 (0.49)**
Sector-310m	1.02 (4.3)	-91.85 (0.81)
Sector-400m	0.62 (2.24)	-92.33 (1.03)
Sector-500m	0 (0)	-90.12 (0.5)
Grid-90m	0 (0)	-75.35 (1.36)
Grid-210m	0.03 (0.18)	-75.825 (2.37)
Grid-300m	0 (0)	-80.42 (1)
Grid-400m	0 (0)	-82.21 (0.9)
Grid-500m	**0 (0)**	**-85.67 (0.94)**
Omni Antenna at the Receiver		
Omni-90m	**0.04 (0.33)**	**-80.92 (0.88)**
Omni-150m	7.63 (12.46)	-90.86 (0.64)
Sector-500m	**0.13 (0.68)**	**-80.92 (0.88)**
Sector-600m	0.07 (0.25)	-89.48 (0.35)
Sector-700m	0.5 (1.05)	-91.22 (0.34)
Sector-800m	3.42 (4.83)	-91.58 (0.41)
Sector-30m	13.01 (14.37)	-90.01 (3.91)
Grid-500m	0.12 (0.49)	-75.25 (0.07)
Grid-600m	0.07 (0.25)	-79.85 (0.24)
Grid-700m	0.15 (0.61)	-82.07 (0.2)
Grid-800m	**0.13 (0.39)**	**-85.76 (0.31)**

Table 3.2. Range measurements on Narrow road / airstrip (Line-of-sight environment)

the modifications mentioned below, the form factor of the *KMote* increased as compared to that of Tmote.

(a) Provision of serial interface in lieu of USB.
(b) Provision of external power supply adapter.
(c) on-board temperature sensor.
(d) LCD panel interface.
(e) Increase in unused GPIO pins.

3.2 Architecture of *KMote*

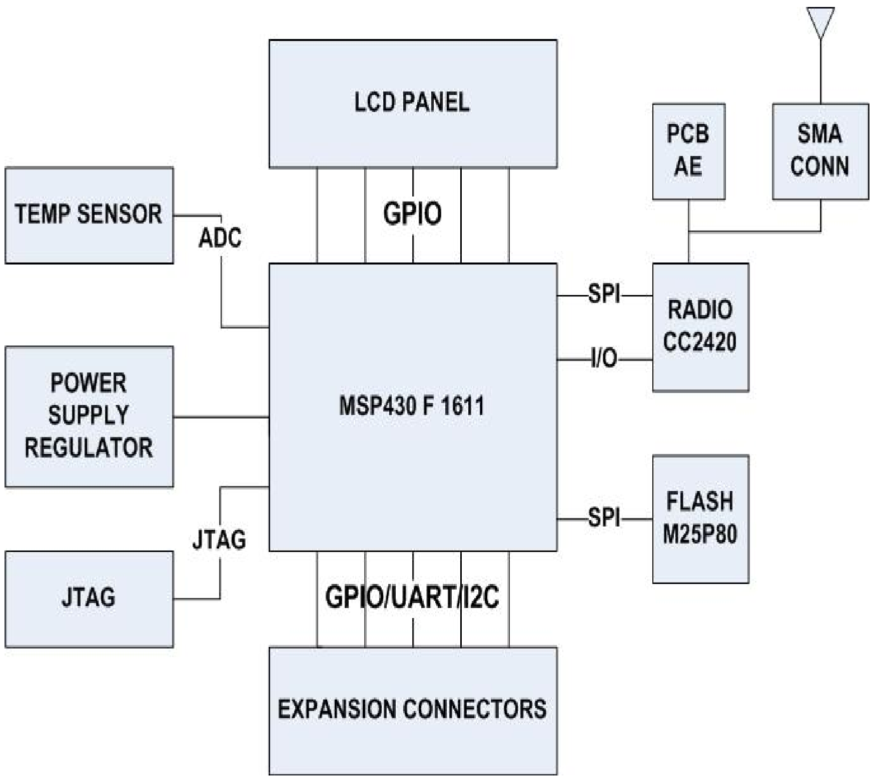

Figure 3.1. Architecture of *KMote*

The architecture of *KMote* after incorporation of features as mentioned above is shown in Figure 3.1. The microprocessor being the building block becomes the core of the platform and the functionality of the mote is built around it. Radio, the other building block of the mote, is interfaced to it using the SPI bus. We chose SPI as full-duplex capability of this bus makes it simple and efficient for an application that has single master and one/two slaves. The other choice was usage of I^2C bus. SPI bus supports data rates upto 20 Mbps while the common data rates of I^2C bus are 100/400 Kbps. This aspect made us choose SPI as the communication bus. In addition to the SPI bus, the radio needs an interface for Digital I/O, Clear Channel Assessment (CCA), Start Frame Delimiter (SFD) pins. I/O pins of microcontroller are used for this purpose.

In order to radiate electromagnetic energy, the mote needs an antenna. We chose an Inverted F Antenna (IFA) for this. This antenna is recommended by Chipcon [4] for usage with CC2420. This antenna has an omni directional radiation pattern and is therefore useful in typical applications where motes are used. To achieve higher ranges, a provision for connecting external antenna is required. This is achieved by providing a Sub-Miniature Version A (SMA) connector which can be connected to a 50 Ω antenna. The procedure for connecting an external antenna to the mote is described in Appendix C.

To enable data logging and storage, the microcontroller is interfaced to a flash STM25P80. This chip has a memory of 8 Mb, operates at low voltages of 2.7 to 3.6 volts and has an SPI interface [8]. We chose this for low operating voltages and good storage capacity. The flash is interfaced through SPI bus and these communication lines are shared with the radio. Hence, careful bus arbitration is required while using any one of them.

In many teaching and research applications, it is useful to have an embedded sensor on a mote. Therefore we interfaced a temperature sensor with the ADC pins of the microcontroller. We have also provided a JTAG interface to enable testing and debugging of the platform.

Another important requirement of a mote is its ability to interface peripherals. Universal Asynchronous Receiver/Transmitter (USART), Inter-Integrated Circuit (I2C) bus and General Purpose Input Output (GPIO) pins of the microcontroller are used for this purpose. These pins are brought out in the form of expansion connectors and peripherals can be connected to them.

In general, for teaching and research purposes, it is useful to have an on-board LCD that aids in better understanding of the functioning of the mote, Also, during our work [20], we felt the need for a better debugging interface than the LEDs provided on Tmote. We wanted to be able to printout some debugging messages which are more useful than just using three LEDs for diagnostic purposes. We therefore included an interface with LCD panel in our design. In addition to debugging, the presence of an LCD panel can also be used for instructional purposes. This panel is interfaced with GPIO pins of the microcontroller.

3.3 PCB Design Issues

After deciding the necessary modifications in telosb design, we incorporated them in our design. Incorporating the modifications and keeping the design as simplistic as possible was challenging and we had to give in for some programming complexity in favour of hardware simplicity. The design challenges we encountered and consequent decisions are enumerated in subsequent paragraphs.

(a) **Provision of a serial interface in lieu of USB**. This interface is primarily for programming the mote from a PC/Laptop and also for transfer of data from a mote to PC. Provision of a serial interface is in accordance with our goal of reducing circuit complexity of the PCB. Also, from the experiences of researchers in [25], we felt that, a serial interface is a lot more advantageous when the motes are interfaced with external devices like GPS which have serial ports as their primary mode of communication.

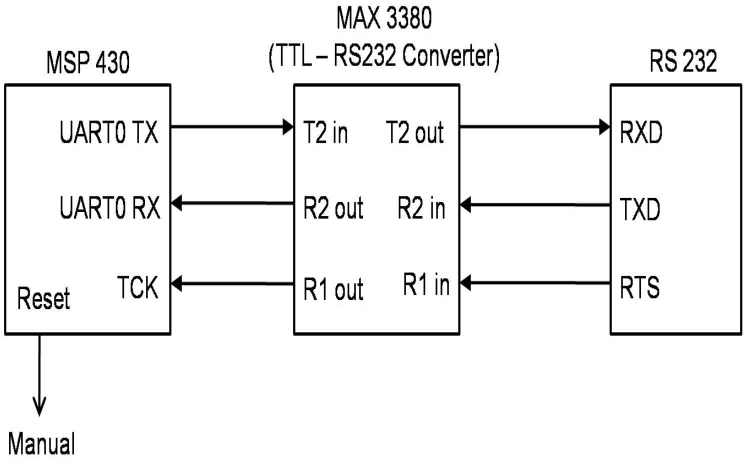

Figure 3.2. Serial Interface for KMote

A simplified block diagram of connections between the serial port and MSP430 in *KMote* is shown in Figure 3.2. The microcontroller has UART interface that supports RS 232 protocol. We have used MAX3380 [5] which has two trans-receivers and converts voltages from TTL (used by MSP430) to RS 232 (used by Serial Port). In addition to the connections shown in the figure between the chips and serial connector, MSP 430 needs a signal to reset the device before it is put into programming mode of operation. It is desirable to put the processor in programming mode in somewhat an automated manner from the PC. This can be achieved with the use of modem control signals on RS232. However these signals use ±15 V signalling. MAX 3380 has only two trans-receivers and both of the receivers are used to drive UART0X and TCK of MSP430 thereby leaving no spare pin to convert the signals to TTL level and to drive reset for MSP430. There were two choices available for us to reset MSP430: (i) To use another MAX 3380 chip and wire the reset pin of MSP430 to one of the receivers (ii) To use only one MAX 3380 chip and use a reset switch of the mote to reset the microcontroller. We chose the latter as we wanted to use minimum components on the PCB thereby simplifying the design.

(b) **Provision of external power supply adapter**. Since the USB interface has been done away with, there was a requirement for external power source to power the mote. This power supply is the primary power supply for the mote with batteries as the secondary mode of power source. Consequently, we designed a voltage regulation circuit to regulate the external power supply. We used the MCP1700-3302 [13] voltage regulator chip for this purpose. The chip gives a regulated power supply of 3.3 volts for an input power supply of upto 6 volts.

(c) **on-board temperature sensor**. The Tmote design has on-board humidity and pressure sensors. These sensors were interfaced with GPIO pins of microcontroller. We felt that it would be convenient if we brought these GPIO pins out as headers thereby providing greater flexibility in interfacing of different kinds of peripherals with the mote. Therefore, we removed these sensors from our design and interfaced an on-board sensor to demonstrate the proof of concept of a sensing application and brought out the remaining GPIO pins to the external headers. Removal of these sensors also helped in bringing down the cost of the mote. We used the Analog Devices TMP36FSZ sensor which has an operating range of -40 to +125°C [9]. The sensor operates at $V_{cc} = 2.7V$ and this supply was provided from microcontroller itself using a GPIO pin.

(d) <u>**LCD panel interface**</u>. A GPIO based 2x16 character LCD panel interface is provided to aid programmers in debugging the applications on the mote. This was a major drawback in Tmote wherein the programmer had access only to 3 on-board LEDs for debugging and this proved inadequate for quick debugging. We were faced two alternatives for choice of LCD Panel: (i) LCD Panel without on-board controller and (ii) LCD panel with on-board controller. MSP430 F1611 does not provide any LCD controller and so we chose an LCD panel with an integrated controller.

(e) <u>**Increase in unused GPIO pins**</u>. As discussed in (c) above, the number of unused GPIO pins in *KMote* have been increased to 26. This would allow a larger number of GPIO based peripherals to be interfaced with *KMote*. With the addition of LCD panel and external power supply adapters, there has been an increase in the form factor of the board as compared to that of Tmote. With an increase in form factor, we were also able to provide an additional mounting hole to improve mechanical the strength of the mote.

3.4 Design of PCB in Protel DXP 2004

This section discusses the our design experience during designing the PCB in Protel DXP. We feel these experiences will be useful for future developers and so discuss them in detail.

The schematics and PCB of *KMote* were designed in Protel DXP 2004. Drawing of the schematic was fairly easy but it took us quite some time to translate the schematic to a PCB design. Protel is a very good software but has a steep learning curve. It took us about a month to explore all the features and capabilities of the software. Although the available documentation is exhaustive, it is poorly organised. We had to design footprints for most of the components in use as the footprints made available by the respective manufacturers were not compatible with Protel. Since we had decided upon the choice of components in advance and ordered them as well, it was easy to design footprints. In retrospect, this was a wise decision as choosing a component and designing a footprint for it is the not the correct way. This is because, there were instances where we chose a certain component but the lead time for procurement was as large as 2-3 months that we had to discard that and look for an alternative. This is a factor specific to India, as we ordered our components from suppliers based in USA. This is because it was easier to locate and order from these suppliers on internet. However, during the course of our work, we realised that many of components which were ordered from USA are also available through suppliers based in India. Integration of footprints also took a lot of time as this was the first experience for us in using Electronic Design Automation (EDA) software.

Just when we thought that it would be easy to design PCB once the footprints are integrated, we realised that the job was not that simple. A major challenge was the decision on the track thickness for connecting the output of Radio to the on-board antenna. The width of the track determines the characteristic impedance, Z_0 of the transmission line connecting the power output from radio to the feed of the antenna. The footprint of the antenna was taken from the Chipcon Application Note [4]. Although there exists detailed formulae to compute the track width, we chose a width of 0.4 mm as per the design of Tmote. There was also a requirement of placing a large number of grounded vias around the path of radiation to minimise RF radiation due to the leakage of radiated power to other parts of the board.

The choice of minimum track width and minimum diameter of plated through holes was another tricky issue. Routing of wires between the components on the board becomes a lot easier if these dimensions are small. However, manufacturer cannot fabricate a PCB below certain minimum dimensions. So, we contacted the manufacturer and confirmed his limitations to be 6 mils and 12 mils as minimum track width and minimum diameter of via respectively. We chose a minimum dimension of 6 mils for track width and 20 mils for via diameter and designed the board accordingly. We chose the clearance width to be 0.4 mm between tracks and vias to prevent any accidental short circuiting of the tracks while soldering the components.

The choice of discrete components for radio circuit was as per the guidelines mentioned in the datasheet [1] and most of them were components with footprint 0402 . However, the solder side of the board where microcontroller was placed, did not have any restriction on the size of the components and we chose components with footprint 0805. All the components chosen were Restriction of Hazardous Substances (RoHS) complaint and lead free. The schematics of the top and bottom layers of *KMote* are present at Appendix A and Appendix B respectively.

The final PCB was a 4 layered one with dimensions of 2 inch by 3 inches. The entire process of generating a gerber file for manufacture of PCBs from the conceptualisation took us about 5 months. This large time frame was mainly due to our lack of experience in PCB design and also due to the number of revisions which went into each stage of the design. It is pertinent to mention that, as an aside to this process, we experimented with designing a simple accelerometer board for use with Tmotes for the BriMon (reference) application. The experience gained in designing and fabrication of these accelerometer boards was immensely helpful in our design of *KMote* which highlighted the role of experience in such a process.

3.5 Fabrication and Assembly of PCB

The PCB material specifications are the following:

(a) 4 Layer PCB 1.0 mm FR4. The total thickness of the board is 1.0 mm and the material is Flame Retardant woven glass reinforced epoxy resin.

(b) Layer Spacing:

  – 1-2: 0.25 mm
  – 2-3: 0.4 mm
  – 3-4: 0.25 mm

(c) 1 oz copper pour, solder plated finish

There were a number of revisions of design due to the interaction with the manufacturer. Ultimately, the fabricated PCBs were received by us only in the third week of March 07.

The next major and very important task in the development cycle of an embedded system is assembly of components on the fabricated PCB. This is not an easy task as there were specialised chips like CC2420 that needed a lot of care while soldering. Also, the form factor of the components used were very small. We were novices vis a vis soldering and manual soldering of the components on the board was strictly not an option if we wanted the board to work. There were however two options: (i) Assemble the board at IIT Kanpur itself by making use of soldering facilities in the 4i lab. (ii) Offload the soldering process to a professional firm identified in Bangalore. We identified the firm in Bangalore as the firms we contacted in Kanpur/Delhi were not eager to take up orders in small quantities and required a minimum order of at least 100 or more. We had sufficient components to assemble five boards and so decided to solder one board in IIT Kanpur and offload the rest to the firm in Bangalore.

The process of soldering the board in 4i lab was very helpful in understanding the circuit better and also in getting experience in soldering these tiny components. The soldering experience came in handy while debugging the hardware for errors (this will be explained in detail in Chapter 4). We carried out soldering of these components in three phases. In the first phase, we used the IR soldering machine to solder MSP430. This was a first time for us to use the machine and the process though intimidating initially, was fairly simple and easy. The machine uses vacuum based tip to pick up the component and with a magnification of 10x available for the user, it was very easy to align the component on to the pads on the PCB. The PCB was tinned during the fabrication process and so we did not tin the boards further. Once the chip was aligned with the pads, it was placed on the pads and Infra Red radiation was used to heat the chip evenly and solder the pins to its pads.

The next phase involved correct placement and alignment of remaining components on their respective solder pads. We then used the LPKF reflow soldering machine to evenly heat the board and solder the components. The process is very simple and convenient and the only time consuming process in this type of soldering is the alignment and placement of components on their respective solder pads. With this method, one can solder upto 10 boards of our PCB size at one go. We however, soldered only one board. After soldering the solder side of the board, we used reflow process to solder the component side as well. We manually soldered expansion headers, switches and power supply jack later on completion of soldering the components.

However, the soldered board did not work as there were a number of dry solder points on the PCB. This is because we soldered the components directly to the tinned pads without tinning the PCB. We later learned that this is not a good practice and boards have to be tinned before soldering them.

The other four boards along with the components were soldered at Peninsula Electronics, Bangalore. Five PCBs were all assembled and ready for testing by the second week of April. There was no other way to test the mote than to integrate it with TinyOS, load an application and check for correct functionality of the program. Once the boards were powered up and tested through JTAG for correct detection of MSP430, we went ahead with the process of integration of *KMote* with TinyOS. A detailed explanation of testing the mote through JTAG is explained in Chapter 4.

3.6 Integration of *KMote* with *TinyOS*

An embedded hardware platform is only useful if it can be programmed to run a desired application. To achieve this, the platform needs to be integrated with an embedded operating system. There are many embedded operating systems available but we chose TinyOS as the operating system. TinyOS is an open source OS developed for embedded systems at U.C.Berkeley and supports both MSP430 and CC2420. Another major factor for choice of TinyOS as the operating system is the fact that we were experienced in its usage and quite comfortable with its structure and design.

After successful completion of fabrication and assembly of *KMote*, it was necessary to integrate it with TinyOS. We chose to integrate it with the current version of TinyOS, i.e. TinyOS 2.x. The steps involved in integration are fairly simple and straightforward. As mentioned earlier, TinyOS supports multiple platforms (e.g. MicaZ, Mica, Tmote, TinyNode etc) and the structure of TinyOS is such that it reuses as much code as possible between platforms. The platforms support either or all the chips supported by TinyOS and so writing platform specific code to use the same chips all over again does not make sense. The chip on a particular platform maybe physically wired differently in each platform but the logic that drives the platform is platform independent. We made use of this platform independence offered by TinyOS to integrate *KMote* with it.

Therefore, all the chip specific directories contain platform independent code. Integrating a new platform is then a matter of pulling in the code for each of the platform's chips, and linking them together. The steps involved in integrating of a *KMote* with TinyOS are explained in detail in Appendix D. These steps ensure that the basic skeleton of *KMote* is successfully integrated with TinyOS. This means that TinyOS recognises *KMote* and that platform definitions for TinyOS are in place. However, this is not sufficient to program the device as there is no platform specific code to drive the low level hardware which is specific to each hardware. We cover the explanation of methodology for testing and enabling the platform for programming in the chapter 4.

Chapter 4

Testing *KMote* and Results

Testing *KMote* was the last major phase in its development. As discussed in the previous chapter, the KMote is integrated with *TinyOS*. In this chapter, we will discuss the procedure to prepare the mote for programming. To be able to program the mote, MSP430 needs to be put into programming mode and hence bootstrapping of MSP430 is neccessary to proceed with programming of the mote. We had not yet written the serial interface driver and so decided to use JTAG interface instead. We first discuss the preparation of the JTAG interface and loading a program using JTAG and gdb on to the mote. We then discuss the testing procedure and subsequent results associated with these tests.

To enable programming of *KMote*, we followed a procedure that not only tested the mote for functionality but also made it ready to be programmed. To achieve this, we decided to test functionalities of the building blocks of the mote viz: Microcontroller and Radio. Hence we decided on the following steps to test *KMote*.

- Prepare the JTAG interface.
- Load a simple Blink application using LEDs to test the microcontroller.
- Program the mote so as to transmit and receive packets thereby testing the radio circuitry.
- During the above test, check the mote for transmission range
- Check the current consumption of the mote under various scenarios.

4.1 Preparing the JTAG interface

Joint Test Action Group (JTAG) interface is a common name for IEEE 1149.1 standard entitled Standard Test Access Port and Boundary-Scan Architecture. This is used for testing digital electronic systems including ICs and printed circuit boards using boundary scan. It is commonly used as a mechanism for debugging embedded systems, providing a convenient back door entry into the system. When used as a debugging tool, the debug module enables the programmer to debug the software of an embedded system.

As was described in the architecture of *KMote*, we designed JTAG extension header that has direct access to TDO, TDI, TMS, TCK, RST/NMI pins of MSP430 and put it into programming mode. In order to access the expansion header from a PC, we made use of MSP430-FET tool. This connects to PC through USB on one end and has a 14 pin connector on the other end. The expansion header we used for JTAG on PCB is an 8 pin one and so we needed an interface board that connects FET tool to the mote. The interconnections for such an interface are shown in Figure 4.1.

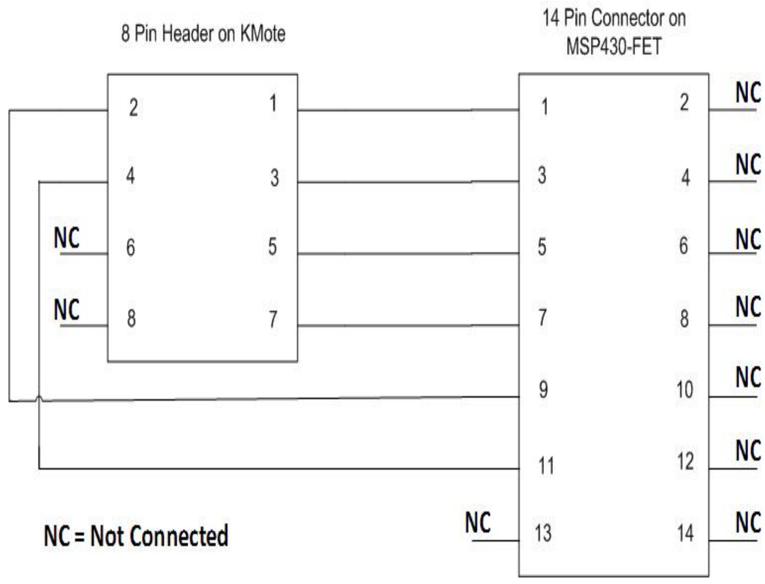

Figure 4.1. JTAG Interface for *KMote*

The mote is connected to TI MSP430-FET through the interface board and the other end of FET is connected to PC through USB port. A program called *msp430-gdbproxy* is run on the PC that initiates communication with the mote. Once the target device is detected, which in this case is MSP430 on the mote, a TCP connection on port 2000 is set up. The next step is to run *msp430-gdb* program thats open a gdb shell. A connection on port 2000 to MSP430 is established and this puts the microcontroller into programming mode. The application is compiled and loaded onto the flash memory of the mote using the *load* command through gdb. Using the *cont* command in *gdb* shell, the program is checked for its correct execution. On correct running of program, the JTAG interface is disconnected from the mote. The mote is then powered up and the bootstrap loader on the mote executes the program loaded on its on-board flash.

4.2 Testing the Microcontroller

The equivalent of a 'Hello World' in embedded systems is to be able to blink a few LEDs by programming the microcontroller. To be able to do so, we first needed to make the mote ready for programming. After its integration with TinyOS, we wanted to program the mote with a simple Blink application that blinks the LEDs interfaced with MSP430 based on a timer. During the integration of *KMote* with TinyOS, we had defined *.platform* file to define platform specific hardware for *KMote* and glued platform independent code associated with MSP430, CC2420 etc. to enable TinyOS to detect *KMote* platform. This is however not sufficient to drive the low level hardware interfaced with MSP430 as there is no corresponding platform specific code. In order to turn off and on LEDs, there needs to be a *PlatformLedsC.nc* file that defines the corresponding interfaces required. This file is platform specific as LEDs maybe connected to different GPIO pins of the microcontroller on different platforms and this file defines wiring between MSP430 and LEDs. The mote is then made ready to be programmed to test a basic Blink application.

We compiled the program in TinyOS and loaded the program through JTAG using the method described above. The program did not work for the first time. We then started checking the code and also whether the integration with TinyOS was correctly carried out. When we did not find anything, we checked the 32 KHz external crystal for correct operation and realised that there was an error in routing the connections to crystal from MSP430. We made corrections by cutting the tracks and re-soldering the wires. Once this was done, the program started running correctly and the LEDs interfaced to MSP430 on the mote started to blink. This proved that our microcontroller was indeed working and that we were able to program the mote.

We however discovered that on removal of JTAG interface, the blinking stopped and powering the mote up had no effect. The LEDs started to blink only when connected to JTAG and the program was executed through *gdb*. We felt there was some problem in the bootstrapping MSP430 but found no error. On closer inspection of the circuit, we realised that the power supply input to MSP430 was greater than the maximum specified value of 3.6 volts. On ensuring the correct supply voltage of 3.0 volts to MSP430, bootstrap loader started working normally.

4.3 Testing the RF Circuit

To test the RF circuit, we compiled a simple code in which a counter is started and the value of counter is put into the packet and transmitted. When the receiver receives the packet, it reads the counter value from the packet and sets LEDs ON or OFF based on the last three bits of the counter value. The receiver also generates a counter value and transmits back to the sender. In sum, two motes each run the same application that comprises of both transmission and reception and turn the LEDs on or off accordingly. As in the blink application, the same process of defining the hardware for radio was done and code compiled for KMote.

We loaded the application on to our embedded platform, and when we tried to run the program through *gdb*, the application did not work. Since JTAG is used as a debugging interface as well, we were able to see an error message that said that device state was unknown. Although, this error message was not very helpful, we could at least make out that there is some problem with the radio chip and that its either not being detected or that there is some problem with the state of the radio chip. A similar code executed on Tmote worked fine to prove that our code was indeed correct.

After many unsuccessful attempts, we then decided to go by the first principles and start checking if the radio is receiving the correct voltage to power it or not. On examining the voltages at various pins as specified by CC2420 datasheet (reference to datasheet), we realised that none of the voltages are available on any of the pins. We were powering the mote with external adapter and when we checked the voltage from it, we found that it was 8.9 volts instead of the specified 4 volts. Also, our voltage regulator circuit seemed to be wrong as it was not regulating the input voltage to the specified 3.3 volts but just reducing the input voltage by 1.0 volt. We then decided to disconnect the external power supply adapter and then power the board using the battery source instead. The radio started transmitting after this change as desired. We learnt a very important lesson here. Commercial power supply adapters are not very reliable for such embedded platforms, where the voltage requirements are very stringent. It is better to design a reliable and precise power supply adapter and use that instead.

4.4 Radio Range Measurements

The methodology followed for RF range calculations is similar to the one we used in our work [20] with a slight modification. The experiment setup for the measurements is shown in Figure 4.2.

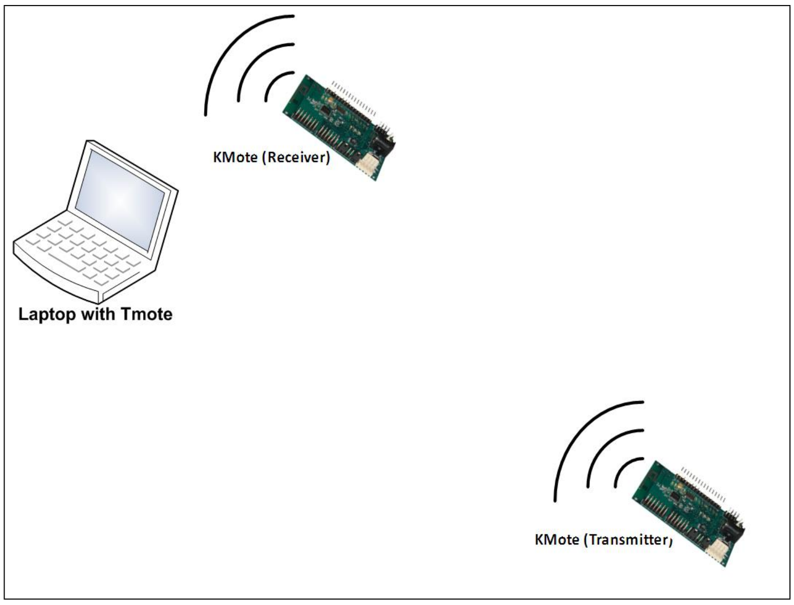

Figure 4.2. Experimental Setup

We used one *KMote* as a packet generator to transmit packets once every 20 ms interval and another as a receiver. On receipt of a packet, the receiver measures the RSSI value of the packet and stores the serial number and its corresponding RSSI value. On completion of transmission, we press the *user* button on the mote which then transmits the stored values to a third mote, Tmote, present in the vicinity of the receiver mote. Tmote ran the TOSBase (a packet sniffer application that captures all the packets transmitted over air) application and is interfaced to a PC and logs these values to a file. KMote can be interfaced to a PC using serial port. However, the drivers required to interface KMote to serial port of PC are not fully developed. Hence, we used the third mote (Tmote) which is interfaced to PC using its USB interface and captures the packets transmitted from *KMote* over a wireless link. The need for a third mote will be eliminated when the serial interface for *KMote* gets ready and it can then directly log the received values to file on a PC.

We checked the radio range in two environments - an airstrip and a road with some foliage on either side of the road. On the airstrip, where its an open space and has very low multi-path effect, a range of about 220m meters was obtained with an average RSSI of about -84dBm. This experiment was repeated for about 5 times in succession and the results have been consistent. The RF range on the road was predictably lesser than the Airstrip environment and it was about 150 meters with an average RSSI of about -84.1 dBm. This can be attributed to the multi path effect due to presence of trees and other obstructions on either side of the road. Note that the values achieved using *KMote*s is better than the ones achieved using Tmotes in our earlier work [20].

4.5 Power Consumption Measurements

To measure the power consumption, we connected an ammeter in series with the power source and measured the current drawn. The results obtained and a comparison of these values with those of Tmote (as per its datasheet [12]) are as shown in the Table 4.1. The values measured are similar to those of Tmote.

Measured Parameter	**KMote**	**Tmote**
Current Consumption: MCU on, Radio TX	19.3 mA	19.5 mA (Nominal) 21 mA (Max)
Current Consumption: MCU on, Radio RX	21.1 mA	21.8 mA (Nominal) 23 mA (Max)
Current Consumption: MCU on, Radio Off	1.5 mA	1.8 mA (Nominal) 2.4 mA (Max)
Current Consumption: MCU standby	6.7 μ A	5.1 μ A (Nominal) 21 μ A (Max)

Table 4.1. Power Consumption of KMote

4.6 Cost Analysis of *KMote*

One of the primary aims of the work was development of low cost embedded platform for wireless sensor networks. In this section we make a cost comparison of *KMote*, Tmote and Tiny Node and argue in favour of development of such platforms in India. Before we compare *KMote* with other state of the art embedded products of similar nature, we show the breakdown cost of *KMote* during its development in Table 4.2. This estimate is for development of 100 motes. The initial cost for art work of PCB during its fabrication is about Rs 7000 (165 USD) and may be ignored as a one time investment.

Description	**Cost**
Cost of Manufacturing a PCB	Rs 100 (2.40 USD)
Cost of Components	
Microcontroller (MSP430)	Rs 372 (8.65 USD)
Radio (CC2420)	Rs 210 (4.86 USD)
Voltage Regulator(MCP3302)	Rs 13 (0.3 USD)
Flash (STM25P80)	Rs 155 (3.62USD)
Serial ID Generator (DS2411)	Rs 23 (0.54 USD)
RS232 Trans-receiver (MAX3380)	Rs 120 (2.78 USD)
Temperature Sensor (TMP36FSZ)	Rs 17 (0.4 USD)
16 Mhz Crystal (for Radio)	Rs 185 (4.3 USD)
Other Passive SMD Components	Rs 430 (10 USD)
Total	**Rs 1625 (37.85 USD)**

Table 4.2. Bill of Material Cost of KMote

The comparison of costs between the three platforms is shown in Table 4.3.

Serial Number	**Product**	**Cost**
(a)	Tmote	Rs 5590 (130.00 USD(without sensors)) Rs 7740 (180 USD (with sensors))
(b)	Tiny Node	Rs 7740 (180.00 USD)
(c)	**KMote**	**Rs 1625 (37.85 USD) (BOM cost only)**

Table 4.3. Cost Comparison of Motes

There is also an option of purchasing a set of 10 Tmotes at a discounted price of Rs 47,300 (1100 USD, with sensors) and Rs 33,540(780 USD, without sensors). We had procured Tmotes at an offer price of Rs 4300 (100 USD, with sensors). From the Table 4.3 it is very evident that development of such a mote in India does have huge cost benefits. Not all educational institutes in India have administrative procedures in place to import these motes. Further, there is a large lead time in procurement and it takes about 4-6 weeks for delivery. Also, purchasing these motes from the suppliers involve an additional shipping and handling charges which are about the order the cost of *KMote* itself. In addition, to enable JTAG testing and debugging, Tmote offers a JTAG interface board for 20 USD. This, we feel is highly expensive as the same board can be made on a small PCB for about Rs 10 or about 0.23 USD!.

4.7 Lessons Learnt

During the course of our work we learnt a number of lessons that would be beneficial in future design or to the developers who would like to develop such platform. A brief mention of the important lessons learnt are as under:

- Prior to taking up work on such embedded platforms, some experience in embedded design is mandatory.
- It is advisable to take up design of some small board using the EDA software before commencing work on design of embedded platform. This has dual advantages: (i) This will help in exploring the EDA software and get used to the software. The software we used (Protel DXP 2004) was very helpful but had a steep learning curve and design of accelerometer boards using this was very helpful in our work. (ii) This would also give an experience to the kind of work involved while translating the design from software to a working PCB. Issues like usage of board fiducial points, correct placement of components, alignment of top and bottom layers, choice of track width etc are better understood.
- Integrating radio chip and microcontroller was quite a challenging job even when we had the reference design with us. The choice of track width for connecting chip output to feed of the antenna is very critical. Also the number of ground vias around the radio circuitry is very important. The more the number of such vias, the better the design.
- Placement of components is very critical to a working design. The process is very laborious and time consuming, but it is worth the effort to ensure that PCB looks good. As is the general rule of thumb, a good looking PCB definitely has a better chance of being a working PCB.
- It is better to crosscheck the design a number of times and by different people before sending it for fabrication and later realise that there has been some routing errors on the PCB after its fabrication. We experienced one such error in labeling the pins of crystal in its footprint and it ultimately translated to erroneous routing of tracks on PCB. Though, we were fortunate to have a simple way to rectify it, it may not always be the case.
- Usage of correct power supply is very critical for the correct functioning of the platform. We were once again fortunate that usage of a much higher voltage than the specified one did not damage the chip. Had this happened, would have probably wasted some more time and effort to realise this. It is always advisable to use a battery source when testing the design where external power supply adapters are involved as the adapters available commercially are not very reliable and pose problems similar to the ones faced by us
- Lead time in fabrication of PCBs is about 4-6 weeks on an average if the design is perfect and there are no modifications required due to the manufacturing process. This time can increase if manufacturer needs design alterations and can go up to as high as 8-10 weeks as experienced by us. Therefore, it is imperative that the design be free from any error prior to

submitting to manufacturer. Also, the cost of initial investment for preparing the artwork is high compared to the cost of fabrication of each board. This process also takes up most of the time in fabricating a PCB. Hence it is very important that PCB design be free from any routing error and save both cost and time.

Chapter 5

Conclusions and Future Scope

The idea of using small, low cost wireless devices for sensing has created a paradigm shift in contact-less, non-intrusive, distributed sensing. Initial experimental deployments of such networks using some of the commercially available sensor nodes are helping people understand the benefits that such networks can offer. The continued growth and evolution of such an experimental field into a mature technology can only occur if people in research and educational institutes have cheap, easy and ready access to suitable hardware platforms. However, state of the art sensor nodes available in the market cost a few thousand rupees and have to be imported. The high cost coupled with all the administrative hassles required to import these devices, makes access to them difficult for most educational and research institutes in India.

In this thesis we have designed and implemented a low cost, low power embedded device called the *KMote* that offers all the capabilities provided by commercially available sensor nodes. The *KMote* has been tested and provides communication ranges of about 150m which is nearly double the range of Tmote, the most popular commercially available state-of-art sensor device. The device is powered by two AA size batteries and consumes about 20 mA during transmission. *KMote* is a research prototype and one would expect the cost of a research prototype to be higher than that of the commercially available motes. The BOM cost of *KMote* is Rs 1625. The total cost of the mote as per industry standards can be computed to be around Rs 5000 which is 3 times the BOM cost. This cost is lower than the cost of a Tmote. Hence it becomes a viable option to fabricate these boards in India for the advantages offered by such a development.

It is our belief that the easy availability of such a platform at a fraction of the cost of equivalent commercially available platforms will allow increased participation by research and educational communities in India in the evolving field of wireless sensor networks and develop applications to target problems specific to India.

In this work, we have successfully designed and implemented the core functionalities of a mote by proving the operation of microcontroller and radio circuitry. However, work on integration of a couple of additional software modules into TinyOS is in progress. The list of future to-do's are:

- Modification of bootstrap loader program to incorporate the serial port interface for programming the mote.
- Integration of LCD panel.

This is the first hardware version of the mote and the experience gained has shown a path for design of better hardware/increasing the flexibility of the mote. A few of the scenarios that may be considered for future design are as under.

- The design may further be improved through usage of new generation microcontrollers that integrate radio and microcontroller on a single chip. This will enable reduction of the board size as the circuitry for interfacing radio with microcontroller will no longer be needed. This will also free the SPI lines of microcontroller and facilitate integration of additional flash/other high speed peripherals.
- Explore the feasibility of using a 32 bit low power microcontroller such as those from Atmel AVR32 UC3 processor family. This would increase the processing capability of the motes and they can be designed for complex applications.
- Explore the possibility of porting a tiny version of Linux on to the on-board flash. Porting an OS such as Linux would increase the flexibility of the mote.
- A design that includes an RF power amplifier at the output stage of the radio chip seems to be an interesting possibility to further increase the range of the mote using the on-board antenna.
- Design of a directional antenna on the PCB can be considered that increases the range of the mote. The motes can then be used in point to point long distance links.

check to see if the text appears on the same page or not..blah blah blah..

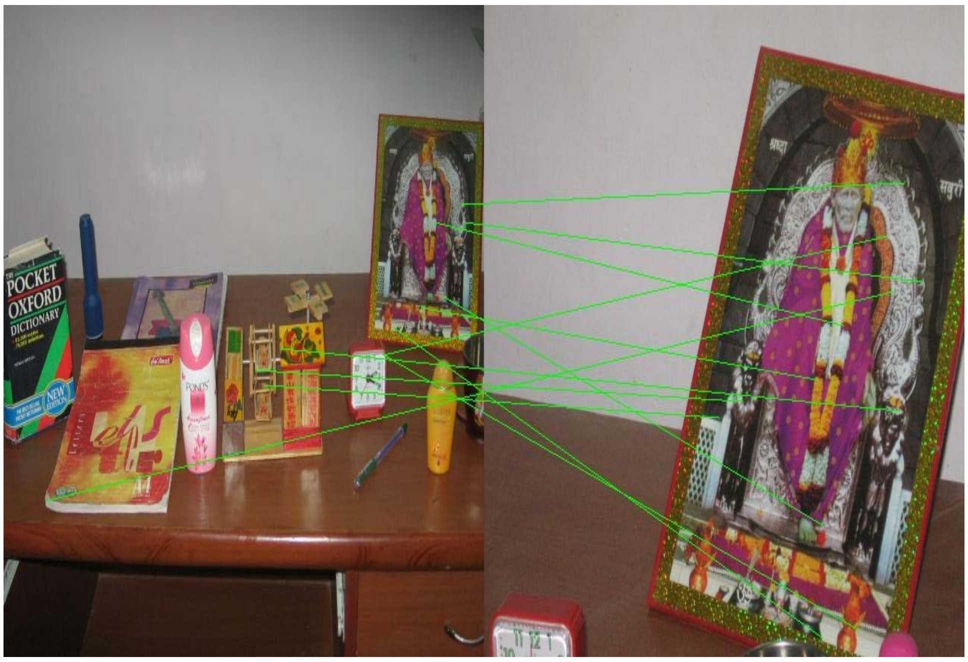

Figure 5.1. Correspondence

Appendices

Appendix A   Schematic of Top Layer

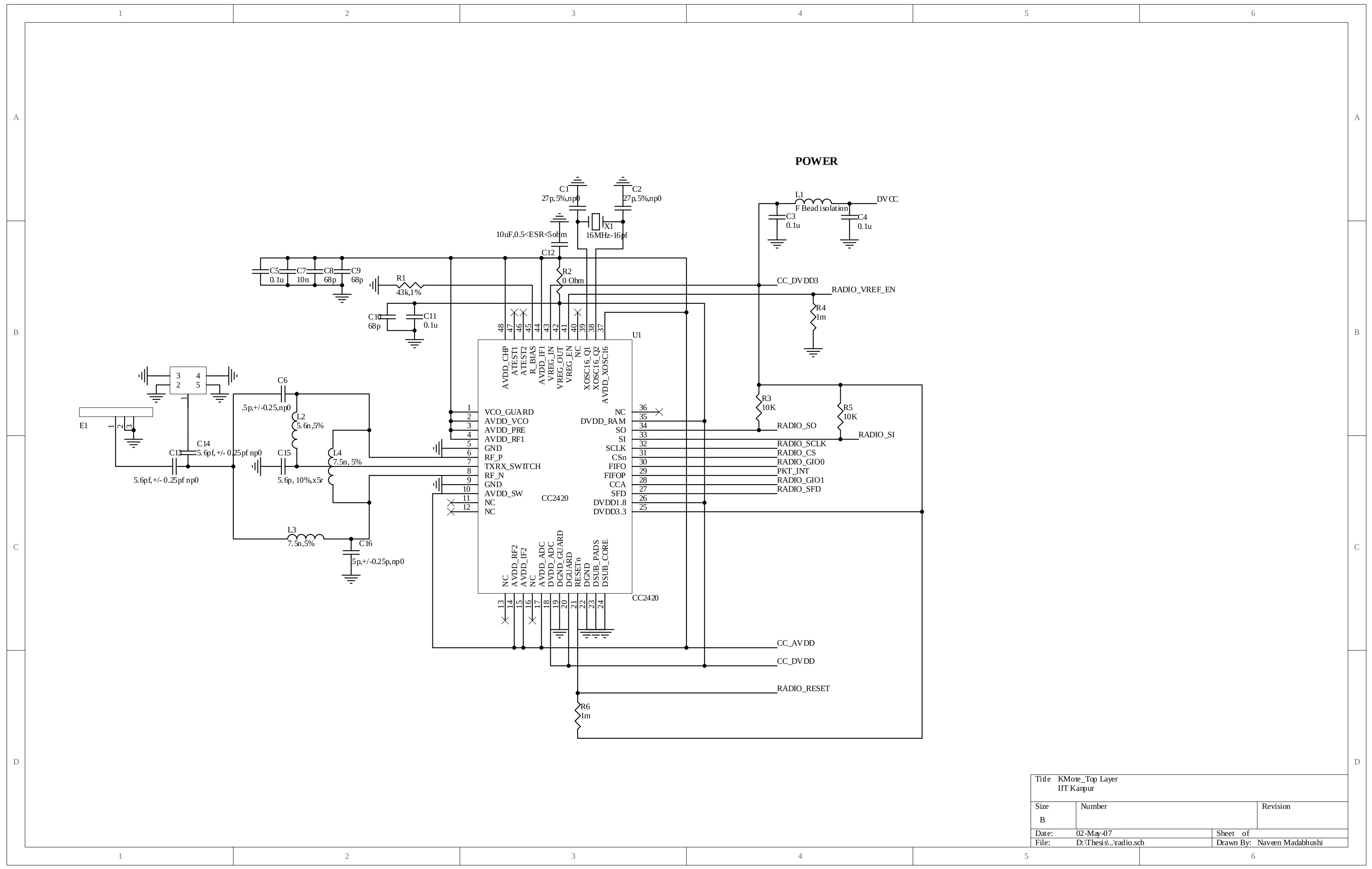

Figure 2. Schematic of $KMote$ Top Layer

Appendix B Schematic of Bottom Layer

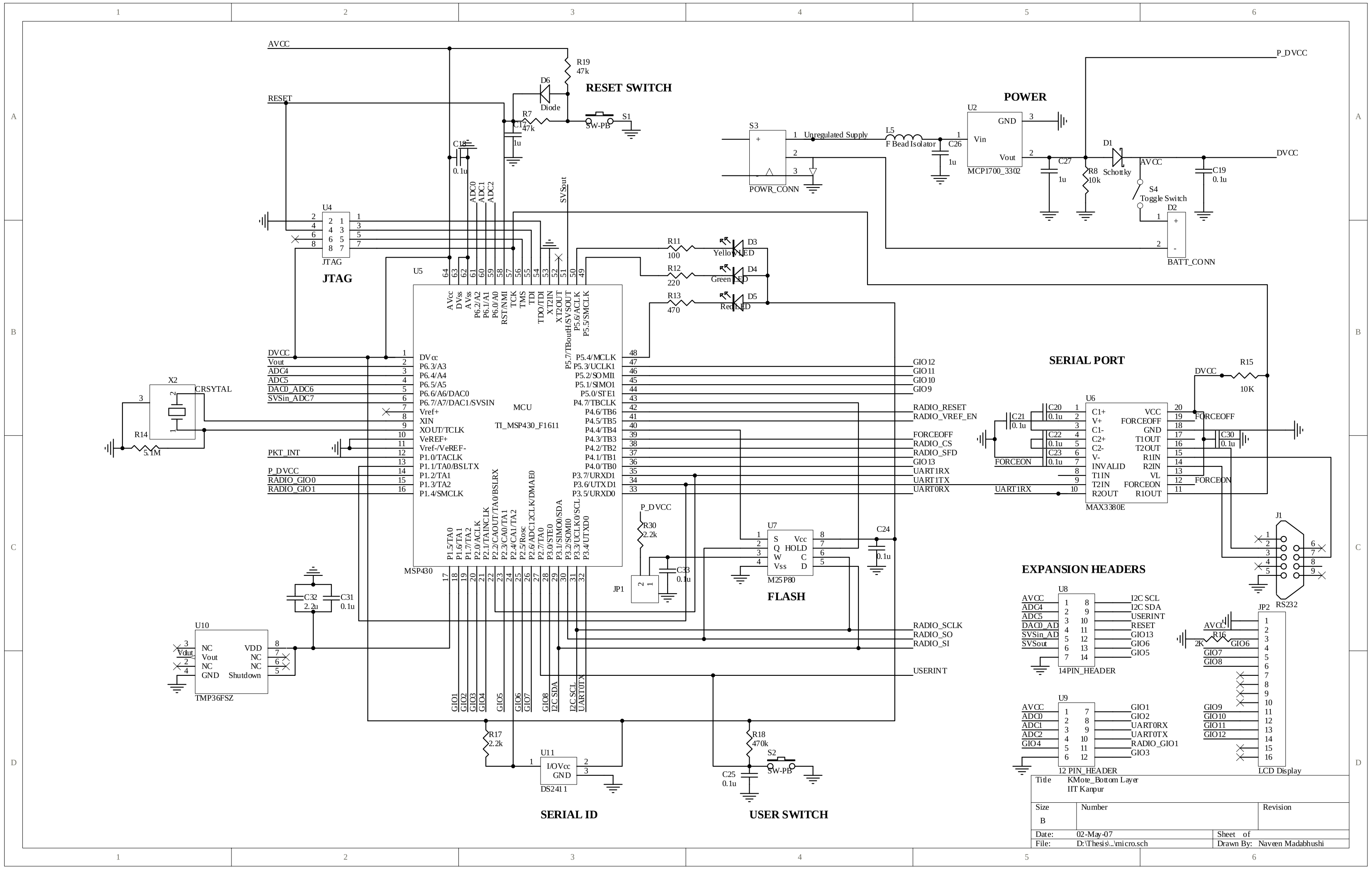

Figure 3. Schematic of $KMote$ Bottom Layer

Appendix C Procedure to Connect External Antenna to *KMote*

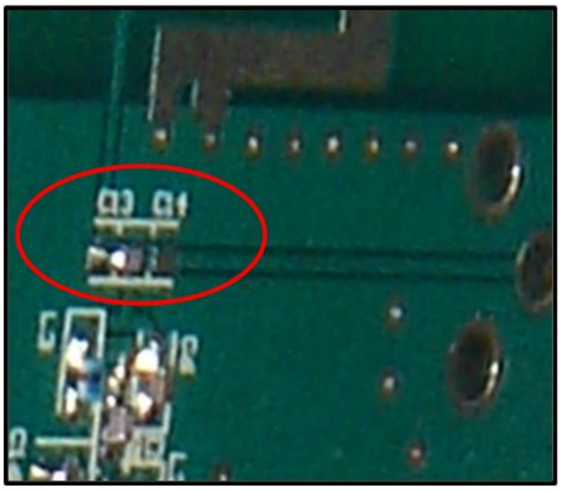

Figure 4. Position of capacitors C13 and C14

To connect an external antenna, the path to antenna has to be disconnected and the output from radio chip has to be connected to the track leading to SMA Connector. To achieve this capacitor C13 has to be de-soldered and soldered on the pads for capacitor C14. The positions of capacitors C13 and C14 on the mote are shown in in Figure 4. Further, an SMA connector has to be soldered to the five vias provided on the PCB. The central via connects to the antenna feed while the other four vias are connected to ground. A 50 Ω can then be connected to the SMA connector to extend the range of *KMote*.

Appendix D Steps Involved in Integrating *KMote* with *TinyOS*

In chapter 3, we briefly described the procedure for integrating *KMote* with *TinyOS*. In this section, we cover in detail the steps involved in integrating the mote. The steps are as enumnerated under:

- Create a new directory for KMote under tos/platforms
- Create a .platform file that contains basic compiler specific information for each platform.
- The next step is to create a hardware.h file and put it in the same directory as the KMote directory. This file is included by default when compiling an application for that platform. This file defines platform-specific constants, pin names, and also include other "external" header files (e.g. msp430hardware.h)
- <u>Setting up the build environment for KMote</u>. Prior to compiling any application, it is important for build environment of TinyOS to be aware of a platform called KMote. The TinyOS build environment is a set of Makerules and definitions that include invoking necessary compilation commands and includes support for other important aspects such as device reprogramming.
- <u>Creation of PlatformP and PlatformC files</u>. After letting the build environment know of existence of KMote, it is also necessary to include PlatformP and PlatformC files. The requirement of these files is described in detail in TEP107 [11]. The basic idea to define these files is that these files provide one and only one instance of *Init* interface.

Appendix E Procedure to load programs on to ***KMote*** using JTAG and ***gdb***

In chapter 4, we mentioned about testing of *KMote* using JTAG and *gdb*. In section, we describe the procedure to do the same in *TinyOS*. A detailed description on the procedure to setup debugging environment under linux is available at [2]. The steps for setting up *gdb* under *Cygwin* are as under:

- Prepare a JTAG interface board as per the connection diagram shown in Figure 4.1.
- Download mspgcc-20060502.exe from [6]. This installer has two packages, msp430-gdb and msp430-gdbproxy which are required to setup the debugging environment.
- Carry out a custom installation and uncheck all the boxes except "Debugger" and "JTAG and BSL".
- Copy msp430-gdb and msp430-gdbproxy to MSP430 tools directory.
- If MSPGCC is installed in the default location, then the following command will perform the necessary task: *cp /cygdrive/c/mspgcc/bin/msp430-gdb* /opt/msp430/bin*
- Copy the files HIL.dll and MSP430.dll using the following commands. These files are needed to interface with TI MSP430FET pod: *cp /cygdrive/c/mspgcc/HIL.dll /opt/msp430/bin* and *cp /cgdrive/c/mspgcc/MSP430.dll /opt/msp430/bin*
- Connect the other end of JTAG interface board to TI MSP430FET pod and plug the USB end of the pod to PC. The open a shell and run the following command: *msp430-gdbproxy msp430 TIUSB*
- If everything works as it should, the rest of the procedure is same as described at [2]

Bibliography

[1] CC2420 Datasheet. www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf.

[2] Debugging with GDB under Linux. http://www.eecs.harvard.edu/~mdw/proj/tmote-gdb/.

[3] Elfsys. http://www.elfsys.net.

[4] Folded Dipole Antenna for CC2420. http://www.chipcon.com/files/AN_040_Folded_Dipole_Antenna_for_CC24XX_1_0.pdf.

[5] MAX3380 Transreceiver Datasheet. http://datasheets.maxim-ic.com/en/ds/MAX3380E-MAX3381E.pdf.

[6] MSPGCC. http://sourceforge.net/project/showfiles.php?group_id=42303.

[7] Schematic of telosb. http://webs.cs.berkeley.edu/tos/hardware/telos/telos-revb-2004-09-27.pdf.

[8] ST M25P80 Flash Datasheet. http://www.st.com/stonline/books/pdf/docs/8495.pdf.

[9] Temperature Sensor TMP36FSZ Datasheet. http://www.analog.com/UploadedFiles/Data_Sheets/TMP35_36_37.pdf.

[10] Tiny Node Datasheet. http://www.btnode.ethz.ch/pub/uploads/Projects/tinynode-datasheet.pdf.

[11] Tinyos TEPs. http://www.tinyos.net/scoop/special/working_group_tinyos_2-0.

[12] Tmote Sky Datasheet. http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf.

[13] Voltage Regulator MCP1700-3302 Datasheet. http://ww1.microchip.com/downloads/en/DeviceDoc/21826b.pdf.

[14] XE1205 Radio chip Datasheet. http://www.semtech.com/pc/downloadDocument.do?id=769.

[15] Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L. S., and Rubenstein, D. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems* (New York, NY, USA, 2002), ACM Press,

pp. 96–107.

[16] Kim, S., Pakzad, S., Culler, D., Demmel, J., Fenves, G., Glaser, S., and Turon, M. Health monitoring of civil infrastructures using wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks* (New York, NY, USA, 2007), ACM Press, pp. 254–263.

[17] Krishnamurthy, L., Adler, R., Buonadonna, P., Chhabra, J., Flanigan, M., Kushalnagar, N., Nachman, L., and Yarvis, M. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems* (New York, NY, USA, 2005), ACM Press, pp. 64–75.

[18] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications* (New York, NY, USA, 2002), ACM Press, pp. 88–97.

[19] Polastre, J., Szewczyk, R., and Culler, D. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks* (Piscataway, NJ, USA, 2005), IEEE Press, p. 48.

[20] Raman, B., Chebrolu, K., Madabhushi, N., Gokhale, D. Y., Valiveti, P. K., and Jain, D. Implications of link range and (in)stability on sensor network architecture. In *WiNTECH '06: Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization* (New York, NY, USA, 2006), ACM Press, pp. 65–72.

[21] Sheth, A., Tejaswi, K., Mehta, P., Parekh, C., Bansal, R., Merchant, S., Singh, T., Desai, U. B., Thekkath, C. A., and Toyama, K. Senslide: a sensor network based landslide prediction aystem. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems* (New York, NY, USA, 2005), ACM Press, pp. 280–281.

[22] Szewczyk, R., Mainwaring, A., Polastre, J., Anderson, J., and Culler, D. An analysis of a large scale habitat monitoring application. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems* (New York, NY, USA, 2004), ACM Press, pp. 214–226.

[23] Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay, D., and Hong, W. A macroscope in the redwoods. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems* (New York, NY, USA, 2005), ACM Press, pp. 51–63.

[24] Warneke, B., Last, M., Liebowitz, B., and Pister, K. S. J. Smart dust: Communicating with a cubic-millimeter computer. *Computer 34*, 1 (2001), 44–51.

[25] Werner-Allen, G., Lorincz, K., Welsh, M., Marcillo, O., John-

son, J., Ruiz, M., and Lees, J. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing 10*, 2 (2006), 18–25.