8-2013

ILSN : An Inexpensive, Long-Lived, Sensor Network Solution

Navin Soni
Clemson University, navinsoni89@gmail.com

ILSN : An Inexpensive, Long-Lived, Sensor Network Solution

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Masters of Science
Computer Science

by
Navin N. Soni
August 2013

Accepted by:
Dr. Jason O. Hallstrom, Committee Chair
Dr. Brian A. Malloy
Dr. Jacob M. Sorber

Abstract

In the recent past, there has been a phenomenal increase in monitoring the physical world using wireless sensor networks comprising tiny computing devices with integrated sensors. These devices are deployed in large numbers, often across large spaces and within hostile environments.

Wireless sensor networks are used to gather meaningful data and to enable important applications. They must satisfy some basic requirements. Specifically, they must be *maintenance free*, *inexpensive*, *reliable*, and *scalable*. The devices used in these networks are deployed in the hundreds to thousands and rely on battery power. It is expensive to change these batteries in such large networks, both in terms of battery cost and personnel time. The cost of an individual device plays an important role in the cost of a sensor network. Further, if these networks are deployed in safety critical contexts, we cannot risk having incorrect data or missing important data. Finally, sensor networks must be able to accommodate new devices and gracefully handle device failures.

This thesis describes a hardware/software solution for wireless sensor networks which is maintenance free, inexpensive, reliable, and scalable. In this thesis, I present a wireless sensing device which harvests solar energy and stores it in a Li-Ion battery. The device works on solar energy during the daytime and relies on battery power during the night. This addresses the problem of maintaining remote devices. The design is also focused on reducing component costs. The cost is low enough to discard the individual devices without significant concern. Finally, using these devices, I present a network protocol and reference implementation which makes data reception reliable, while supporting network scalability.

Dedication

to my loving parents....

Acknowledgments

I owe my deepest gratitude and respect to my advisor, Dr. Jason O. Hallstrom. If it were not for his ideas, support, guidance and motivation, none of this would have been possible. I would like to thank Dr. Brian Malloy and Dr. Jacob Sorber, who served on my committee, and who encouraged and motivated me.

I thank Yvon Fester and Cullum Smith for their encouragement and suggestions. I am greatly in debt to Ravi and Aravindh for their amazing company. I would also like to thank Yang, Yuheng, Gyan, Neeraj, Sanjay, and Jiannan, who always encouraged and supported me. They made graduate school and life much more enjoyable. A special thanks to everyone who wished well for me. Last, but not least, I thank my parents and brother for all their love, sacrifices, and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless sensor networks consist of small computing devices known as *motes*, connected to each other wirelessly, in the hundreds to thousands of devices. Motes are equipped with sensors to sense physical and/or chemical parameters, and radios to transmit the sensed data to a base station for further processing and storage. These networks are important tools for monitoring the physical world. They can be used to monitor volcanoes [57], the structure of buildings and bridges [8], the flow of water [56], soil moisture [6], and other phenomena of interest. These are only a few examples of what wireless sensor networks can support. We can store, process, and analyze the data gathered from these networks to aid in analytical tasks that would not have been possible just a decade ago. Data gathered from a vibration sensor, for example, might be used to enable predictions regarding the lifetime of a bridge. Data gathered from an environmental network might be used for monitoring air pollution [35] or water quality [59]. Data gathered from an earth monitoring network might be used for landslide detection [29].

1.1 Motivation

Ideally, wireless sensor networks should be maintenance free, inexpensive, reliable, and scalable. But like every other type of system, wireless sensor networks have limitations. The first involves network maintenance, a significant obstacle. Most of the maintenance time is spent in replacing depleted batteries [58]. Changing batteries in a large wireless sensor network can be expensive and time consuming. The cost of deploying a network also plays an important role. We should be able to deploy low cost, even disposable motes, in order to achieve better sensor coverage. We must also receive data reliably. Wireless sensor networks are often deployed in remote and perhaps in hostile environments, in large numbers [58]. We cannot risk receiving incorrect data or missing important data. Finally, we should be able to accommodate the dynamic addition or unexpected removal of a mote in a network without affecting reliability.

Our solution approach satisfies all of these requirements. We present the design of a mote that is maintenance free and inexpensive. We present a networking solution that is reliable and scalable.

1.2 Problem Statement

The objective of this thesis is to develop a wireless sensor network that satisfies the stated key characteristics. First, the mote hardware must be maintenance free. To achieve a reduction in maintenance overhead, we must design a mote that harvests energy continuously. Second, the cost of each mote must be reduced to allow users to cover a large area at a reasonable cost. To achieve this reduction, we must develop a low-cost mote that provides the necessary components for sensing and transmitting data to a base station. Third, the network must be reliable. We deploy wireless sensor networks in remote, and perhaps even hostile environments. We cannot risk receiving incorrect data or losing data. Fourth, the network must be scalable. It should be able to accommodate the dynamic addition and unexpected removal of a mote without affecting overall reliability.

1.3 Solution Approach

Our solution approach consists of several steps. First, to achieve a reduction in maintenance time, our design includes an energy harvesting circuit, which harvests solar energy and stores it in a Li-Ion battery. This stored energy is used whenever there is insufficient solar power. Further, we know energy saved translates to longevity improvements, so we use custom software to keep the microcontroller in a sleep state whenever possible. Second, we focus on low-cost hardware to sense the physical and chemical parameters in a micro environment. We rely on components that are low cost, but still sufficient for a simplified mote design. Instead of using more common, costly radios such as the CC2420 (10.08 USD) [26], the CC2400 (8.77 USD) [24], and others, we use the

RFM12 (5.56 USD) [17]. This is an inexpensive radio, but has all the features of more popular radios, including high data-rate capabilities, variable transmission power, multi-channel operation, and long range . Further, the RFM12 doesn't require any external components to transmit and receive data, which simplifies the design, whereas the CC2420, and CC2400 each need many external components to function. A complex design and additional components increases the cost of manufacturing. We are reducing the cost of the mote by simplifying its design and using components which best satisfy our needs. Third, we develop network protocols to receive data reliably. To achieve this, we develop a protocol which uses a lightweight time-division multiple access approach (TDMA) [7], with acknowledgments to ensure that every packet reaches the base station. Fourth, we develop a network that accommodates the dynamic addition and removal of motes without affecting network reliability. The resulting network typically operates at a 10% duty-cycle, saving significant power.

1.4 Contributions

This thesis describes a low-cost, maintenance-free sensor networking platform, supported by lightweight, yet reliable and scalable networking software. Design simplicity is a focal point throughout. The hardware/software solution exhibits four key characteristics. First, we describe a —maintenance free solution. We accomplish this by harvesting solar energy and by efficiently using the available energy. Second, we describe an —inexpensive solution. We accomplish this by designing a mote platform which uses basic electronic components —resistors, capacitors, and diodes —instead of more complex integrated circuits. Third, we describe a —reliable solution that ensures high yield, even in the presence of intermittent and permanent device faults[1]. We accomplish this by developing lightweight route formation, time synchronization, and TDMA protocols; the latter includes application-level acknowledgments to ensure reliable data transmission and reception. Finally, we describe a —scalable solution that accommodates the dynamic addition and removal of motes without affecting network reliability. The sum total of these characteristics yields a low-cost sensor networking solution that enables large-scale, long-lived network deployments —the contribution of this thesis.

[1]The solution assumes that the underlying physical topology always accommodates a connected routing topology.

1.5 Thesis Organization

Chapter 2 surveys the most closely related work in the field of wireless sensor networks and solar energy harvesting. Chapter 3 describes the design of the hardware. Chapter 4 describes the design of the network software and associated protocols. Chapter 5 describes the results of our evaluation. Finally, chapter 6 presents a summary of contributions and conclusions.

Chapter 2

Related Work

2.1 Hardware Platform

This section considers existing hardware platforms designed to sense parameters of the physical world. The discussion is divided into four subsections, focused on solar energy harvesting, vibrational energy harvesting, radio frequency energy harvesting, and other modern sensing platforms.

2.1.1 Solar Energy Harvesting Platforms

2.1.1.1 TinyNode

Ferriere et al. present a platform for wireless sensor networks called TinyNode [16]. TinyNode consists of a core module and an array of extension boards to provide additional functionality. This modular design enables the platform to be tailored in an application-specific manner. The core module consists of an MSP430F1611 [25] microcontoller, an XE1205 [50] transceiver, flash storage, a voltage regulator, and an extension connector. TinyNode offers a wide range of options for energy, storage, and communication. The platform offers ethernet, WLAN, and GPRS connectivity, and is supported by the TinyOS [31] operating system. By including a solar harvesting board, the device is capable of long-term outdoor operation. TinyNode utilizes a two-tier architecture for energy storage. The primary storage component is a super capacitor, and the secondary storage component is a Li-Ion battery. The main advantage of this hierarchical structure is a reduction in the number of charging cycles for the Li-Ion battery. Voltage is measured at the primary storage component, and the secondary storage component; solar current is also measured.

2.1.1.2 Heliomote

Lin et al. present a system to harvest solar energy called Heliomote [33]. Heliomote can be used to power a variety of sensor nodes, including those in the Mica and Telos family. Heliomote uses solar panels to harvest solar energy from the environment and stores the harvested energy in super-capacitors and batteries. While similar to the TinyNode harvesting design, Heliomote implements enhanced features to increase the efficiency of energy harvesting. First, it is autonomous in making decisions about energy harvesting and energy storage. It has a dedicated circuit to track the maximum power point (MPP) of the solar panels, which it uses to maximize the energy transfer from the solar panels to the storage devices. MPP tracking is adaptive and does not depend on the specific solar panel being used. Next, it has overcharge and undercharge circuits for optimal utilization of the battery and super-capacitor. Finally, the output voltage level is configurable. It also exposes an I2C interface that allows the connected wireless sensor node to gather information about the harvesting circuit.

2.1.2 Other Platforms

2.1.2.1 Mica motes

A research team at the University of California, Berkeley designed a family of platforms for wireless sensor networks called Mica motes [15]. After designing the first iteration of the Mica platform, the team created the Mica2 [12], Mica2Dot [13], and MicaZ [14] platforms with similar architectures, all of which were widely adopted. Each mote in the Mica family consists of a micro-controller, a transceiver, flash storage, and a voltage regulator. The communication frequency of the Mica motes vary depending on their generation. Mica2 motes use an ATMEGA128L [2] micro-controller and communicate at a frequency of 916 MHz, whereas MicaZ motes communicate at a frequency of 2400 MHz. The Mica2Dot is a smaller version of the Mica2 —approximately the size of a quarter. All the Mica platforms are supported by the TinyOS operating system.

2.1.2.2 Telos mote

Polastre et al. present a platform for wireless sensor networks called the Telos mote [45]. Telos is an ultra-low power wireless sensor node. Telos motes consist of an MSP430 microcontroller, a CC2420 transceiver [26], and on-board sensors. Each Telos mote has a planar inverted folded antenna built on the printed circuit board, eliminating the need for an external antenna. In addition, each sub-circuit is isolated, allowing power to be toggled on and off for each sub-circuit. The primary benefit of Telos motes is their low power consumption.

In contrast to motes described in subsections 2.1.1 and 2.1.2, our hardware platform is a low-cost solution for wireless sensor networks. It use basic electronic components to reduce the cost of the device, without affecting its functionality. It has energy harvesting circuit for maintenance free operation for long periods of time. It can power the processing circuit and charge the battery simultaneously. It has a decision making circuit that can autonomously select between solar power supply and battery power supply depending on the voltage of the battery.

2.1.3 Vibrational Energy Harvesting Platforms

Ottaman et al. [41] [42], Sodano et al. [53] [54], and Paradiso et al. [43] focus on vibrational energy for energy harvesting. Piezoelectric elements are used to harvest vibrational energy. These elements transform ambient vibrations into electrical energy, which is then stored and used to power other devices. However, in most cases, the energy produced by these elements is too small to directly power an electrical device. Hence, the energy generated by these elements is stored in supercapacitors and used only when a specific energy level is reached. These devices have a short execution life. In contrast, our system harvests solar energy, which in most cases generates enough energy to directly power an electrical device.

2.1.4 Radio Frequency Energy Harvesting Platforms

Smith et al. describe a platform called WISP [52] [48] [44], consisting of a general-purpose programmable flash microcontroller, RFID [51] tags, and sensors. WISP, having no secondary energy storage components, harvests all of its energy from radio frequency signals. As a result, it has a very short execution life and cannot be used to perform complex operations. There are many applications of WISP, including a simple RFID sensor network [5], and radio-triggered wake-ups with addressing capabilities for extremely low power sensor networks [1]. In contrast, our system has a long life and can perform complex operations.

2.2 Data Routing Protocols

This section discusses prior work in the development of data routing protocols to sense and collect parameters of the physical world. The discussion is divided into three subsections, focused on hierarchical routing protocols, data-centric protocols, and location based routing protocols.

2.2.1 Hierarchical Routing Protocols

This subsection discusses some of the most important hierarchical data routing protocols used to sense and collect data.

2.2.1.1 LEACH

Heinzelman et al. developed a hierarchical routing protocol called LEACH [22] for sensor networks. LEACH forms clusters of sensor nodes based on their received signal strength. Each sensor node selects the cluster head that requires the minimum amount of energy for communication. It uses local cluster heads as relays to the ***root*** mote. This saves energy, as only cluster heads transmit data to the root mote. In LEACH, data processing takes place locally within clusters. In order to balance the energy dissipation of motes, cluster heads change randomly over time. LEACH does not require any prior knowledge of the network. However, LEACH forms a two-hop network, where each mote transmits data to the local cluster head, and the cluster heads transmit data to the root mote. This makes LEACH unsuitable for large networks. In contrast, our protocol forms a multi-hop network, making it suitable for large, geographically distributed networks.

2.2.1.2 PEGASIS

Lindsey and Raghavendra describe a hierarchical routing protocol called PEGASIS [34] for sensor networks. In PEGASIS, sensor motes join with their neighbors to form chains so that they can transmit and receive data from each other. One node from this chain is selected to transmit the aggregated data to the root. Due to chain formation, PEGASIS introduces additional delays during data transmission. All nodes must also have prior knowledge of the network. In contrast, our protocol does not require any prior knowledge of the network.

2.2.1.3 TEEN

Manjeshwar and Agrawal describe a hierarchical routing protocol called TEEN [36] for sensor networks. TEEN is designed to be responsive to sudden changes in observation data. In TEEN, nearby nodes form clusters. After formation of the clusters, the cluster heads broadcast two thresholds, hard and soft, for sensed data. TEEN transmits data only if the value of the data is greater than the hard threshold, or the change in the value of the data is greater than the soft threshold. TEEN uses this approach to reduce the number of transmissions. However, TEEN is not suitable for applications where periodic data transmissions are required, as the root mote will not receive any data if the thresholds are not reached. In contrast to this, our protocol is suitable for applications that require periodic data transmission.

2.2.2 Data-centric Protocols

In many applications of wireless sensor networks, motes are deployed in large numbers with a constantly changing network topology. It is not feasible to assign unique identification numbers to each mote in such large networks, due to constant changes in their topology. Due to the lack of unique identification numbers, it is impossible to query specific sets of motes, and hence the queries must be transmitted to each mote in the network through broadcasts. In response, the sensing motes that satisfy a query send the response data to the querying mote. These queries can be based on attributes, such as the names of objects, time intervals, time durations, and geographical area. Some examples of data-centric protocols include SPIN [23] , Directed Diffusion [19] [27], Rumor Routing [4], Gradient-based Routing [49], Constrained Anisotropic Diffusion Routing [10], COUGAR [61], and ACQUIRE [47].

In contrast to these protocols, our protocol is hierarchical, and every mote in the network has a unique identification number. In our network, each mote can be queried based on this number.

2.2.3 Location Based Routing

In some applications of sensor networks, awareness of a mote's location is important. The location can be used to calculate the distance between motes, or to transfer data in a more efficient manner. Most location-based routing protocols use a low power global positioning system (GPS) [38] [55] to determine location. Minimum Energy Mobile Wireless Networks [46], SMECN [32], and GAF [60] are examples of protocols that use GPS to determine a mote's location. In addition, a mote's location can be calculated using the number of hops from the base station. GEAR [62] and GPSR [28] are examples of protocols that use this method to locate motes. Our protocol calculates its routing level based on the number of hops from the base station; it is not geographically aware.

2.3 Time Synchronization

This section discusses prior work in the development of time synchronization protocols.

2.3.1 FTSP

Maroti et al. present a time synchronization protocol called Flooding Time Synchronization Protocol (FTSP) [37]. FTSP achieves per-hop synchronization with an accuracy of under one microsecond. This high efficiency comes at the cost of frequent transmission of synchronization packets. If there is an application which transmits data every few minutes, most of the network's energy will be spent on time synchronization. Further, since FTSP uses a broadcast mechanism for time synchronization, it can lead to channel congestion, exacerbating the power consumption problem.

2.3.2 RBS

Elson et al. describe a time synchronization protocol called Reference Broadcast Synchronization (RBS) [18]. In RBS time synchronization, the sender transmits the time synchronization packet. This packet does not have a time stamp. After receiving this packet from a sender (at approximately the same instant), receivers exchange their local time to achieve pair-wise time synchronization. This excess exchange of packets can lead to channel congestion.

2.3.3 TPSN

Ganeriwal describe a time synchronization protocol called Timing-sync Protocol for Sensor Networks (TPSN) [20]. TPSN is a two step protocol, consisting of level discovery and synchronization. The level discovery step runs once and forms a hierarchical topology, where each mote is assigned a level. In the synchronization step, a two-way message exchange takes place between each pair of motes. Pair-wise synchronizations are performed along the edge of the hierarchical structure. TPSN assumes that the clock drift between each pair of nodes is constant in the small time period during message exchange. It also assumes that the propagation delay is constant in both directions.

A significant problem in using the FTSP, RBS, and TPSN time synchronization protocols is that each assumes the motes in the network will always be on. This thesis combines the best features of FTSP and TPSN to achieve time synchronization across the network. FTSP transmits synchronization packets very frequently, and has a significant overhead. In contrast, in our time synchronization protocol, the synchronization frequency is relatively less, and the synchronization overhead is negligible.

Chapter 3

Hardware

Hardware plays an important role in making a wireless sensor network low-cost and maintenance free. Designing a custom mote has advantages over using off-the-shelf motes in terms of ease of adding the energy harvesting circuit and the cost.

There are two basic strategies in designing a custom mote. First, the design could use (relatively) high-end integrated circuits, including an energy harvester, boost regulator, buck regulators, etc. The advantage of this approach is that it is easy to use integrated circuits, and the resulting circuits are small. The trade-off of this approach is that integrated circuits are expensive. Second, the design could use basic components, such as resistors, diodes, and capacitors. The advantage of this approach is a reduction in the cost of the device. The basic electronic components are less expensive than integrated circuits, and the resulting Printed Circuit Boards (PCB) are less expensive to produce. The disadvantage of this approach is that the size of the resulting circuit is typically larger. The device described in this thesis offers many of the benefits provided by devices designed using integrated circuits, except for the size. However, given that the size of the circuit is smaller than the size of our target solar panel, the circuit size is not a concern.

The hardware discussed in this thesis is a complete system, ready to be deployed in remote and hostile environments. The device is maintenance free and generates power by harvesting solar energy. The charging circuit is designed using basic electronic components, significantly reducing the cost of the device. The device works in two modes:

1. Solar Powered: The device performs two tasks in solar powered mode. First, it supplies power

to the circuit. Second, it charges the Li-Ion battery.

2. Battery Powered: The circuit operates in battery powered mode whenever there is insufficient solar power. The circuit uses power from the Li-Ion battery in this mode.

The device consists of a switching circuit which switches the circuit from solar powered mode to battery powered mode and vice-versa. The mode switches whenever the output power of the solar panel is less than the required value. The most important use of a switching circuit is to prevent power loss.

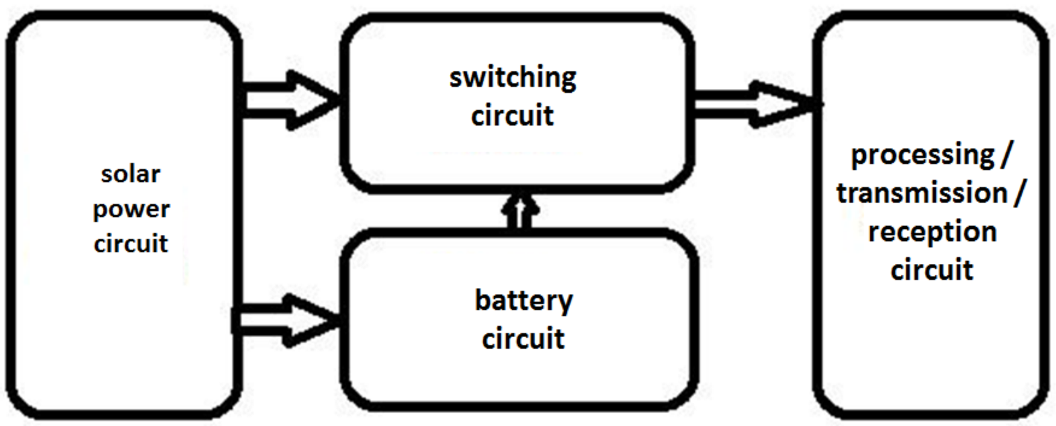

Figure 3.1: Block Diagram of the Mote Platform

The block diagram of the mote platform as shown in Figure 3.1 is divided into four parts:

1. Solar power circuit
2. Battery circuit
3. Switching circuit
4. Circuit for processing, transmission, and reception of data

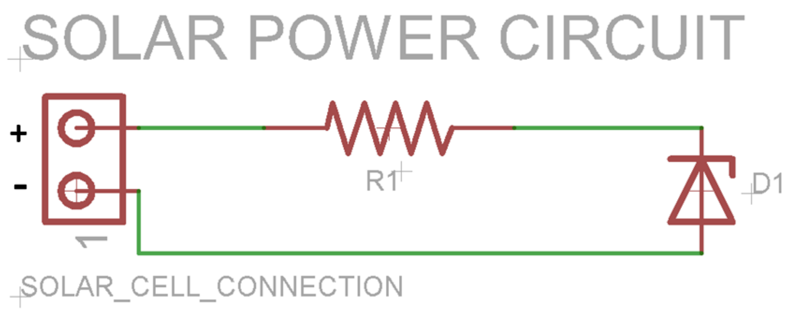

Figure 3.2: Solar Power Circuit

3.1 Solar Power Circuit

As shown in Figure 3.2, the solar power circuit consists of a 5 *volt*, 100 *mA* solar cell, a 100 *ohm* resistor, indicated by R1, and a 3.3 *volt* zener diode, indicated by D1. The zener diode is connected in reverse biased mode as a voltage regulator. The zener diode ensures that the circuit receives a stable supply of 3.3 *volts*. The resistor is used as a current limiting element. It restricts the flow of current so that only a specified maximum amount of current flows through the circuit, and thus prevents the zener diode from being damaged. As shown in Figure 3.2, R1 and D1 form a voltage divider circuit, which ensures that the circuit connected parallel to the zener diode maintains a stable voltage of 3.3 *volts* across it.

The resistance value is based on Ohm's law:

$$R = \frac{V_s - V_z}{I_r} \tag{3.1}$$

where,

R	=	Resistance (*ohms*)
V_s	=	Output voltage of solar cell (*volts*)
V_z	=	Breakdown voltage of zener diode (*volts*)
I_r	=	Amount of current required by circuit (*mA*)

The solar power circuit performs two functions.

1. Provide the voltage and current required by the processing circuit.

2. Provide the excess current to the battery circuit to charge the battery.

3.2 Battery Circuit

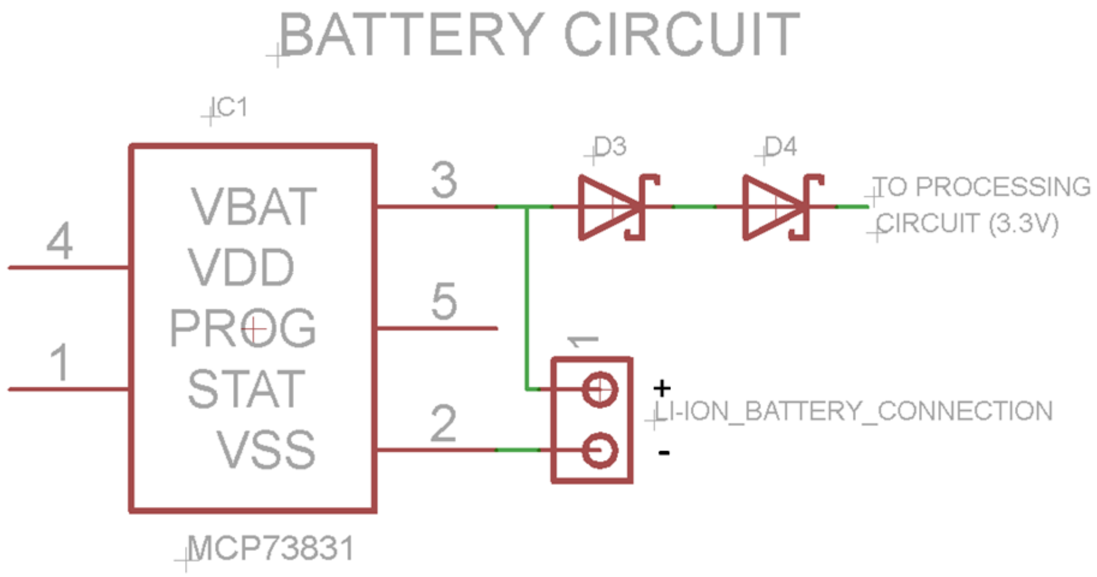

Figure 3.3: Battery Circuit

As shown in Figure 3.3, the battery circuit consists of an MCP73831 integrated circuit, an 850 *mAh* Li-Ion battery, and two Schottky diodes [30], indicated in Figure 3.3 as D3 and D4. The MCP73831 is used to charge the Li-Ion battery. Schottky diodes are usually used for fast switching of inputs, as their response time is faster, and their forward voltage drop is less than normal diodes. The Schottky diode has a low forward voltage drop of only 0.2 *volts* and a high reverse breakdown voltage of 15 *volts*. In this circuit, we use the low forward voltage drop property of the Schottky diode to reduce the voltage. According to the discharge curve of the Li-Ion battery [21], the output voltage of the battery is fairly stable at 3.7 *volts*. The processing circuit operates properly at 3.3 *volts*. This voltage drop is provided by the diodes. As shown in Figure 3.4, if solar energy is not available, voltage at the cathode of diode D4 is less than the voltage at its anode. This makes diode D4 forward biased, and hence the circuit runs on battery power. If solar energy is available, and the output voltage of the battery is less than or equal to 3.7 *volts*, these diodes become reverse biased as voltage at the cathode of the Schottky diode is more than the voltage at the anode, and hence the circuit runs only on solar power while the battery charges. This helps conserve the battery whenever solar energy is available.

The lifetime of a Li-Ion battery is directly proportional to how the battery is charged. There are three charging stages:

1. Precondition charging
2. Constant current charging
3. Constant voltage charging

Depending on the state and capacity of the battery, we must decide on the maximum charging current that will not damage the battery. Li-Ion batteries are very sensitive to charging voltage; the charging voltage should not exceed 1.19% of the rated value. To ensure the charging voltage is within an acceptable range, the circuit uses a MCP73831 [39] integrated circuit instead of basic electronic components. MCP73831 determines the charging stage of the battery. The MCP73831 is also fairly inexpensive (0.69 USD).

3.3 Switching Circuit

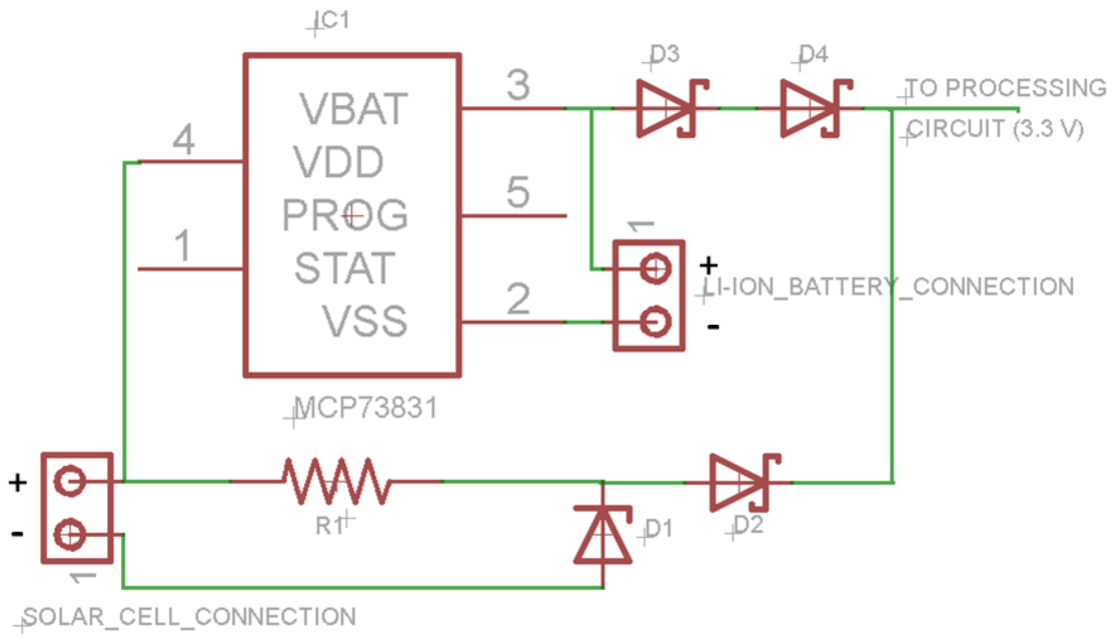

Figure 3.4: Complete Circuit Diagram

Figure 3.5 shows the switching circuit, which consists of a Schottky diode, indicated by D2. The Schottky diode is highly suitable for our application. Consider the situation when it is night;

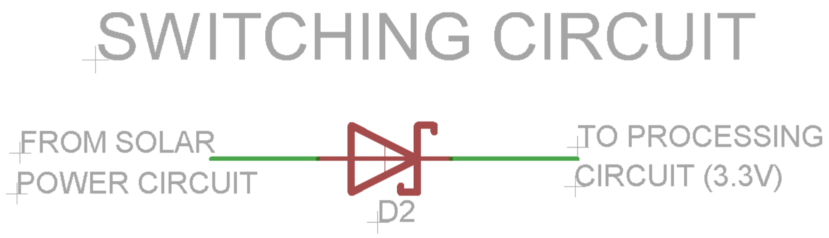

Figure 3.5: Switching Circuit

the solar panel will not source power; instead, it will sink power. If the Schottky diode shown in Figure 3.4 as D2 is not present, the zener diode denoted as D1 will connect to the battery and it will continuously consume power from the battery, resulting in battery failure. In this situation, the Schottky diode D2 becomes reverse biased, resulting in high reverse breakdown voltage, causing the path between the battery circuit and solar power circuit to become (effectively) an open circuit, resulting in savings in battery power.

3.4 Circuit for Processing, Transmission and Reception of Data

This section describes the processing and transmission circuit, and the connection of the ***root*** mote with the *base station*. The processing circuit consists of an ATMEGA168 microcontroller [3], and a radio frequency transceiver, the RFM12 [17]. The RFM12 transmits and receives data. It offers many advantages over more costly radios, such as the CC2420, the CC2400, and others, including programmable variable data transmission rates, variable channel frequency, and most importantly, low power consumption and cost. The ATMEGA168 is used to sense and process data *in situ*. The ATMEGA168 transmits the processed data using the RFM12 transceiver to the neighboring motes. If the mote is the ***root***, it transmits the data to the base station using an FT232R [9] chip, a UART to USB converter. The base station runs Python code that processes, analyzes, and stores the data.

The main motivation for creating this design and not using off-the-shelf hardware, such as the TelosB mote [40], is the hardware cost and the ease of adding the harvesting circuit. This thesis focuses on making an inexpensive network that provides all the basic functionality of off-the-shelf motes. Table 3.1 shows the cost of the individual components used to make the device, as well as the costs when the components are purchased in volume.

	Quantity		
Components	**1**	**100**	**1000**
Solar Panel	6.90	5.50	5.50
Li-Ion battery	11.61	3.87	3.22
RFM12	6.95	5.56	5.56
ATMEGA168	2.43	1.35	1.35
MCP73831	0.68	0.42	0.42
Resistor, Capacitor, and Diodes	2.97	1.59	0.88
Total	31.54	18.29	16.93
Cost of 1 unit	31.54		
Cost of 100 units		1829.00	
Cost of 1000 units			16930.00

Table 3.1: Cost of Components (in USD) [11]

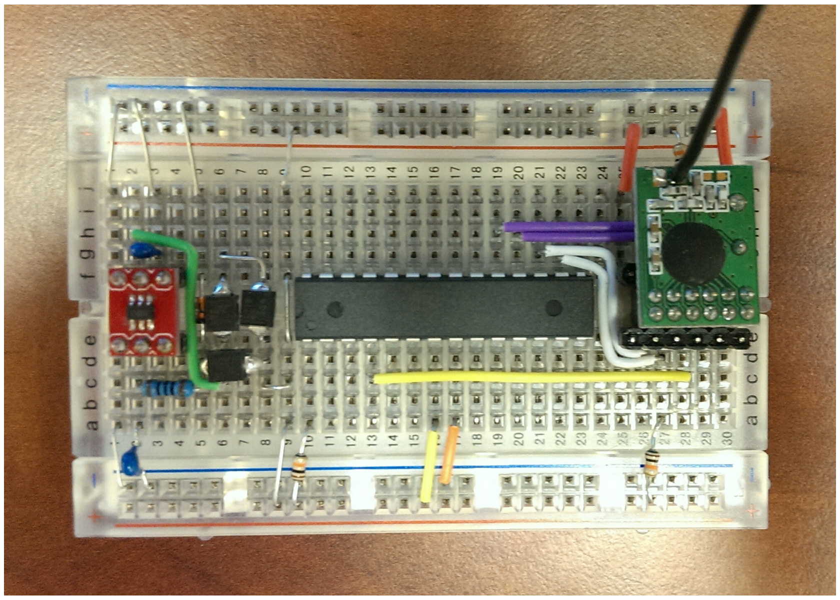

Figure 3.6: Prototype Board

The prototype of the board is shown in Figure 3.6. The prototype board consists of a MCP73831 Li-Ion battery charger, Schottky diodes, an ATMEGA168 microcontroller to process data, and an RFM12 transceiver to transmit and receive data from other motes. In the future, the prototype board can be converted to a PCB to further reduce the size of the circuit. This circuit can be built on a PCB that is 1 square inch.

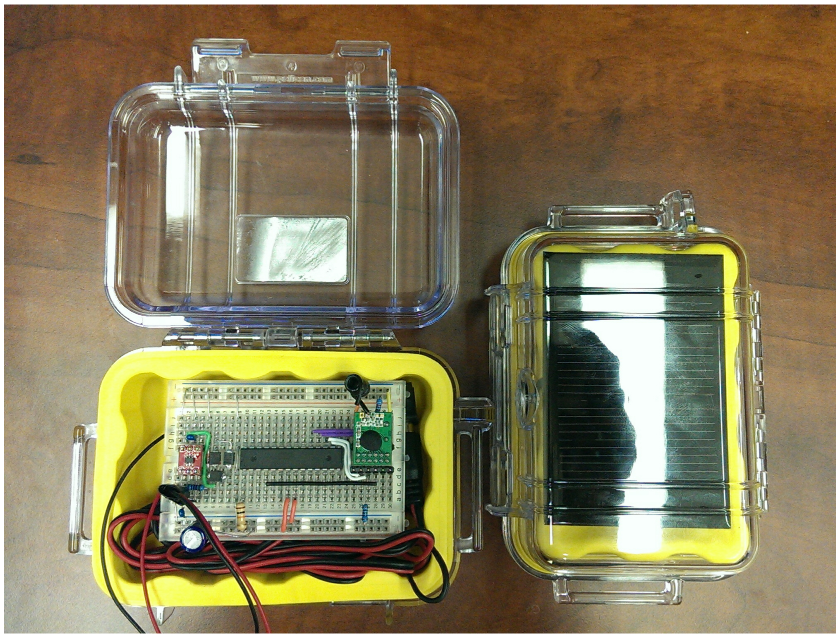
Figure 3.7: Safety Enclosure

As shown in Figure 3.7, the circuit is then enclosed in a plastic container to prevent it from being damaged by environmental factors like rain or dust. Figure 3.7 shows the prototype board and the solar panel enclosed in the plastic container.

Chapter 4

Software

Wireless sensor networks must be reliable. If a network mote fails, motes connected to the failed device must find another path to reliably transmit data to a base station. At the same time, wireless sensor networks must be scalable. A sensor network should be able to handle the addition of any number of new devices. The software described in this thesis supports reliable and scalable wireless sensor networks. We rely on the following terms throughout the exposition :

1. *base station* : A *base station* is a field deployed server machine which receives, processes, analyzes, and stores the data received from the motes.

2. *level* : The *level* of a mote is the distance (in hops) of the mote from a *base station*.

3. *parent* : The *parent* of a mote A is the mote to which A is connected. Its *level* is one *level* less than that of A.

4. *child* : A *child* of a mote B is the mote connected to B, with B serving as its *parent*. Its *level* is one *level* more than that of A.

4.1 Mote States

In our system, every mote in the network is in one of the following four states:

1. ***root***

   The ***root*** state indicates that the current mote is directly connected to a base station. All the messages received by this mote are forwarded directly to the base station. All other motes in the network are synchronized with the local clock of this mote. The ***root*** mote periodically transmits a ***sync*** signal to its children, and in response, the children transmit data to the ***root***.

2. ***idle***:

   The ***idle*** state indicates that the mote is not connected to any other motes; it is searching for a *parent*. All the motes except the ***root*** mote are in the ***idle*** state at boot. ***idle*** motes periodically send a beacon and try to connect to a responding mote.

3. ***leaf***:

   The ***leaf*** state indicates that the mote is now connected to a *parent*. It also indicates that the mote can transmit its data to the base station through its *parent*. The mote enters the ***leaf*** state from the ***idle*** state.

4. ***node***:

   The ***node*** state indicates that the mote is connected to a *parent* and other motes are connected to it as children. In this state, the mote will periodically send a ***sync*** signal to its children and receive sensor data in response. Each ***node*** will transmit the received data along with its own data to its *parent*, which will in turn continue to relay data until it reaches the base station.

4.2 Software Services

The software described in this thesis provides four major functions : time synchronization, transmission slot management, parent search, and data transmission. We describe the design of these facilities in the following subsections.

4.2.1 Time Synchronization

Time synchronization is an important factor in wireless sensor networks. In typical applications, motes do important work only for short periods; most of the time they are idle. The majority of the total power consumption stems from the radio. To conserve power, the radio must be turned off and switched on only when required. If a transmitting mote and the corresponding receiver are not synchronized, the data will be lost. This problem is exacerbated if the network is multi-hop. To achieve reliable multi-hop communication, every mote must be time synchronized and must know when to transmit and receive data so there is no packet loss.

To achieve time synchronization, every mote maintains a local clock. We use the internal Timer2 of the ATMEGA168 microcontroller. Timer2 is programmed to generate an interrupt every millisecond. The interrupt handler increments the count of the local clock by one, creating a millisecond timer. Timer2 computes this time based on the cycles generated by the internal resistor-capacitor (RC) oscillator. Unfortunately, in practical situations, the generated clock is not perfect; its accuracy depends on factors including temperature and humidity. Due to these dependencies, the error rate of the RC oscillator is quite high: $\pm$ 10% [3]. To overcome this, motes are synchronized frequently.

Commonly used time synchronization protocols in wireless sensor networks include Flooding Time Synchronization Protocol (FTSP), Reference Broadcast Synchronization (RBS), and Timing-sync Protocol for Sensor Networks (TPSN).

A significant problem in using the FTSP, RBS, and TPSN time synchronization protocols is that each assumes the motes in the network will always be on. This thesis combines the best features of FTSP and TPSN to achieve time synchronization across the network. In this design, the network is hierarchical. Each parent in the routing tree transmits a synchronization (SYNC) packet only when checking whether its child has data to transmit. If data is transmitted every minute, the parent will send a SYNC packet every minute. As soon as a child receives a SYNC packet, it sets its clock based on the clock of its parent, and then transmits its data packet to its parent. Instead of concentrating on network-wide synchronization, this approach focuses on synchronization between parents and children, which indirectly synchronizes the whole network. This in turn eliminates the need for unnecessary broadcast packets, which saves power and reduces the likelihood of network congestion.

Another problem in implementing time synchronization is clock drift. We know that the error rate of the RC oscillator is $\pm$ 10%, causing drift in the local clock of every mote. To minimize clock drift, each clock maintains a variable that contains the measured drift between it and its parent. The transmission interval between synchronization packets is the same throughout the network. Further, holding external factors constant, the drift in clock time between a pair of motes will always be the same between two consecutive synchronization packets. We calculate the clock drift by subtracting the value of the local clock of the child with the local clock of the parent. To minimize clock drift, each time a synchronization packet is received, the difference is added to the local clock of the parent and stored in the local clock of the child.

4.2.2 Transmission Slots

Wireless sensor networks are prone to channel congestion, leading to packet loss and excessive power consumption. To reduce channel congestion, this thesis makes use of synchronized transmission slots. Each parent assigns transmission slots to its children. A parent and child can communicate only during this time. The parent cannot assign the same transmission slot to multiple children.

The duration of a transmission slot is calculated based on the time required to transmit a packet with a maximum payload. Specifically, it is twenty times the time required to transmit a packet with a maximum payload. In the worst case, the transmitter can resend data nineteen times (assuming no delay), which in most cases is enough for the data to reach the receiver.

$$T_{ts} = 20 * T_{mp} \tag{4.1}$$

where,

$$\begin{aligned} T_{ts} &= \text{Time duration of a transmission slot} \\ T_{mp} &= \text{Time required to transmit a packet with maximum payload} \end{aligned}$$

$$T_{mp} = \frac{8 * (maximum\ size\ of\ payload\ (in\ bytes)\ +\ packet\ overhead\ (in\ bytes))}{baudrate\ of\ radio\ (in\ bits\ per\ sec)} + backoff\ delay \tag{4.2}$$

$$T_{mp} = \frac{8 * (255\ +\ 10)}{57600}\ +\ 3 * \frac{8 * (255\ +\ 10)}{57600} = 147.24\ millisecond$$

$$T_{ts} = 20 * 147.24 = 2944.8\ millisecond = 2.9448\ second$$

Transmission slots reduce the power consumption of each mote. A mote is powered up only during its assigned transmission slots (to communicate with its parent and children); otherwise, it is sleeping.

4.2.3 Parent Search

This section discusses the steps taken by a mote to join the network.

As shown in Figure 4.1, mote A is in the ***idle*** state, searching for a parent. Mote B is in the

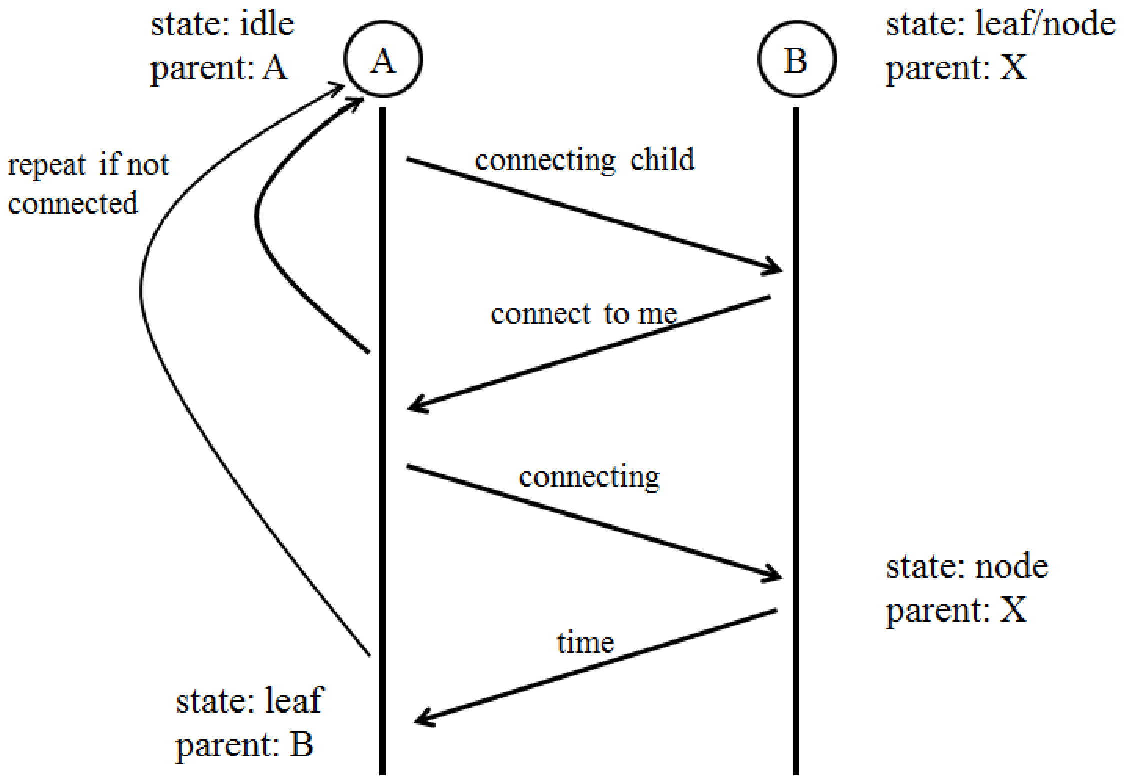

Figure 4.1: Finding a Parent Mote in the Network

leaf or the ***node*** state. Mote B is connected to the tree, and mote X is the parent of mote B. In the ***idle*** state, mote A broadcasts CONNECTING CHILD with period t. If there is another mote in the network which is time synchronized and has a transmission slot available, say B, it receives this message and transmits the CONNECT TO ME signal to mote A after some random time $\bar{t}$, where ($\bar{t} << t$). When mote A receives the CONNECT TO ME signal from mote B, it transmits the CONNECTING signal to mote B. As soon as mote B receives the CONNECTING signal, it searches for an available transmission slot and assigns that transmission slot to mote A. It sends the TIME signal and sets its state to the ***node*** state. Mote B then sets mote A as its child. As soon as mote A receives the TIME signal, it sets its local time to that of mote B, sets its state to ***leaf***, and sets mote B as its parent.

4.2.4 Data Transmission

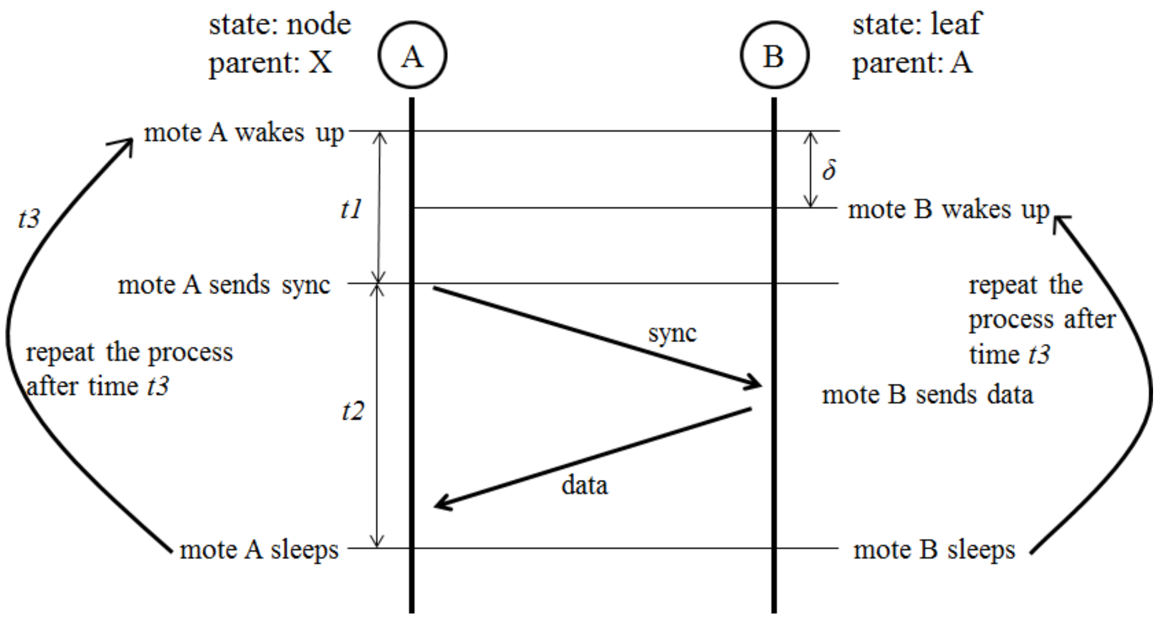

Figure 4.2: Sending the Sync Signal and Receiving Data

The main task of any network is to enable communication. This section discusses how data transmission takes place in our sensing system.

In Figure 4.2, mote A is in the ***node*** state, and mote B is in the ***leaf*** state. Mote A is the parent of mote B. When mote A wakes, it transmits the SYNC signal to its child, say B, after a delay of time $t1$. Mote A waits for time $t1$ to make sure that mote B is on; there might be a delay in waking mote B due to clock drift. Next, mote A waits up to the duration of a transmission slot, say $t2$, to receive DATA from its child. If mote A has multiple children, the process of sending the SYNC signal is repeated multiple times. Finally, after receiving DATA from all its children, mote A sleeps for time $t3$, and the above process is repeated.

As shown in Figure 4.2, mote B wakes up at time δ. As soon as mote B receives the SYNC signal from its parent, it waits for a random amount of time before transmitting to avoid a collision. It then transmits the DATA to its parent mote. Next, mote B sleeps for time $t3$. This process is then repeated.

4.2.4.1 Heartbeat Mechanism

Data transmissions also help to detect motes which have failed. Data packets serve as the heartbeat of the network. When a mote fails, other motes detect this failure to maintain the overall health of the network. Network healing is one of the most important factors that makes data transmission in a network reliable.

When a parent fails, a child mote must detect it. To detect the death of a parent, every child maintains a variable, $parent\_life$. Whenever a child receives a SYNC signal from its parent, it sets $parent\_life$ to a predefined value. When a child wakes up from sleep, it decrements $parent\_life$ by one. If $parent\_life$ becomes zero, the child declares its parent dead and transmits the PARENT DYING signal to its children. Next, the child begins searching for another parent.

When a child mote fails, the parent mote must detect it. To detect the death of a child, say, $child_i$, every parent maintains a variable, $child\_life_i$, specific to that child. When $child_i$ gets connected to a parent, the parent sets $child\_life_i$ to a predefined value. Whenever a parent transmits a SYNC signal to $child_i$, it decrements $child\_life_i$ by one. If the parent receives DATA from $child_i$, it again sets $child\_life_i$ to its predefined value. If $child\_life_i$ becomes zero, the parent declares $child_i$ dead. It then frees the transmission slot of that child and makes it available for a new child.

4.3 Packet Formats

This section discusses the packet formats used by the software for communication. The packets used for communication include (i) CONNECTING CHILD, (ii) CONNECT TO ME, (iii) CONNECTING, (iv) TIME, (v) SYNC, (vi) DATA, and (vii) PARENT DYING. Each of these packets uses the packet format shown in Table 4.1.

destination address (1 byte)	***source address*** (1 byte)	***flag*** (1 byte)	***length*** (1 byte)	***payload*** (255 bytes)	***crc*** (1 byte)

destination address	The address of the destination mote
source address	The address of the transmitting mote
flag	Indicates whether the current packet is a data packet or an acknowledgment
length	Contains the size of the payload
payload	Contains the message
crc	Contains the cyclic redundancy check (CRC-8) computed over the payload

Table 4.1: Packet Format

4.3.1 Payload Format

This section summarizes the payload formats of the packets used by the software.

1. CONNECTING CHILD: The ***payload*** format of a CONNECTING CHILD message is as follows:

CONNECTING CHILD (1 byte)

   Here, the ***CONNECTING CHILD*** field contains the CONNECTING CHILD token.

2. CONNECT TO ME: The ***payload*** format of a CONNECT TO ME message is as follows:

CONNECT TO ME (1 byte)	***level*** (4 bytes)

   Here, the ***CONNECT TO ME*** field contains the CONNECT TO ME token. The ***level*** field contains the *level* of the transmitting mote.

3. CONNECTING: The ***payload*** format of a CONNECTING message is as follows:

CONNECTING (1 byte)

   Here, the ***CONNECTING*** field contains the CONNECTING token.

4. TIME: The ***payload*** format of a TIME message is as follows:

TIME (1 byte)	***transmission slot*** (4 bytes)

   Here, the ***TIME*** field contains the TIME token. The ***transmission slot*** field contains the time slot of the receiving mote.

5. SYNC: The ***payload*** format of a SYNC message is as follows:

SYNC (1 byte)	***local time*** (4 bytes)

   Here, the ***SYNC*** field contains the SYNC token. The ***local time*** field contains the local time of the transmitting mote.

6. DATA: The ***payload*** format of a DATA message is as follows:

DATA (1 byte)	***source address*** (1 byte)	***parent address*** (1 byte)	***message number*** (4 bytes)

   Here, the ***DATA*** field contains the DATA token. The ***source address*** field contains the address of the transmitting mote. The ***parent address*** field contains the address of the transmitting mote's parent. The ***message number*** field contains the message number of the packet being transmitted by the transmitting mote. In real applications, this data packet would be extended to include the sensor's control data.

7. PARENT DYING: The ***payload*** format of a PARENT DYING message is as follows:

PARENT DYING (1 byte)

   Here, the ***PARENT DYING*** field contains the PARENT DYING token.

Chapter 5

Evaluation

In this chapter, we evaluate our system on the basis of reliability, scalability, and network longevity.

5.1 Reliability

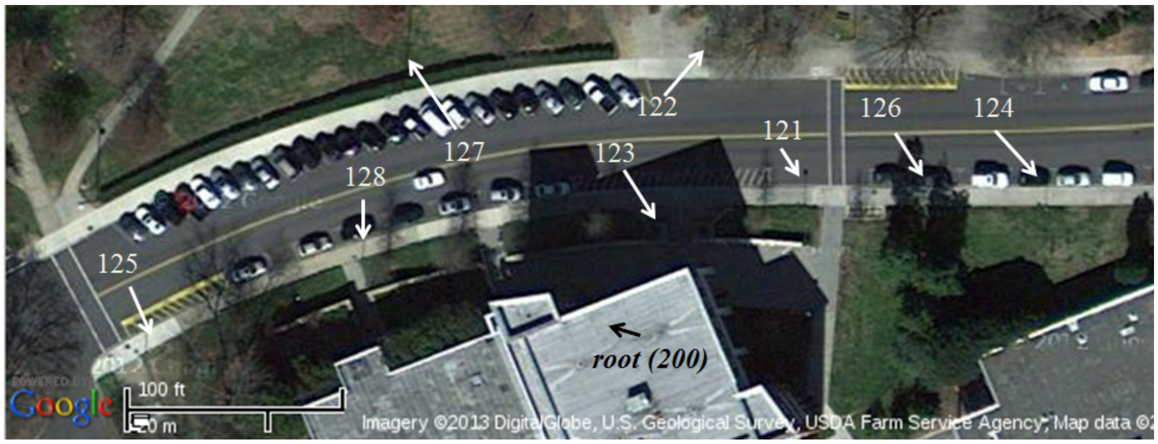
122
126
124
121
127
123
128
125
root (200)
100 ft
20 m
Google
Imagery ©2013 DigitalGlobe, U.S. Geological Survey, USDA Farm Service Agency, Map data ©2

Figure 5.1: Mote Deployment

In this section, we evaluate the reliability of the network based on two parameters: the percentage error in packet reception, and the ability of the network to tolerate mote failure. To evaluate these parameters we deployed eight motes on the lamp posts in front of Clemson University's School of Computing. The ***root*** mote was deployed inside the School of Computing, as shown in Figure 5.1. The ***root*** mote was connected to a desktop server process and periodically received data

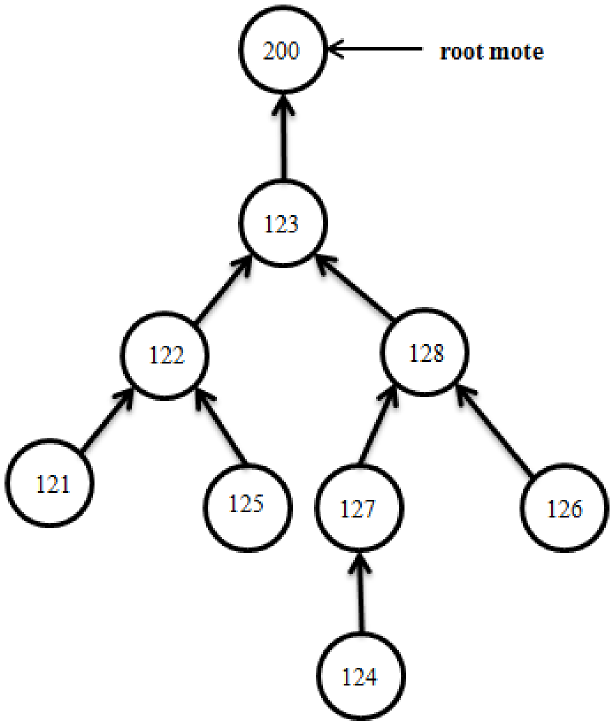

Figure 5.2: Observed Routing Topology

from the deployed motes. Figure 5.2 shows the observed routing topology formed by the deployed motes.

5.1.1 Percentage Error in Packet Reception

In this subsection, we evaluate the percentage error in packet reception for each mote in the network. The percentage error in packet reception is defined as the ratio of the difference between the number of packets transmitted by the mote and the number of packets received at the base station, to the number of packets transmitted. That is, we consider end-to-end packet reception. To evaluate the percentage error in packet reception, we allowed the deployed motes to run continuously for 7 days. Each data packet sent by a mote to its parent contained a count of the total number of tries it took to successfully send the message, as well as the message number of the data being transferred. Table 5.1 shows the total number of packets transmitted, the total number of packets received, the percentage error in packet reception, and the total number of tries before successful transmission of the packets over a period of 7 days.

As we can see from Table 5.1, the percentage error in packet reception is 0. The packets

Mote	**Count of packets transmitted**	**Count of packets received**	**Percentage error in packet reception**	**Total number of retries**
121	8714	8714	0	122
122	9047	9047	0	344
123	10152	10152	0	4
124	8129	8129	0	161
125	9282	9282	0	165
126	6240	6240	0	184
127	9182	9182	0	64
128	5195	5195	0	389
Total	65941	65941	0	1433

Table 5.1: Data Transmission Metrics

transmitted by each mote are received at the base station. That is, the network achieved 100% data yield. The number of retries before successful transmission of a data packet is less than 3% of the total packets transmitted by all motes.

5.1.2 Ability to Tolerate Mote Failure

In this subsection, we evaluate the ability of the network to withstand mote failures. To evaluate this property of the network, we followed the same experimental setup described in sections 5.1 and 5.1.1. Figure 5.2 shows the observed routing topology formed by the deployed motes. To evaluate the ability of the network to withstand mote failure, we introduced a fault by deactivating mote 128. Due to this, motes 127, 126, and 124 were disconnected from the network. We observed that motes 127, 126, and 124 detected the failure of mote 128, and rejoined the network, choosing another parent.

Regular (i.e., real) failure cases were also considered. Here, failure is defined as the removal of the mote from the network due to an obstruction between motes, an out of range mote, power failure, or a mote in the process of resetting. To determine the failure of the mote, we monitor the data received at the base station and timestamp the data. If the data is not received from a particular mote for 5 minutes then the mote is declared as failed. After that, whenever data is received from the failed mote, we conclude that the mote rejoined the network. Table 5.2 shows, over a period of a day, the number of mote failures recorded, the number of rejoins that occurred within 10 minutes, and the number of failures that persisted beyond 10 minutes. Table 5.3 shows, for each mote, over a period of 7 days, the number of failures the mote experienced, the number of times the mote was able to rejoin the network after a failure, and the number of times the mote was unable to rejoin the network after a failure. These figures suggest that the network is able to withstand mote failure.

Day	**Number of mote failures**	**Number of failures corrected within 10 minutes**	**Failures not corrected within 10 minutes**
June 20	4	4	0
June 21	48	48	0
June 22	56	55	1
June 23	42	40	2
June 24	61	61	0
June 26	29	29	0
June 27	18	18	0

Table 5.2: Mote Failure Statistics (per day)

Mote	**Number of failures**	**Number of failures corrected within 10 minutes**	**Failures not corrected within 10 minutes**
121	7	7	0
122	9	9	0
123	0	0	0
124	9	9	0
125	9	9	0
126	38	38	0
127	5	5	0
128	181	178	3

Table 5.3: Mote Failure Statistics (per mote)

Table 5.2 shows that one mote was unable to recover on June 22, and two motes were unable to recover on June 23. We can see from Table 5.3 that the only mote that was unable to recover is mote 128. The failure of mote 128 was purposefully introduced by disconnecting the battery. We can see from Tables 5.2 and 5.3 that every other mote eventually recovered and rejoined the network within 10 minutes after failing. This supports our claims that the network is able to withstand mote failures.

The results presented in sections 5.1.1 and 5.1.2 support our claim that the network is reliable.

5.2 Scalability

This section evaluates the scalability of our system. In this context, scalability depends on the internal message buffer size, the number of transmission slots per mote, and the baud-rate of the transceiver.

Internal buffer size is an important factor in the evaluation of network scalability. In the experiments discussed in the previous sections, the size of the internal message buffer is 255 bytes, and the size of a data packet is 7 bytes. The maximum number of motes allowed in each transmission slot of the ***root*** mote in the network is equal to the number of message packets that can be accommodated in the internal message buffer. It is assumed that each mote transmits exactly one data packet.

$$number\ of\ motes \ = \ \frac{255}{7} \ = \ 36.43$$

Since the *root* mote immediately transmits all received data to the base station, the network can accommodate 36 motes *per child* of the *root* mote, assuming an internal buffer size of 255 bytes per mote and a message packet size of 7 bytes.

The total number of motes that can be accommodated in a network, depends on the number of transmission slots per mote. In our system, the duration of each transmission slot is 3 seconds (calculated in Section 4.2.2), and data is transmitted every 60 seconds. Therefore, the total number of transmission slots that can be present in each period is $\frac{60\ (second)}{3\ (second)} \ = \ 20\ slots$. The total number of motes that can be accommodated in our system is as follows:

$$T_{motes} \ = N_{slots} * N_{motes} \tag{5.1}$$

where,

T_{motes}	=	the maximum number of motes that can be accommodated in the network
N_{slots}	=	the maximum number of transmission slots in a transmission period
N_{motes}	=	the maximum number of motes in each transmission slot of the ***root*** mote

$$T_{motes} \ = \ 20 \ * \ 36 \ = \ 720\ motes.$$

The baud-rate of the transceiver is also an important factor in the scalability of the network.

The baud-rate of the RFM12 transceiver is 57.6 Kbps. The number of bytes that can be transmitted in one transmission slot, i.e. 3 seconds, is as follows:

$$number\ of\ bytes\ =\ \frac{3 * 57600}{8}\ =\ 21600$$

Now we calculate the maximum number of bytes that can be transmitted in 3 seconds. The equation is as follows:

$$T_{max} = N_{motes} * D_{bytes} \tag{5.2}$$

where,

T_{max}	=	the maximum number of bytes transmitted during each transmission
N_{motes}	=	the maximum number of motes in each transmission slot of the ***root*** mote
D_{bytes}	=	the size of a data message, including packet overhead

$$T_{max}\ =\ 36\ *\ 12\ =\ 432\ bytes$$

In our system, even if each mote sends its data 20 times, we transmit a maximum of 8640 bytes (432 * 20) in 3 seconds per mote. The radio can easily handle the transmitted data.

The calculations for internal buffer size, number of transmission slots per mote, and baud-rate of the transceiver together show that given the buffer size of 255 bytes and a message packet size of 7 bytes, this network can scale to accommodate 720 motes. To increase the number of motes beyond 720, we must increase the internal message buffer size.

5.3 Network Longevity

This section evaluates the longevity of the network. Here, network longevity is defined as the time period for which the network will work. The circuit operates both in the absence and presence of sunlight. We consider the lifetime of network in both cases.

5.3.1 Absence of Sunlight

This subsection considers the lifetime of the network running only on battery power, in the absence of sunlight. Table 5.4 shows the power consumption of the microcontroller and the RFM12 transceiver in the active and sleep states.

	Micro-controller	**RFM12**	**Other components**	**Total**
active	4.00	8.04	0.96	13.00
sleep	0.32	0.10	0.20	0.62

Table 5.4: Device Current Consumption (*milliamps*)

Our system works at a 10% duty cycle; the motes transmit data every 60 seconds. Within a 60 second timeframe, the motes are in the active state for 6 seconds, and in the sleep state for 54 seconds (assumed each mote has maximum 2 children). The current consumption is calculated using the following formula:

$$C = \frac{A * on\ time + S * off\ time}{total\ time} \tag{5.3}$$

where,

C	=	Current consumption in *milliamps*
A	=	Active state current consumption in *milliamps*
S	=	Sleep state current consumption in *milliamps*

$$C = \frac{6 * 13.00 + 54 * 0.62}{60} = 1.858\ milliamps$$

The lifetime of a mote is calculated using the following formula:

$$Lifetime = \frac{Battery\ Capacity * 0.80}{Current\ Consumption} \tag{5.4}$$

The 0.80 in the equation indicates that we can use up to 80% of the battery to run the circuit.

$$Lifetime = \frac{850 * 0.80}{1.858} = 365.98\ hrs = 15.25\ days$$

The above calculation indicates that the network can run on battery power, in the absence of sunlight, for a period of 15.25 days.

5.3.2 Presence of Sunlight

This subsection evaluates the lifetime of the network running in the presence of sunlight. The solar panel generates 89 milliamps of current, and a voltage of 5 volts. Of the 89 milliamps of current, the solar panel supplies 17 milliamps of current to the processing circuit, and it supplies 50 milliamps of current to charge the Li-Ion battery. The remaining 12 milliamps of current is not used.

As calculated in equation 5.3, the mote consumes 1.858 milliamps of current. Let us assume that sunlight is available for 6 hours each day. As stated above, the solar panel will charge the battery by 6 * 50 = 300 mAh (best case charging). Each day, circuit runs on the battery for 18 hours. The power consumption from the battery is 18 * 1.858 = 33.44 mAh. As we can see, energy consumed is just 11.148 % of the energy stored each day.

The above calculations in subsections 5.3.1 and 5.3.2 support our claim that the network will function properly for long periods without changing the batteries.

Chapter 6

Conclusion

Wireless sensor networks are used to gather meaningful data and enable important applications. They are important tools for monitoring the physical world. However, to be effective, they must satisfy some basic conditions. Specifically, they must be *maintenance free*, *inexpensive*, *reliable*, and *scalable.* The devices used in these networks are deployed in numbers ranging from the hundreds to thousands, and they often rely on battery power. Replacing the batteries in such large networks is expensive, both in terms of battery cost and personnel time. The cost of an individual device also plays an important role in the cost of a sensor network. Further, if these networks are deployed in safety-critical contexts, the risk of missing data or collecting incorrect data must be minimized. Finally, a sensor network must be able to accommodate any number of devices and gracefully handle device failure.

To address these issues, our solution uses a multi-faceted approach. First, to minimize the cost of maintaining a network, we design a maintenance free mote. To accomplish this, our sensing platform includes an energy harvesting circuit which harvests solar energy and stores it in a Li-Ion battery. This stored energy is used whenever there is insufficient solar power. Further, we use custom software to put the controller in deep sleep state whenever possible to save energy. Second, we focus on a low cost mote designed to sense the physical and chemical parameters in a micro-environment. We use components which are low in cost, but sufficient for a simplified mote design. Instead of using common, expensive radios, we use the RFM12, which offers high data-rates, variable transmission power, multi-channel operation, and long range, at a low cost. The RFM12 does not require any external components to transmit or receive data, which further simplifies our design.

Third, we develop a network design to transmit and receive data reliably. To achieve this, we design a protocol based on a lightweight TDMA strategy with packet acknowledgments to ensure that every packet reaches the base station. Fourth, we focus on a scalable solution which accommodates the addition of new devices and handles mote failures. In addition, we also develop a lightweight time synchronization protocol and a route formation protocol. We then evaluate our solution on the basis of reliability, scalability, and network longevity.

This thesis describes a low-cost, maintenance-free sensor networking platform, supported by lightweight, yet reliable and scalable networking software. The hardware/software solution exhibits four key characteristics. First, we present a maintenance free solution. We accomplish this by harvesting solar energy and by efficiently using the available energy. Second, we present an inexpensive solution. We accomplish this by designing a mote platform which uses basic electronic components, such as resistors, capacitors, and diodes instead of using more costly integrated circuits. Third, we present a reliable solution that ensures high yield, even in the presence of intermittent and permanent device faults. We accomplish this by developing lightweight route formation, time synchronization, and TDMA protocols; the latter includes application-level acknowledgments to ensure reliable data transmission and reception. Finally, we present a scalable solution that accommodates the dynamic addition and removal of motes without affecting network reliability. We believe that this solution will help users construct maintenance free, inexpensive, reliable, and scalable networks.

Bibliography

[1] Junaid Ansari, Dmitry Pankin, and Petri Mähönen. Radio-triggered wake-ups with addressing capabilities for extremely low power sensor network applications. *International Journal of Wireless Information Networks*, 16(3):118–130, 2009.

[2] Atmel Corporation. 8-bit atmel microcontroller with 128kbytes in-system programmable flash. `http://www.atmel.com/Images/doc2467.pdf`, June 2011.

[3] Atmel Corporation. 8-bit atmel microcontroller with 4/8/16k bytes in-system programmable flash. `http://www.atmel.com/images/doc2545.pdf`, May 2011.

[4] David Braginsky and Deborah Estrin. Rumor routing algorthim for sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pages 22–31, New York, NY, USA, 2002. ACM.

[5] Michael Buettner, Richa Prasad, Alanson Sample, Daniel Yeager, Ben Greenstein, Joshua R. Smith, and David Wetherall. Rfid sensor networks with the intel wisp. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 393–394, New York, NY, USA, 2008. ACM.

[6] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer. Field testing a wireless sensor network for reactive environmental monitoring [soil moisture measurement]. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, pages 7–12, 2004.

[7] Tom S Chan. Time-division multiple access. *Handbook of Computer Networks: LANs, MANs, WANs, the Internet, and Global, Cellular, and Wireless Networks, Volume 2*, pages 769–778.

[8] K. Chintalapudi, T. Fu, Jeongyeup Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, and S. Masri. Monitoring civil structures with a wireless sensor network. *Internet Computing, IEEE*, 10(2):26–34, 2006.

[9] FTDI chip. Ft232r usb uart ic. `http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf`, March 2012.

[10] Maurice Chu, Horst Haussecker, and Feng Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *International Journal of High Performance Computing Applications*, 16(3):293–313, 2002.

[11] Digi-Key CORPORATION. Costs. `http://www.digikey.com`, March 2013 (Last Accessed).

[12] Crossbow Technology Incorporated. Mica2. `http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`, March 2013 (Last Accessed).

[13] Crossbow Technology Incorporated. Mica2dot. https://www.eol.ucar.edu/rtf/facilities/isa/internal/CrossBow/DataSheets/mica2dot.pdf, March 2013 (Last Accessed).

[14] Crossbow Technology Incorporated. Micaz. http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf, March 2013 (Last Accessed).

[15] David Culler, Jason Hill, Mike Horton, Kris Pister, Robert Szewczyk, and A Woo. Mica: The commercialization of microsensor motes. *Sensors Magazine*, 2002.

[16] Henri Dubois-Ferrière, Laurent Fabre, Roger Meier, and Pierre Metrailler. Tinynode: a comprehensive platform for wireless sensor network applications. In *Proceedings of the 5th international conference on Information processing in sensor networks*, IPSN '06, pages 358–365, New York, NY, USA, 2006. ACM.

[17] HOPERF ELECTRONIC. Rfm12 universal ism band fsk transceiver. http://www.hoperf.com/upload/rf/RFM12.pdf, 2006.

[18] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, December 2002.

[19] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: scalable coordination in sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 263–270, New York, NY, USA, 1999. ACM.

[20] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 138–149, New York, NY, USA, 2003. ACM.

[21] Ltd Guangzhou Markyn Battery Co. Polymer lithium battery data sheet. http://www.powerstream.com/lip/GM063048.pdf, March 2013 (Last Accessed).

[22] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2000.

[23] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '99, pages 174–185, New York, NY, USA, 1999. ACM.

[24] Texas Instruments. cc2400. http://www.ti.com/lit/ds/symlink/cc2400.pdf, March 2006.

[25] Texas Instruments. Msp430f15x, msp430f16x, msp430f161x mixed signal microcontroller. http://www.ti.com/lit/ds/symlink/msp430f1611.pdf, March 2011.

[26] Texas Instruments. cc2420. http://www.ti.com/lit/ds/symlink/cc2420.pdf, February 2013.

[27] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 56–67, New York, NY, USA, 2000. ACM.

[28] Brad Karp and Hsiang-Tsung Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM, 2000.

[29] J.D. Kenney, D.R. Poole, G.C. Willden, B.A. Abbott, A.P. Morris, R.N. McGinnis, and D.A. Ferrill. Precise positioning with wireless sensor nodes: Monitoring natural hazards in all terrains. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 722–727, 2009.

[30] M. A. Laughton and D.F. Warne. Schottky barrier diodes. `http://books.google.com/books?id=5jOblzV5eZ8C&pg=SA17-PA25#v=onepage&q&f=false`, 2003.

[31] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. In Werner Weber, JanM. Rabaey, and Emile Aarts, editors, *Ambient Intelligence*, pages 115–148. Springer Berlin Heidelberg, 2005.

[32] Li Li and J.Y. Halpern. Minimum-energy mobile wireless networks revisited. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 1, pages 278–283 vol.1, 2001.

[33] Kris Lin, Jennifer Yu, Jason Hsu, Sadaf Zahedi, David Lee, Jonathan Friedman, Aman Kansal, Vijay Raghunathan, and Mani Srivastava. Heliomote: enabling long-lived sensor networks through solar energy harvesting. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys '05, pages 309–309, New York, NY, USA, 2005. ACM.

[34] Stephanie Lindsey and Cauligi S Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *Aerospace conference proceedings, 2002. IEEE*, volume 3, pages 3–1125. IEEE, 2002.

[35] Yajie Ma, Mark Richards, Moustafa Ghanem, Yike Guo, and John Hassard. Air pollution monitoring and mining based on sensor grid in london. *Sensors*, 8(6):3601–3623, 2008.

[36] Arati Manjeshwar and Dharma P Agrawal. Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In *Proceedings of the 15th International Parallel & Distributed Processing Symposium*, page 189, 2001.

[37] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 39–49, New York, NY, USA, 2004. ACM.

[38] Yutaka Masumoto. Global positioning system, May 11 1993. US Patent 5,210,540.

[39] MICROCHIP. Miniature single cell, fully integrated li-ion, li-polymer charge management controller. `http://ww1.microchip.com/downloads/en/DeviceDoc/21984a.pdf`, December 2005.

[40] Moteiv Corporation. Telos. `http://moss.csc.ncsu.edu/~mueller/rt/rt11/readings/projects/g4/datasheet.pdf`, May 2004.

[41] G.K. Ottman, H.F. Hofmann, A.C. Bhatt, and G.A. Lesieutre. Adaptive piezoelectric energy harvesting circuit for wireless remote power supply. *Power Electronics, IEEE Transactions on*, 17(5):669–676, 2002.

[42] G.K. Ottman, H.F. Hofmann, and G.A. Lesieutre. Optimized piezoelectric energy harvesting circuit using step-down converter in discontinuous conduction mode. *Power Electronics, IEEE Transactions on*, 18(2):696–703, 2003.

[43] J.A. Paradiso and T. Starner. Energy scavenging for mobile and wireless electronics. *Pervasive Computing, IEEE*, 4(1):18–27, 2005.

[44] M. Philipose, J.R. Smith, B. Jiang, A. Mamishev, S. Roy, and K. Sundara-Rajan. Battery-free wireless identification and sensing. *Pervasive Computing, IEEE*, 4(1):37–45, 2005.

[45] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 364–369, 2005.

[46] Volkan Rodoplu and Teresa H Meng. Minimum energy mobile wireless networks. *Selected Areas in Communications, IEEE Journal on*, 17(8):1333–1344, 1999.

[47] N. Sadagopan, B. Krishnamachari, and A. Helmy. The acquire mechanism for efficient querying in sensor networks. In *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pages 149–155, 2003.

[48] A.P. Sample, D.J. Yeager, P.S. Powledge, and J.R. Smith. Design of a passively-powered, programmable sensing platform for uhf rfid systems. In *RFID, 2007. IEEE International Conference on*, pages 149–156, 2007.

[49] C. Schurgers and M.B. Srivastava. Energy efficient routing in wireless sensor networks. In *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, volume 1, pages 357–361 vol.1, 2001.

[50] SEMTECH. Xe1205. `http://www.semtech.com/images/datasheet/xe1205.pdf`, December 2008.

[51] Steven Shepard. *RFID: radio frequency identification.* McGraw-Hill New York, 2005.

[52] JoshuaR. Smith, AlansonP. Sample, PaulineS. Powledge, Sumit Roy, and Alexander Mamishev. A wirelessly-powered platform for sensing and computation. In Paul Dourish and Adrian Friday, editors, *UbiComp 2006: Ubiquitous Computing*, volume 4206 of *Lecture Notes in Computer Science*, pages 495–506. Springer Berlin Heidelberg, 2006.

[53] Henry A Sodano, Daniel J Inman, and Gyuhae Park. A review of power harvesting from vibration using piezoelectric materials. *Shock and Vibration Digest*, 36(3):197–206, 2004.

[54] Henry A Sodano, Daniel J Inman, and Gyuhae Park. Comparison of piezoelectric energy harvesting devices for recharging batteries. *Journal of Intelligent Material Systems and Structures*, 16(10):799–807, 2005.

[55] James J Spiker. *The Global Positioning System: Theory and Application*, volume 1. Aiaa, 1996.

[56] Robert Szewczyk, Eric Osterweil, Joseph Polastre, Michael Hamilton, Alan Mainwaring, and Deborah Estrin. Habitat monitoring with sensor networks. *Commun. ACM*, 47(6):34–40, June 2004.

[57] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 108–120, 2005.

[58] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. *Internet Computing, IEEE*, 10(2):18–25, 2006.

[59] David L White, Samuel Esswein, Jason O Hallstrom, Farha Ali, Shashank Parab, Gene Eidson, Jill Gemmill, and Christopher Post. The intelligent river©: Implementation of sensor web enablement technologies across three tiers of system architecture: Fabric, middleware, and application. In *Collaborative Technologies and Systems (CTS), 2010 International Symposium on*, pages 340–348. IEEE, 2010.

[60] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 70–84. ACM, 2001.

[61] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM Sigmod Record*, 31(3):9–18, 2002.

[62] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical report, Citeseer, 2001.