ZIGBEE WIRELESS SENSOR NETWORK FOR CAVE ENVIRONMENT MONITORING

By

Yang Li

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical Engineering

New Mexico Institute of Mining and Technology
Department of Electrical Engineering
Socorro, New Mexico
August, 2009
ABSTRACT

Cave environmental studies and cave environment monitoring are important. For instance, through the studies of cave environments, we can better protect cave ecology. Many past experiments have monitored cave environments, although most of those were based on individual sensor nodes such as data loggers. In this thesis, I introduce and discuss of a ZigBee wireless sensor network-based platform used for cave environment monitoring. I carried out a proof-of-concept experiment in Junction Cave at El Malpais National Monument in New Mexico. That experiment monitored temperature, humidity, and air turbulence inside the cave. The instrumentation consisted of a turbulence tower with five thermocouple-based sensors, reaching from the floor to the ceiling of the cave, temperature/humidity sensors distributed throughout the cave, and a low-power embedded Linux computer for data collection and storage. The experiment measured interesting air turbulence variations at different heights, which can be related to weather changes outside the cave and human activities inside the cave. The experiment also observed variations of air temperature at different locations inside the cave. The experiment demonstrated that a ZigBee wireless sensor network-based monitoring system is a potentially feasible platform for a cave environment monitoring system. However I also found that improvements in network reliability and power consumption of the monitoring system should be made in the future.
ACKNOWLEDGMENT

I would like to thank my five committee members for their invaluable assistance in completing this thesis project. Dr. Anders Jorgensen gave me many advices both on the experiment and writing this thesis and most importantly his encouragement. Dr. John Wilson provided great guidance, insight, and many support throughout the project. Dr. William Rison, Dr. Aly El-Osery, and Dr. Ning Liu all gave me many great advice and inspirations on this project.

I would like to acknowledge the cooperation of the National Park Service at El Malpais National Monument, New Mexico for allowing us to carry out the experiment on national park service land.

I would also like to thank Nikki Rendon from Earth & Environmental Science Department who helped me a lot during the whole experiment in the Junction Cave. Dr. Anders Jorgensen and Dr. John Wilson also spent many time and put efforts on the experiment.

I would not have done this thesis project without the help and support from everybody.
# TABLE OF CONTENTS

ACKNOWLEDGMENT ............................................................................................................................ ii

LIST OF FIGURES .............................................................................................................................. vi

LIST OF TABLES ................................................................................................................................. viii

CHAPTER 1 INTRODUCTION .............................................................................................................. 1

CHAPTER 2 ZIGBEE WIRELESS SENSOR NETWORK ................................................................. 8
  2.1 ZigBee Network Basics ................................................................................................................. 8
   2.1.1 ZigBee and IEEE 802.15.4 ...................................................................................................... 11
   2.1.2 Frequencies of operation and Data Rates ............................................................................. 12
   2.1.3 ZigBee Device Types and Roles .......................................................................................... 14
   2.1.4 ZigBee Addressing ............................................................................................................... 15
   2.1.5 ZigBee Networking Topologies ........................................................................................... 16
   2.1.6 ZigBee is Self Forming and Self Healing .......................................................................... 18
   2.1.7 ZigBee Gateway .................................................................................................................. 20
  2.2 ZigBee Development Tools ........................................................................................................ 20

CHAPTER 3 HARDWARE AND CONFIGURATIONS .................................................................... 22
  3.1 ZigBee Devices .......................................................................................................................... 22
  3.2 Single Board Computer .......................................................................................................... 26
  3.3 Sensors and Conditioning Circuits ......................................................................................... 29
LIST OF FIGURES

FIGURE 2-1: ZIGBEE WIRELESS NETWORKING PROTOCOL LAYERS [12] ............................................. 12
FIGURE 2-2: A STAR NETWORKING TOPOLOGY ........................................................................ 16
FIGURE 2-3: A TREE NETWORKING TOPOLOGY ....................................................................... 17
FIGURE 2-4: A MESH NETWORKING TOPOLOGY ....................................................................... 17
FIGURE 3-1: DIMENSION OF THE MC13213-NCB ..................................................................... 23
FIGURE 3-2: DIMENSION OF THE MC13213-SRB ..................................................................... 23
FIGURE 3-3: THE MC13213 CHIP FRONT AND BACK ................................................................. 23
FIGURE 3-4: A SIMPLIFIED MC13213 SYSTEM LEVEL BLOCK DIAGRAM [21] ......................... 24
FIGURE 3-5: A DIAGRAM OF THE NETWORK LAYOUT USED IN THIS THESIS WORK ............. 26
FIGURE 3-6: TS-7260 SINGLE BOARD COMPUTER ................................................................... 27
FIGURE 3-7: BASE STATION ........................................................................................................ 28
FIGURE 3-8: ZED WITH AN EXTERNAL SENSOR BOARD ON TOP ............................................. 30
FIGURE 3-9: THE CIRCUIT TO PROVIDE +3.3V AND +5V REGULATED POWER SUPPLY ........ 30
FIGURE 3-10: THE CIRCUIT OF A VOLTAGE DIVIDER FOR THE THERMISTOR ...................... 31
FIGURE 3-11: THE CIRCUIT OF A VOLTAGE DIVIDER FOR THE RELATIVE HUMIDITY SENSOR ................................................................................................................................. 31
FIGURE 3-12: RESISTANCE VS. TEMPERATURE PLOT FOR PS503J2 THERMISTOR ............... 34
FIGURE 3-13: THE THERMOCOUPLE USED IN THIS PROJECT ............................................... 37
FIGURE 3-14: THERMOCOUPLE IS PLACED INSIDE OF A 1 FOOT LONG PVC PIPE WITH THE

  1ST STAGE AMPLIFIER ........................................................................................................... 37
FIGURE 3-15: THE CONFIGURATION OF THE FIRST STAGE AMPLIFIER CIRCUIT FOR THE

  THERMOCOUPLE ............................................................................................................... 37
FIGURE 3-16: OFFSET NULLING CIRCUIT FOR THE OP27 OP-AMP ........................................... 41
FIGURE 3-17: THE CIRCUIT USED TO PROVIDE THE 0.75V REFERENCE ..................................41
FIGURE 3-18: THE SECOND STAGE NON-INVERTING AMPLIFIER .................................41
FIGURE 3-19: THE REFERENCE JUNCTION IS SOLDERED ONTO THE PCB BOARD ..........42
FIGURE 3-20: THE MEASURING JUNCTION OF THE THERMOCOUPLE STICKING OUT OF THE PVC PIPE AND EXPOSED TO SURROUNDING AIR .................................................42
FIGURE 4-1: A TIMING MEASUREMENT, TIME 1: THIS IS THE TIME THE ZED HAS JUST FINISHED TRANSMITTING A PACKET OUT OVER-THE-AIR, TIME 2: THIS IS THE TIME THE ZC STARTED TO WRITE THE PACKET ON THE DATA LINE OF THE SERIAL CABLE TO THE SBC. ..................................................................................................................45
FIGURE 4-2: AN EXAMPLE OF A PRACTICAL ZIGBEE NETWORK .................................52
FIGURE 5-1: THE ZIGBEE COORDINATOR, THE SINGLE BOARD COMPUTER AND THEIR BATTERIES PLACED TOGETHER AS THE BASE STATION ..........................................................60
FIGURE 5-2: THE TOWER IS PLACED ROUGHLY 75 FEET FROM THE ENTRANCE. LIGHT FROM THE ENTRANCE IS VISIBLE IN THIS PICTURE .......................................................60
FIGURE 5-3: WHOLE NETWORK LAYOUT IN THE JUNCTION CAVE ...............................61
FIGURE 5-4: 1-MINUTE STANDARD DEVIATIONS OF THE FIVE SENSOR SIGNALS ..........64
FIGURE 5-5: TEMPERATURE DATA ACQUIRED DURING THE CALIBRATION PROCESS ....66
FIGURE 5-6: A CLOSER LOOK AT THE TEMPERATURE CALIBRATION TAIL .................67
FIGURE 5-7: CALIBRATED DATA PLOT FOR ALL FOUR TEMPERATURE SENSORS ..........68
FIGURE 5-8: DATA PLOT FOR FOUR RELATIVE HUMIDITY SENSORS .........................70
LIST OF TABLES

TABLE 2-1: A COMPARISON OF ZIGBEE, WI-FI, AND BLUETOOTH STANDARDS .................. 10
TABLE 2-2: IEEE 802.15.4 DATA RATES AND FREQUENCIES OF OPERATION ....................... 14
TABLE 4-1: A LIST OF THE HARDWARE USED IN THE PROJECT ........................................ 53
TABLE 5-1: SENSOR HEIGHTS ABOVE THE CAVE FLOOR ..................................................... 64
TABLE 5-2: READINGS FROM FOUR TEMPERATURE SENSORS AT 20:20 AND EVALUATED
OFFSETS .......................................................................................................................... 68
CHAPTER 1 INTRODUCTION

Cave environments are full of interesting information. Caves and caverns are often habitat for animals and for human beings as well in ancient times [1]. Caves are being discovered and explored all over the world. Cave environment monitoring and research is important for many reasons. For example: (a) to protect caves because of their archaeological and cultural importance [1]. For example, the Great Cave of Niah, in Malaysia, contains evidence of human activity and habitation dating back to 40,000 years ago [2]; (b) to evaluate the impact of human activity on the environment in different caves and to understand how it might affect life in caves; (c) to protect caves as natural resources [3] and to protect cave ecological systems. Many efforts had been made to protect various animals living in caves. For example, in [4] cave micrometeorology studies were used to help restore a large population of endangered Indiana bats; (d) useful to geological research because they can reveal details of past climatic conditions [1]; (e) to evaluate the relationship between the changes of cave environments and global climatic changes (such as global warming and pollution problems); (f) to measure and evaluate caves as analogs to caves on other planets, including Mars. It has been proposed that Martian caves are one of the most likely places to find life [5]. Since many caves on Mars are probably lava tubes similar to the ones on earth, these Martian caves may also be good places for future human habitations because the radiation environment on the surface of Mars is harsh and not suitable for humans. For this reason, efforts have been
made to develop the necessary technologies to enter Martian caves and detect life in them [5]. There are many other reasons why we are interested in studying cave environments. For example, certain anomalous vertical temperature gradients, abnormal temperature differences between different parts of a cave, and unusual air flow directions have been observed [6,7]. We are interested in all unknown sources or factors which are causing these abnormal phenomena as well. Parameters that we may be interested in measuring in a cave environment may include: air pressure, air temperature, air flow, air turbulence, composition of various gasses in the air, including humidity, fluid temperature, soil temperature, soil humidity, etc.

Cave environments had been studied and monitored in many published works [8, 9, and 10]. In each of these experiments stand-alone data loggers were used as the primary platforms of the cave environment monitoring systems. When using data loggers in caves querying data from each sensor manually is tedious. Furthermore, obtaining real-time information is not easy because data are only available when people enter those caves, and in many cases this human activity will disturb the cave environment.

Getting real-time data is potentially important because it can also be used to trigger actions of the monitoring system which may consume more energy and/or memory. These may include: 1) increasing the sensors' sampling rate when the system detects or predicts that a particular interesting event is taking place or about the take place. The system may decide that it is necessary to use more energy and more memory space to record more detailed information of that event; 2) turning on a surveillance camera which can be used to detect and record events such as human or animal intrusions or to record significant weather changes (storms, rain) outside the cave. These kinds of
information from cameras can help us understand and explain weather changes inside the cave or relate outside and inside weather conditions. These advanced capabilities of a cave environment monitoring system all make use of real-time information and the monitoring system's ability to exchange information within the system between nodes and process that information.

According to the study of [11], temperature variations in caves can sometimes have a cycle up to several years. As a consequence of this observation, we expect to monitor a cave environment up to the scale of years as well which requires the cave environment monitoring system to have very low power consumptions. We also want the monitoring system to have a long lifespan because we want to minimize the need for human intervention and accompanying disturbance to the cave environment while monitoring is in progress (the cave environment can remain disturbed for a relatively long period of time even after a short entry). Long run time and reliability are important for this reason.

We also require the entire cave environment monitoring system to have very low power consumption not only for the purpose of achieving a long run time, but also because power consumption in the system generates heat which will affect the cave environment. An interesting calculation regarding the heating effects on the cave environment by electrical components by Dr. Richard Sonnenfeld is reproduced in Appendix B.

To sum up, the general performance specifications for this cave environment monitoring system should are: avoiding cables, thus requiring a wireless network, being
able to be carried and deployed easily, be affordable, reliable, and have a low maintenance overhead requiring infrequent visit, having very low power consumption and long run time, preferably years, have little impact on the environment, and be impacted little by the environment. These specifications lead us to consider and choose the ZigBee wireless communication standard as a platform for the cave environment monitoring system over other available technologies.

ZigBee is a wireless network standard which defines a set of communication protocols for self-organizing, low power, low data rate, short range wireless networking [12]. Compared to other available wireless networking protocols, such as Wi-Fi and Bluetooth, the advantages of ZigBee protocol include: 1) support of mesh networking topology in which devices can both talk to each other directly and have messages relayed by other devices to a destination by the means of multi-hoping, 2) significantly lower power consumption for similar transmission ranges, and 3) much lower hardware and software complexity. A detailed comparison between these wireless networking protocols is shown in Table 2-1. These advantages of the ZigBee protocol simply make it a better candidate for a platform for a wireless monitoring system in a cave environment where resources, such as power, are extremely critical. Moreover, ZigBee Wireless Sensor Networks (WSNs) make it easy to obtain real-time data from the measuring system through gateways which are used as interfaces between ZigBee and other communication protocols (such as Wi-Fi), where the gateway transmits data out to any place or available platform we desire. The main noticeable disadvantage of ZigBee protocol is a much lower data rate (250Kbit/second) when it is compared to others. However it should be
enough for a monitoring system in cave environments without sensors measuring data at high frequencies. A basic introduction to the ZigBee standard is provided in chapter 2.

In this thesis, I present our first attempt at implementing a ZigBee-based WSN in a cave environment as a monitoring system platform. The experiment was carried out in the Junction Cave lava tube at El Malpais National Monument in New Mexico. Because this was a first attempt we only planned to monitor a few basic and essential parameters of the cave environment. These included: air temperature, air relative humidity, and air turbulence. We want to measure air turbulence inside caves because air turbulence is actually a measure of heat flow inside caves which is an important measure of air movements in a cave environment. In this thesis work, I measured air turbulence by using thermocouples which can be used to measure rapid air temperature variations [13]. I designed and built a tower with five thermocouple-based sensors placed at five different heights. These sensors measure the difference between an averaged air temperature value (averaged over a time period of at least several minutes) and a instantaneous air temperature value (with a response time of about 0.05 seconds). The response time is that short because the thermocouple junction is very thin. Output from the thermocouple-based sensors are sampled and reported wirelessly to the base station at 30 Hz for this particular experiment. Cave environments are usually rather static. For instance, the air temperatures of most caves are close to the mean-annual temperature of the overlying surface [14]. Changes in cave temperatures can thus be very small such that relatively high precision and accuracy measurements are needed to detect these changes. According to the study of [15], the resolution of temperature measurement should be 0.001 degree C in order to detect small temperature fluctuations (0.02-0.05 degree C), although it varies
in different cave environments. However, in this thesis work, I use thermistors as temperature sensors with a resolution of 0.1 degree C because this is the best thermistor we can get based on our budget. Since this experiment is a proof-of-concept and practical first attempt we find this to be a reasonable compromise, especially since the main focus was to see and evaluate the performance of the ZigBee WSN as a platform for a cave environment monitoring system. The resolution of the relative humidity sensors I used in this experiment is 3.5% which is based on our budget as well. In the future all the sensors should be designed with sufficient resolution based on the cave environments and of course project budgets.

The rest of the thesis is organized as follows: Chapter 2 begins with a basic background introduction to ZigBee Wireless Sensor Networks. The options of choosing ZigBee development tools from different vendors are also briefly discussed.

Chapter 3 discusses the experiment design specifications and decisions and also technical aspects of all the hardware used in the experiment. These include: temperature and relative humidity sensors, a thermocouple tower used to measure air turbulence in the cave, conditioning circuits for all these sensors, all the ZigBee devices used to collect and gather data from these sensors, and a low-power embedded Linux computer which is for data storage.

Chapter 4 presents my analysis of the ZigBee WSN. I discuss the network performance according to two critical parameters: 1) network size limitations; 2) network life-span limitations. In the analysis of network size limitations I discussed the network inherent limitations, the limitations caused by practical factors such as hardware
processing capabilities, and limitations imposed by network performance requirements such as network latency tolerance. In the analysis of network lifespan limitations, I discuss power consumption of all the devices used in the network and memory consumption of the embedded Linux computer.

Chapter 5 is dedicated to the experiment that was carried out in the Junction Cave lava tube at El Malpais National Monument in New Mexico. All experimental results are shown in this chapter. These include: air turbulence monitored by the thermocouple tower, temperature variations measured by sensors at different locations in the Junction Cave, relative humidity sensor readings, and network performance.

Chapter 6 concludes this thesis by discussing the contribution of this thesis work to both the field of cave environment studies and the field of ZigBee WSN practical applications. In addition, I discuss all the lessons learned from the experiment and some suggestions for future work.

Appendix A provides a brief set of instructions on how to use cross-compilers to compile C program on full Linux computers for the ARM-based embedded Linux computer. It also lists the C program used on the embedded Linux computer to retrieve data from the ZigBee Network Coordinator through the serial communications link.

Appendix B provides an interesting calculation of the heating effects to cave environments which are produced by electrical components consuming power. This calculation was performed by Dr. Richard Sonnenfeld who is an Associate Professor and Research Scientist in the Physics Department at New Mexico Tech.
CHAPTER 2 ZIGBEE WIRELESS SENSOR NETWORK

ZigBee is a standard which defines a set of communication protocols for self-organizing, low power consumption, low data rate, and short range wireless networking [12]. As indicated in Chapter 1 I found, based on the performance specifications for the cave monitoring system that ZigBee is a better candidate for the monitoring system. The reasons why I chose ZigBee as a platform for the cave environment monitoring systems include: 1) ZigBee supports mesh network topology which makes it easier to implement information exchange within the whole network; 2) ZigBee has significantly lower power consumption as compared to other protocols for similar transmission ranges; 3) ZigBee devices have much lower hardware and software complexity. This chapter begins with a basic introduction to ZigBee WSN which is provided here to familiarize readers with the subject. I also compare the ZigBee protocols to other wireless protocols. I also discuss options for choosing ZigBee development tools.

2.1 ZigBee Network Basics

Some scientists believe that the complexity of the honey bees' language is second only to ours [16]. Imagine a group of honey bees are flying from flower to flower collecting sweet juices or nectar. When they want to communicate a certain message all
the way back to their hive, no single bee will fly all the way back, instead, they will use message relaying. Each bee will perform a certain zigzag dance for another bee, which then repeats the message for another bee closer to the hive. The process is repeated until the message is relayed all the way back to the hive. The name ZigBee was chosen because all the devices in the network communicate with each other in similar manner to how the bees do their job. In a ZigBee wireless network, a message from a node is relayed by other nodes by short-range communications when the destination of the message is out of radio range of the originating node. The main advantage of relaying a message rather than one straight transmission between two nodes is that it consumes less RF power. The transmission power used by radios is proportional to the distance squared. So, for example, if one message needs to be transmitted from a sender some distance to the receiver, it is consumes 10 times less transmission power to use 10 intermediate relays or hops through other nodes than to transmit in one single hop from the sender to the receiver. Another advantage of this multi-hop network protocol is that it is possible to use a channel simultaneously in different parts of the network to transmit messages without collisions.

ZigBee-based wireless devices generally operate in these three unlicensed frequency bands: 868 MHz, 915 MHz, and 2.4 GHz with a maximum data rate of 250 K bits per second. ZigBee is targeted for battery-powered applications where low cost, low data rate and extended battery life are required. Zigbee was also built to enable potentially very large mesh networks.

The ZigBee standard is developed by the ZigBee Alliance [17] which was founded in 2002 as a nonprofit organization. This organization has hundreds of members,
such as Freescale semiconductor, Philips, Texas Instruments, Atmel, Honeywell, GE, Hitachi, Samsung, and Sony, from semiconductor industry developers to original equipment manufacturers (OEMs).

By comparing ZigBee standard with some other wireless standards such as IEEE 802.11 WLAN and Bluetooth, we can find out what the differences are between these standards and so that we can get more familiar with ZigBee. The table below outlines some of the main differences between ZigBee standard and others.

Table 2-1: A Comparison of ZigBee, Wi-Fi, and Bluetooth Standards

<table>
<thead>
<tr>
<th></th>
<th>ZigBee</th>
<th>802.11a,b,g,n (Wi-Fi)</th>
<th>Bluetooth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Rate</strong></td>
<td>20, 40, and 250 Kbits/s</td>
<td>54,11,54,+100 Mbits/s</td>
<td>1 Mbits/s</td>
</tr>
<tr>
<td><strong>Range</strong></td>
<td>10-100 meters</td>
<td>32m Indoor ~ +½ miles outdoor</td>
<td>10 meters</td>
</tr>
<tr>
<td><strong>Networking Topology</strong></td>
<td>Ad-hoc, star, tree, mesh</td>
<td>Point to hub, Ad-hoc</td>
<td>Ad-hoc, very small networks</td>
</tr>
<tr>
<td><strong>Operating Frequency</strong></td>
<td>868 MHz (Europe) 900-928 MHz (NA) 2.4GHz (worldwide)</td>
<td>2.4 and 5 GHz</td>
<td>2.4 GHz</td>
</tr>
<tr>
<td><strong>Complexity (Device and application impact)</strong></td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td><strong>RF Power</strong></td>
<td>Minimum 1 mW (defined by 802.15.4, no specified maximum), indoor 30m, outdoor 100m</td>
<td>200mW</td>
<td>100mW, 2.5mW, 1mW (Max Range: 100m, 20m, 10m ) [17]</td>
</tr>
<tr>
<td><strong>Power consumption (Battery option and life)</strong></td>
<td>Very low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td>128 AES plus application layer security</td>
<td>WPA, WPA2, EAP-TLS</td>
<td>64 and 128 bit encryption</td>
</tr>
<tr>
<td><strong>Typical Applications</strong></td>
<td>Industrial Control and monitoring, Wireless Sensor Networks, building automation, home control and automation</td>
<td>Wireless LAN connectivity, broad band internet access</td>
<td>Wireless connectivity between devices such as phones, laptops, and headsets</td>
</tr>
</tbody>
</table>
Notice that in the table above the Networking Topology of ZigBee is unique: supporting ad hoc, star, tree and mesh networking. Ad-hoc means instead of relying on a base station to coordinate the flow of messages to each node in the network, the individual network nodes forward packets to and from each other; Mesh networking in a ZigBee network means multi-hop data transmission with automatic routing in the protocol layer such that the application does not need to worry about any of that. ZigBee standard also has the lowest power consumption compared to the other two. ZigBee’s low power consumption is rooted not in RF power, but in a sleep mode specifically designed to accommodate battery powered devices [18]. When the goal of an application is to send and receive commands and/or collect information from sensors such as temperature and humidity sensors over a short distance from a large number of sensors, ZigBee provides a cost-effective way compared to the other two standards described above. Further detailed ZigBee basics are discussed in the following sections.

2.1.1 ZigBee and IEEE 802.15.4

Figure 2-1 shows the ZigBee wireless networking protocol layers, the bottom two networking layers: Physical Layer (PHY) and the Medium Access Control Layer (MAC) is defined by IEEE 802.15.4 standard [18]. This standard is developed by the IEEE 802 standards committee and was released in 2003 initially. IEEE 802.15.4 defines only PHY and MAC layers of wireless networking but no other higher networking layers. The ZigBee standard defines the Network Layer, the Application Layer and security layers of the protocol.
Figure 2-1: ZigBee Wireless Networking Protocol Layers [12]

Characteristics that belong to physical level are all determined by the PHY layer specifications. Hence, parameters of a ZigBee network and devices such as data rate, receiver sensitivity requirements and frequencies used in operations are all specified in the IEEE 802.15.4 standard.

2.1.2 Frequencies of operation and Data Rates

According to the latest version of IEEE 802.15.4, there are three frequency bands for this standard to operate: 868 - 868.6 MHz (868 MHz band) 902 - 928 MHz (915 MHz band) 2400 - 2483.5 MHz (2.4 GHz band). The 868 MHz band is mainly used in Europe. The 915 MHz and 2.4 GHz band are in the Industrial, Scientific and Medical (ISM) frequency band. In North America, these bands are the 260 ~ 470MHz, 902 ~ 928MHz, and 2.4GHz [18]. For ZigBee, there are 16 channels in the 2.4 GHz band and they are numbered 11-26 (channels 0-11 are used in sub 1 GHz bands). Channels are physically
separated by 5 MHz, to minimize interference. The entire ZigBee Network operates on the same channel at any given time. Channel hopping may be permitted in the future; for now, a Personal Area Network (PAN) forms on one channel only [19]. On any given channel, in a given location, only one radio may be transmitting at one time. The ZigBee protocol uses the Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) protocol to allow multiple devices to share the same frequency. Devices in a ZigBee Network listen for a clear channel before transmitting or back off for a random time before continuing [13]. This mechanism is actually a Time Division Multiplexing shared channel communication protocol.

The 2.4 GHz band of ZigBee is used worldwide and it has the maximum data rate of 250Kbit per second. It is popular for manufacturers to develop transceivers in the 2.4 GHz band, although there are some disadvantages of using this frequency band. For instance, the lower the frequency band is, the better the signals can penetrate walls and objects [13]. As a result, some users find the 868 MHz and 915 MHz band do a better job in their applications.

There are three kinds of modulations that are used in IEEE 802.15.4: Binary Phase Shift Keying (BPSK), Amplitude shift Keying (ASK) and Offset Quadrature Phase Shift Keying (O-QPSK). All the schemes use either Direct Sequence Spread Spectrum (DSSS) or Parallel Sequence Spread Spectrum (PSSS) techniques. Both of the techniques can help improve the receiver performances in multipath environments [19], such as in a building or in a cave environment. In the table below, further information about the frequencies of operation and data rates are provided.
Table 2-2: IEEE 802.15.4 Data Rates and Frequencies of Operation

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Number of Channels</th>
<th>Modulation</th>
<th>Bit Rate (Kbits/s)</th>
<th>Spreading Method</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>868-868.6</td>
<td>1</td>
<td>BPSK</td>
<td>20</td>
<td>Binary DSSS</td>
<td>Mandatory</td>
</tr>
<tr>
<td>902-928</td>
<td>10</td>
<td>BPSK</td>
<td>40</td>
<td>Binary DSSS</td>
<td>Mandatory</td>
</tr>
<tr>
<td>868-868.6</td>
<td>1</td>
<td>ASK</td>
<td>250</td>
<td>20-bit PSSS</td>
<td>Optional</td>
</tr>
<tr>
<td>902-928</td>
<td>10</td>
<td>ASK</td>
<td>250</td>
<td>5-bit PSSS</td>
<td>Optional</td>
</tr>
<tr>
<td>868-868.6</td>
<td>1</td>
<td>O-QPSK</td>
<td>100</td>
<td>16-array orthogonal</td>
<td>Optional</td>
</tr>
<tr>
<td>902-928</td>
<td>10</td>
<td>O-QPSK</td>
<td>250</td>
<td>16-array orthogonal</td>
<td>Mandatory</td>
</tr>
<tr>
<td>2400-2483.5</td>
<td>16</td>
<td>O-QPSK</td>
<td>250</td>
<td>16-array orthogonal</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>

2.1.3 ZigBee Device Types and Roles

Two types of devices are provided in a ZigBee network: Full-Function Devices (FFDs), and Reduced-Function Devices (RFDs). A FFD has the ability to perform all the tasks specified in ZigBee standard, which includes receiving messages from other devices, sending messages to other devices, and relaying data for other devices. Since the FFDs are supposed to always leave their Radio on so that they can perform all their tasks, the FFDs are usually envisioned as being powered by wall power [20]. Whereas, RFDs on the other hand are designed to be battery-powered with minimum RAM and ROM space and they do not leave their receiver on all the time.

Each ZigBee device plays one of three roles in a ZigBee network: ZigBee Coordinator (ZC), ZigBee Router (ZR), and ZigBee End Device (ZED).

The ZC roles include:
Starting a network
Selecting a Personal Area Network Identifier (PAN ID) for the network
Allowing devices to join or leave the network
Performing all the functions of a ZigBee router
Containing the trust center in a secure network

The ZR roles include:
- Route data between ZigBee devices
- Allow devices to join or leave the network
- Manage messages for its children that are end devices
- Optionally perform all the functions of a ZigBee end device

The ZED is a reduced-function device that:
- Sleeps to save power, so it can be battery powered
- Requires fewer memory resources because it does not store network-wide information
- Performs functions such as switching a light on or off or monitoring an occupancy sensor

### 2.1.4 ZigBee Addressing

Every IEEE 802.15.4 radio has a 64-bit address that is unique in the world. Every node in a ZigBee network has a 16-bit network address that is unique within that network. ZigBee does not send messages with 64-bit addresses. When a ZigBee application tells the software stack to send a message to an IEEE address, the Application Support Sub layer attempts to discover the 16 bit address prior to sending the packet [13].
2.1.5 ZigBee Networking Topologies

There are basically three types of networking topologies in ZigBee networks:

- A Star Network Topology
- A Tree Network Topology
- A Mesh Networking Topology

In a ZigBee Star Network, as shown in Figure 2-2, one or more ZEDs are directly joined to ZC. All ZEDs talk to ZC directly and the ZC relays messages between end devices.

![Figure 2-2: A Star Networking Topology](attachment:image.png)
In a ZigBee Tree Network, as shown in Figure 2-3, the network extends the Star Network with the use of ZigBee Routers (ZRs). ZC forms the network with one or more routers joined, optionally, one or more ZEDs associated in a hierarchical structure. All messages in this Tree Network move up or down the parent-child hierarchy. Each message transfer from one node to the next is called a hop. At the time of network
formation, when a node B is joining the parent node A, the parent node records the address of the child node B in an address map so that later when the parent node A receives a message, it will be able to tell if the recipient of the message is below it in the tree. If the recipient is not below it, the message will be passed to node A's own parent.

In a ZigBee Mesh Network, as shown in Figure 2-4, routers can talk to each other directly as long as they are within radio range of each other and the message moves in one hop from one router to another, without any other node involved. If they are beyond each other's radio range, the message will travel through other routers, according to a path which the network established based on its routing efficiency. ZigBee uses the Ad Hoc On-Demand Vector routing protocol for mesh routing [20]. To find a mesh route, a route request broadcast is sent out by the originator. Other nodes will receive this broadcast and see if they know the destination. They will forward the route request if they do not know the destination, with the incremented path cost updated. If they do know the destination, they will send a reply to the originator. The relay message contains the total cost of the routing path to the destination which makes it possible for the originator to determine which route is the best when multiple route replies are received.

2.1.6 ZigBee is Self Forming and Self Healing

A ZigBee network is formed when a ZC declares itself a coordinator and also establishes a Personal Area Network ID (PAN ID) that is not used by another ZigBee Network in range on a preferred channel. If there is more than one ZC in a ZigBee Network with the same PAN ID, they will form separate networks on different channels.
without conflicts [20]. The channel is always selected by the ZC within the set of the channels programmed in it, with the least network traffic, the least noise. A ZC can be programmed to choose either a random PAN ID or a default one which is programmed beforehand. A ZC permits other devices to join the network with the same programmed default PAN ID. Devices that are programmed to use a random PAN ID can join any ZigBee Network. ZigBee devices discover others by broadcasting or unicasting a message [20]. ZRs and ZEDs can join the network either by joining the ZC directly or by joining ZRs which are already joined to this network, all by sending an association request to the network. Once the request from one device is acknowledged, by the ZC or by a ZR, the device is registered in this network [20]. Then a neighbor table is updated on each node to remember their relations. If there is more than one parent node in a network that a ZigBee device is joining, the device chooses the parent closest to the root of the tree which is also accepting children. Each router has by default the capacity for 6 router children and 14 end device children. RFDs are generally designed to be programmed as ZEDs. ZEDs do not accept children. Nodes that are not within each other's radio range can still talk to each other because ZC and ZRs can relay messages from one to another. FFDs are used mostly as either a ZC or ZRs. Since no supervision is required during the network forming process, ZigBee Networks are considered as Self Forming Networks.

In ZigBee Tree and Mesh Networks, when more than one way to relay a message from one device to another are available, the most cost-effective route is selected by the originator dynamically based on the connectivity and the route path cost assessment. This means that if a router in a route path fails for any reason, such as the exhaustion of its battery, the originator will find an alternative router to relay the message to the
destination, again, as long as there is any router within range to do so. Therefore, ZigBee network is also considered to be a Self Healing Network both in a Tree Networking Topology and in a Mesh Networking Topology.

2.1.7 ZigBee Gateway

A ZigBee gateway device serves as a bridge between ZigBee-based networks and other type of networks. For example, a ZigBee-Ethernet gateway device will provide the interface between a ZigBee network and an Ethernet IP-based network. The gateway translates ZigBee packets to Internet protocol packets, and vice versa.

2.2 ZigBee Development Tools

In this thesis project, we chose the MC1321x Development Kit from Freescale Semiconductor, Inc. This is the Freescale second generation development kit which offers several different hardware platforms for evaluating the MC1321x system in package (SiP). The MC13213 chip which is used in this development kit integrates a 2.4 GHz RF transceiver and a MC9S08GT60 microcontroller into a single chip only 9x9x1mm in size. The combination of the radio and a microcontroller in a small footprint package allows for a cost-effective solution. Descriptions of the devices provided in this development kit and detailed information are presented in Chapter 3.

There are also many other ZigBee development kits that are available in the market from other companies: Atmel, Texas Instruments, Jennic, Ltd., Digi International/Maxstream, Ember, Meshnetics, and Microchip Technology, etc.
Digikey.com provides a Wireless Solution section in the Technology Zones. It is a good place to see and compare all the available ZigBee development tools from different companies and to make a choice based on specific development requirements. Information about products from different vendors is also summarized on the ZigBee Alliance webpage. Atmel sells some chips in DIP packages which are easy to experiment with on a prototype board. The Freescale tools are expensive and the chips are sold in LGA packages which are difficult to work with in the laboratory. Zigbee devices from different vendors should be compatible in theory.
CHAPTER 3 HARDWARE AND CONFIGURATIONS

In this chapter, I discuss all the hardware and detailed designs used and built in this project. Hardware performance issues such as power consumptions, memory consumptions, and Network performances will be further discussed in Chapter 4.

In Section 3.1, I discuss the ZigBee devices that I used in this project and their configurations, in Section 3.2, I discuss the Single Board Computer I used in this project as storage of all the data collected from the network, and then in Section 3.3 I discuss all the sensors and their conditioning circuits in detail.

3.1 ZigBee Devices

The MC1321x development kit consists of two types of boards: MC13213-SRB Sensor Reference Board and MC13213-NCB ZigBee Coordinator Board. SRBs are mainly used as End Devices in a ZigBee wireless sensor network. I use NCBs as either a ZigBee Coordinator or a Network Router. The NCB and SRB are shown in Figure 3-1 and Figure 3-1 respectively with dimensions.
Figure 3-1: Dimension of the MC13213-NCB

Figure 3-2: Dimension of the MC13213-SRB

Figure 3-3: The MC13213 chip front and back

The MC13213 chip, as shown in Figure 3-3, mounted on these two boards contains 60K of flash and 4K of RAM. The MC13213 chip supports ZigBee applications
that use a stack from 3rd party vendors as well [21]. The MC1321x family is Freescale’s second-generation ZigBee platform which integrates a low power 2.4 GHz radio transceiver and a MC9S08GT60 8-bit microcontroller into a single 9x9x1 mm 71-pin Land Grid Array (LGA) package which is soldered directly onto the Printed Circuit Board (PCB). The MC9S08GT60 microcontroller contains an 8 Channel 10-bit analog to digital converter (ADC) module, and up to 32 General Purpose Digital Input/Outputs (DIO). A serial peripheral interface (SPI) module handles the all data communication between the microcontroller and the transceiver. A simplified block diagram of the MC13213 chip is shown in Figure 3-4 [21].

![Figure 3-4: A simplified MC13213 System Level Block Diagram [21]](image)

For this thesis four SRBs are all programmed as ZEDs to read two ADC input voltages (with 10-bit conversions) from a thermistor and a relative humidity sensor on a
external sensor board and report data to the ZigBee Coordinator once every 10 seconds. I used 10 seconds in this project just for demonstration, but any other time interval can be used as well and it is likely that longer time intervals are sufficient, increased sampling intervals means decreased duty cycle and thus decreased power consumptions. Discussions on power consumption are provided in chapter 4. The valid ADC input range is from 0V to 3V and the maximum range that ADC input pins can tolerate is from -0.3V to 3.3V. A DIO pin of this SRB is programmed to turn on the power of the external sensor board before reading in voltages and turn off the power of this external sensor board once the reading is done. This external sensor board is discussed in Section 3.3. The time it takes to power up the external sensor board, allowing it to stabilize, and to sample and transmit data is less than 150 ms. The SRBs and their external sensor boards spend the rest of the time in low power mode to conserve battery power.

One NCB is programmed to be the Network Router in this project. The Router is running with the transceiver turned on all the time to listen to the reports from any ZED which needs this Router to relay reports to the ZigBee Coordinator. The routing is taken care of automatically by the protocol layer without any programming by me other than to define it to be a ZR. At the same time, this NCB is programmed to read five ADC input voltages (with 10-bit conversions) at the sampling rate of 30 Hz from five thermocouples which are installed in a single row vertically, separated evenly on a 14 feet high tower. The same 0-3V ADC input range applies. These thermocouples are used to measure air turbulence by measuring rapid temperature fluctuations. The sensor boards are discussed in Section 3.3.
Another NCB is programmed to be the ZigBee Coordinator which listens to the reports from the Router and any End Nodes that are reporting directly to it. The transceiver on the ZigBee Coordinator is left on all the time as well. When data are received over-the-air, they are transmitted immediately through the RS232 interface on the NCB to a Single Board Computer. The Single Board Computer is discussed in Section 3.2. Figure 3-5 shows the specific network layout I used in this thesis work for the experiment carried in Junction Cave. The network layout varies accordingly to different situations.

![Network Layout Diagram](image)

**Figure 3-5: A diagram of the Network Layout used in this thesis work**

### 3.2 Single Board Computer

The TS-7260 from Technologic Systems®, as shown in Figure 3-6, is a compact, full-featured Single Board Computer (SBC) based upon the Cirrus EP9302 ARM9 CPU. The EP9302 features an advanced 200 MHz ARM920T processor design with a memory...
management unit (MMU) that allows support for high level operating systems such as Linux, and Windows CE. In this project, I installed TS-Linux on a 2GB SD Card which is then used as the computer's primary boot device and storage. The SD card is inserted into the on-board SD card socket. TS-Linux is a mini-Linux distribution. I wrote a C-program for the SBC to collect data from the serial port. It looks for synchronization words within the reports received by ZC and for the ID of the transmitting node, and then stores the data packets in files, one for each originating node. The C code is compiled in another full Linux operating system with a cross-compiler for ARM-based processors. The C program and some simple instructions on the cross-compiling are provided in the Appendix A.

![Figure 3-6: TS-7260 Single Board Computer](image)

This SBC requires less than 1 Watt for full-speed operation and ¼ Watt at a lower speed which makes this SBC ideal for use in power sensitive designs, such as this battery-powered system. The SBC can run for more than 24 hours on just a few 9V batteries. It can run for much longer with a heavier battery of course. The power consumptions and battery choices are discussed in Chapter 4.
The SBC is equipped with three serial ports, one 10/100 Ethernet port and two USB ports. One of the three serial inputs is configured as a serial login console, such that the computer can be accessed with a terminal program running on a laptop (for example Minicom in Linux or HyperTerminal in Windows). It is also possible to connect to the computer via an Ethernet cable (using a cross-over cable for a direct connection to a laptop) with ‘ssh’ or ‘telnet’. The Ethernet connection is faster and more convenient for downloading large data sets from the SBC.

As shown in Figure 3-7, we call it the Base Station. The SBC is connected to the ZigBee Coordinator (a NCB) through the RS232 serial port interface. All the data received by the ZigBee Coordinator over-the-air are transferred and stored immediately onto the SD card of the SBC. A real-time clock on the SBC is used to add a timestamp at the end of each report when the data are received by the SBC.

![Figure 3-7: Base Station](image)
3.3 Sensors and Conditioning Circuits

In this section I discuss the different sensors and related conditioning circuits. In Section 3.3.1 I discuss the design decisions about all the sensors I used in this project. In Section 3.3.2 I describe the temperature and the relative humidity sensors used with the ZEDs. In Section 3.3.3, I describe the thermocouples and related designs.

3.3.1 Design decisions and specifications

According to the performance specifications and our plan to measure the basic and essential weather parameters in a cave environment for the first proof of concept experiment, I designed and implemented temperature sensors, relative humidity sensors, and thermocouple based sensors which are used to measure air turbulence. The decisions of choosing the temperature sensors, relative humidity sensors and thermocouples are all based on a weighting of cost and performance. We decided to use thermistors with an accuracy of 0.1 as our temperature sensors although they may not of sufficient accuracy for some cave experiments, and although that decision deviates from the performance specifications. That decision was based on a limited budget and on the fact that this is a preliminary experiment which is focused on proving the idea of a ZigBee WSN-based platform for a cave environment monitoring system. For the same reasons, we chose relative humidity sensors with an accuracy of 3.5%. I chose to use two operational amplifiers throughout the experiment which also fit the budget: The OP-27 because of its low noise, and the OP-90 because of its low power consumption.
3.3.2 Temperature and relative humidity sensors

In this section, I describe the essential components of the external sensor board used with the ZEDs. On this external sensor board, we have a thermistor (as a temperature sensor) and a relative humidity sensor. Figure 3-8 shows a picture of the ZED with an external sensor board on top.

Figure 3-8: ZED with an external sensor board on top

Figure 3-9: the circuit to provide +3.3V and +5V regulated power supply
Figure 3-10: the circuit of a voltage divider for the thermistor

Figure 3-11: the circuit of a voltage divider for the relative humidity sensor

As shown in Figure 3-9, the external sensor board holds circuits that provide +3.3V and +5V regulated power supply to the thermistor circuit and the relative humidity circuit. The chip TPS7150Q and the chip TPS7133Q are +5V regulator and +3.3V regulator respectively. Their operations are the same. Pin 3 of these two voltage
regulators are connected to a 9V battery as a power supply. Voltage outputs are available from Pin 6 only when Pin 2 is driven low. In order to save power, we want the voltage outputs only for a short duration only when measurements are performed, pin 2 should be drive high most of the time. On the other hand we do not want the digital I/O (DIO) programmed high for most of the time, because when the DIO is programmed high, it consumes more power. An NPN type transistor is used here to solve this conflict. When DIO is low, the transistor is in cutoff mode and no current flows from the collector to the emitter, thus providing a high input to pin 2. When the DIO is high, the transistor is operating in saturation mode with a small voltage drop from the collector to the emitter, thus providing pin 2 with a low signal.

3.3.2.1 The thermistor and related circuit

Thermistors are thermally sensitive resistors which exhibit a large, predictable and precise change in electrical resistance when subjected to a corresponding change in body temperature. Because of their predictable characteristics and their excellent long term stability, thermistors are considered to be excellent sensors for many temperature measurement applications. When a thermistor is used in a temperature measurement application, one thing to make sure is that the power dissipated within the device is not sufficient to cause self heating. When there is no self heating, the body temperature of the thermistor will follow the temperature of the environment. Figure 3-10 shows the circuit for the thermistor, through the voltage divider a change of the resistance of the thermistor caused by temperature change of the environment will result in a voltage change to the positive input of the voltage follower. The voltage follower is used to present high input
impedance to the voltage divider and to provide a low output impedance to the ADC input pins so that the voltage between the resistor R3 and the thermistor will not be affected by reading it. The input impedance of the ADC input pin is 7KΩ [21]. The OP-90 Op-amp is used because it consumes very little power and is able to operate in a unipolar, rail-to-rail mode (i.e. the output can go all the way to the supply voltages at the high and low end). On the other hand, OP-90 also has more noise. But since we are not dealing with very small voltages in this case, the OP-90 is a better choice than a low-noise op-amp such as the OP-27 which consumes more power and does not function in a unipolar mode. The noise explanations and calculations of the OP-90 and OP-27 are shown in section 3.3.2. The output of the voltage follower is equal to the positive input voltage. The output of the voltage follower is then passed directly to an ADC input pin of the SRB which is discussed in the Section 3.1.

The thermistor used in this project PS503J2 has a thermal constant of 10 sec max in still air. A resistance and temperature plot is shown in Figure 3-12. Since we use this thermistor in a cave environment, the estimated temperature range is from 5 °C to 15 °C which corresponds to a thermistor resistance range from 166960 Ohms to 78560 Ohms. The temperature coefficient of R3 is 350ppm/°C which contributes approximately 0.1mV voltage error, in this particular circuit, to the input of the ADC pin when there is a whole 1 degree temperature change. Given the digitization step of the ADC is 3mV and the output voltage changes about 36mV per degree, this error is small enough to be ignored in the voltage calculations.
Figure 3-12: Resistance vs. temperature plot for PS503J2 thermistor

To maximize the temperature measurement resolution, I need to maximize the output voltage range in the corresponding temperature range. If I want to maximize the output voltage range when the temperature is in 5°C to 15°C, assign $R_{t1}$ to be the resistance of the thermistor at 5°C and $R_{t2}$ to be the resistance of the thermistor at 15°C.

Then, the output voltage $V_{o1}$ at 5°C is

$$V_{o1} = 3.3 \times \frac{R_{t1}}{R_3 + R_{t1}}$$

The output voltage $V_{o2}$ at 15°C is

$$V_{o2} = 3.3 \times \frac{R_{t2}}{R_3 + R_{t2}}$$

Now, we want to find the value of $R_3$ to maximize $V_{o1} - V_{o2}$. Since $R_{t1}$ and $R_{t2}$ are known, and fixed, $R_3$ can be found out by calculating
\[
\frac{d (V_{o1}-V_{o2})}{d R3} = 0
\]

In this project I choose the temperature range of 5°C to 15°C. The resistance range of the thermistor corresponding to this temperature range is from 126960 Ohms to 78560 Ohms. After the calculation, the best value of resistor R3 is 100KΩ. This results in a maximum of 0.358V output voltage change in 5°C to 15°C temperature range, or a response of 36 mV/°C. Since the digitization step of the ADC is 3mV, the detectable temperature variation level is then \(3mV / (36 mV/°C) = 1/12 °C\). Calibrations of the thermistor-based temperature sensors are discussed in Chapter 5.

3.3.2.2 The relative humidity sensor and related circuits

The HIH-4000 relative humidity (RH) sensor used here has a linear voltage response versus percentage relative humidity change. With a typical current draw of only 200µA, the HIH-4000 RH sensor is ideal for low drain, battery operated systems. The sensing element’s multilayer construction provides good resistance to most application hazards such as wetting, dust, dirt, oils and common environmental chemicals. These sensors have an accuracy of ± 3.5% RH and they are individually calibrated at 5V DC power supply.

As shown in Figure 3-11, a voltage divider and a voltage follower is used together with the relative humidity sensor to make the output of the sensor suitable for the ADC input of the SRB device. The voltage output range goes roughly from 1V at 0% RH to 4V at 100%RH which is beyond the range of the ADC input range. So, this voltage divider is
used here to cut the output range of the RH sensor to the range that is suitable for the ADC. Again, I use a voltage follower to present high input impedance to the voltage divider and low output impedance to the ADC inputs.

### 3.3.3 Thermocouples and related designs

In this experiment, I use thermocouple-based sensors to measure air turbulence. As mentioned earlier in section 3.1, I installed five thermocouples in a single row, vertically spaced approximately evenly on a 14 foot high tower. One NCB is attached to the tower and has two main functions. Firstly, it acts as a router to relay reports from other nodes to the ZigBee Coordinator. This function is performed automatically in background (and this requires the radio to be left on all the time to listen). Secondly, it reads the voltage outputs from these five thermocouples at a sampling rate of 30 Hz and reports them to the ZigBee Coordinator. In this subsection I detail all the issues related to the thermocouples. I will introduce the working principle of thermocouples first. Then I will discuss the reason why we are using thermocouples and what we are actually measuring when we use them. The circuits with the thermocouples and detailed design issues are also described here.

Thermocouples are based on the principle that when two dissimilar metals are joined at one end, a predictable net thermoelectric voltage between the open pair will be generated which relates to the difference in temperature between the measuring junction and the reference junction (connection to the measuring device). Figure 3-13 shows a
picture of the thermocouple used in this project, the measuring junction is on the left hand side of the picture.

![Picture of Thermocouple](image1.png)

**Figure 3-13:** The Thermocouple used in this project

![Diagram of Thermocouple](image2.png)

**Figure 3-14:** Thermocouple is placed inside of a 1 foot long PVC pipe with the 1st stage amplifier

![Diagram of First Stage Amplifier](image3.png)

**Figure 3-15:** The configuration of the first stage amplifier circuit for the thermocouple
The total amplification factor is chosen based on an estimation of temperature variation range of the measuring environment and the input limit of the ADC. For instance, in our case, the estimated air temperature variation range is ±5 °C which is made based on a previous experiment [22] and also because we expected the cave temperature fluctuation is much smaller than that. The ADC module input range is 0-3V. Since the thermocouples I used have a response of 40 μV/°C, if I amplify the output from the thermocouple by a factor of about 8000, the output response is approximately 0.3V/°C. So, within the input range of the ADC (0-3V), the measurable air temperature variation has a range of about 10 °C (with a resolution of about 0.01 °C). A two-stage amplification configuration is used in my experiment which is because it is easier to change the amplification factor later by changing the amplification factor on the second stage, just in case. As shown in Figure 3-14, the thermocouple is placed in a 1 foot long PVC pipe with the first stage amplifier. The second stage amplifier is placed on the tower which I will discuss later. As shown in Figure 3-15, the first stage amplifier is a non-inverting amplifier which made use of an OP-27 Op-amp. I use an OP-27 in the first stage non-inverting amplifier because it has lower noise than OP-90, although OP-27 consumes more power than OP-90. Lower noise is necessary when amplifying small signals with a large amplification factor which is about 4000 times, in this case. OP-90 has a noise of 40 nV/√Hz from 10 Hz to 30 Hz and a linear noise from 300 nV/√Hz to 40 nV/√Hz over the bandwidth from 0 Hz to 10 Hz [23]. I calculated the total RMS (root mean square) input noise of interest over the bandwidth (0-30 Hz) to be 0.7μV. The amplification factor used with the first stage amplifier is about 4000 which will generate an output noise of 2.8 mV. After amplified 2 times by the second stage amplifier
(discussed later), a 5.6mV noise will be presented at the input of the ADC which is larger than the digitization step of 3mV. However, one can improve the performance by averaging multiple measurements before reporting. As for OP-27, it has a noise of only 3nV/√Hz over the bandwidth (0-30 Hz). The total RMS input noise generated by OP-27 is calculated to be 16 nV. After it is amplified 8000 times, it is a noise of 0.128 mV to the input of the ADC which is much smaller than the noise generated by OP-90. This is why I choose OP-27 over OP-90 in the first stage amplifier. To get more accurate measurements, we may consider ADCs with higher bits (the ADC on the ZigBee chips are unipolar and 10-bit), such as a 12-bit ADC. With the same measuring range (0-3V), the digitization step is 0.73 mV (0.0025 °C resolution in our case) when a 12-bit ADC is used.

As shown in Figure 3-16, an Offset Nulling configuration is also implemented on the OP-27 used in the first stage amplifiers which is not reflected in Figure 3-15. This Offset Nulling is used here to make sure when the output of the thermocouple is 0 V, the output of the amplifier is also 0 V or at least very close to, which made it easier to adjust the output voltage to the range suitable for the ADC module on NCB. Since the output range of the thermocouple will go from negative to positive, and so does the output of the first stage amplifier, I have to shift the range of the input to the ADC up to a positive range in 0-3 V. It is ideal to let the null point of the input voltage to ADC be 1.5 V. To do this, we have a 0.75 V voltage reference to the first stage amplifier which is shown in Figure 3-15. Following is the derivations: assuming the output voltage of the thermocouple is Vt, the output voltage of the first stage amplifier is V0 then V3 = 0.75 + Vt, we also know V2 = V3, then we have V2 = 0.75 + Vt, so we have
\[
\frac{V_2 - 0.75}{R_2} = \frac{V_{o1} - 0.75}{R_2 + R_3}
\]

\[
\frac{0.75 + V_t - 0.75}{R_2} = \frac{V_{o1} - 0.75}{R_2 + R_3}
\]

\[
\frac{V_t}{R_2} = \frac{V_{o1} - 0.75}{R_2 + R_3}
\]

\[
V_{o1} = V_t \left(1 + \frac{R_3}{R_2}\right) + 0.75
\]

We notice that the VO1 is shifted up by 0.75 V. After amplified with a factor of 2 by the second stage amplifier, we have a null voltage of 1.5 V (when Vt = 0 V). The exact location of the null voltage point is not important because what we are only interested in is the difference between the output voltage value and the null voltage point. I chose 1.5 V as the null point because I want to have equal valid measuring ranges on both sides (up to 3.0 V and down to 0 V). The circuit to provide 0.75 V reference is shown in Figure 3-17. The reason why I use a potentiometer Rp1 is that I need this voltage output be flexible so that when I change the amplification factor on second stage amplifier, the circuit can still give the ADC the null voltage of 1.5 V. Since the 1.5 V does not have to be precise, it is safe to use the potentiometer Rp1 in the circuit (the resistance of a potentiometer may not be precise enough in some other situations). The second stage amplifier is shown in Figure 3-18. Voltage output of the second stage amplifier will be fed in to the ADC.
Figure 3-16: Offset Nulling Circuit for the OP27 Op-amp

Figure 3-17: The circuit used to provide the 0.75V reference

Figure 3-18: The second stage non-inverting amplifier
After explaining the circuits for the thermocouples, the question now is “what exactly are we measuring?” Since I soldered the thermocouples to the PCB board that holds the first stage amplifier at the reference junction (connection to the measuring device), as shown in Figure 3-19, this reference junction reflects the temperature of the PCB board. The temperature of the PCB board is considered to be the average of the environment temperature over a time scale which is at least many minutes. Meanwhile, the measuring junction of the thermocouple, as shown in Figure 3-20, is sticking out of the PVC pipe and is exposed to the surrounding air of the environment directly. Since the
measuring junction is extremely thin and responds rapidly to the temperature changes in the environment, on time scales of ten milliseconds or less. So what we are really measuring out of the thermocouple is that the actual temperature difference between an averaged value of the environment and the instant temperature of the air temperature at the measuring junction.
CHAPTER 4 DISCUSSIONS OF NETWORK LIMITS

In this chapter I discuss two aspects of the system performance: 1) ZigBee Network Size Limits, 2) ZigBee Network Lifespan Limits.

In section 4.1, I discuss factors that will restrict the Network Size. This includes: ZigBee Network inherent Limits, ZC and SBC processing ability Limit, Network Latency Tolerance, and Network Bandwidth Limit.

In Section 4.2, I discuss factors that will restrict the Network Lifespan. This includes: Power consumptions of all the components in the Network, and Memory consumption of the storage.

4.1 ZigBee Network Size Limits

4.1.1 ZigBee Network inherent Limit

In a ZigBee wireless sensor network, at least for the current ZigBee Specification [12], devices use the 16-bit addressing. So theoretically, without considering other factors a ZigBee Network can handle up to $2^{16}$ nodes (65536). Moreover, since a routing table must be maintained on each router, which in turn has limited memory capacity, a limit of 20 children is imposed in the stack profile. Each ZR has by default the capacity for 6 ZR
children and 14 ZED children. The depth of Zigbee Wireless Sensor Network is the maximum number of hops a message must make to get from a source to a destination. The maximum depth is defined in the stack profile to be 15.

4.1.2 ZC and SBC Processing Ability Limit

Figure 4-1: A Timing Measurement, Time 1: This is the time the ZED has just finished transmitting a packet out over-the-air, Time 2: This is the time the ZC started to write the packet on the data line of the serial cable to the SBC.

Figure 4-1 shows data from a test to measure the time delay for a data packet to travel from a ZED to the SBC. In the test I arranged a ZigBee Network with only 2 Nodes: one ZC and one ZED. The ZED was programmed to transmit a report to the ZC once every 8 ms. Probe 1 (orange trace or trace 1 in the figure) was connected to the data
line of the serial port of the SBC. Probe 2 (blue trace or trace 2 in the figure) was connected to one DIO on the ZED.

I programmed the ZED to raise the DIO to high once the ZED finishes scheduling a report to transmit. During the time period when blue line (DIO) is high, the ZED is performing a 2 channel 10-bit ADC conversion. Once the ADC is done, the blue line (DIO) is programmed to drop to low. Then the program in the ZED will wait for a timer to expire (8ms) and when the timer expires the ZED starts to schedule another report to transmit out with the content of last ADC conversions. At the instant when the ZED is done with scheduling the report to transmit, the DIO is driven high again and this process repeats over and over again. The data written to the serial data line between Time 0 and Time 1 is the data received from the last ZED transmission (the one scheduled to transmit out at Time 0). The data written to the serial data line between Time 2 and Time 3 is the data scheduled to transmit out by the ZED at Time 1. We can convince ourselves of this by increasing the reporting interval to be much larger than 8 ms. Thus, Figure 4-1 shows that after the ZED has just finished scheduling the data to transmit out it takes about 7.5 ms for the data to arrive at the serial data line. In other words, during this 7.5 ms time period, the report went through the transmitter of the ZED, travelled over-the-air, went through the receiver of ZC, passed through the MC13213 chip (SPI communication link between the radio and Micro control unit), and finally arrived on the serial data line. Other researchers have measured a 7 ms hop time [24], which is consistent with my observation. At the time instant Time 2 (first black dashed line), we see the data line of the serial cable is driven with data by the ZC and it took 1ms for the ZC to finish writing the data on the serial data line. Since the ZC is writing a 14-byte data packet (112 bits),
and the Baud Rate of the serial communication between the ZC and the SBC is set to 115200, theoretically it takes 0.97ms to finish transmitting which matches with the time we observed between the two black dashed lines in Figure 4-1.

To make it clear, the 7.5 ms period should cover these following three activities:

a) After the report has been scheduled to transmit by the program code, the ZED processes this and sends it out over-the-air;

b) Data packet travels from ZED to ZC over-the-air;

c) ZC catches the data packet, processes it and then start to transmit on serial line to the SBC.

A safe guess is that only one data packet can be transmitted during this 7.5 ms period because either the receiver on the ZC is not ready and still busy preparing the serial port transmission or because the airwave is busy with previous packet. Since the report itself is 14 bytes, and according to the ZigBee protocol, an overhead of 33 bytes (264 bits) is added to every message sent out over-the-air when the message itself is under 104-byte size limit (832 bits) [12]. In our case, the ZC received this 47 bytes of data during the 7.5 ms. In other words, the practical data rate we see in this case is (47 bytes) * (8bits/byte) / 7.5 ms = 50 K bits/sec. This is under the limit of the ZigBee protocol Bandwidth and the best we can do for the time being. So, the bottle neck in terms of data rate is constrained here with 50K bits/sec. On the other hand, if we assume the data link can achieve the data rate of 250 K bits/sec. then this 47-byte (376 bits) data packet (data plus overhead) should occupy the frequency channel for at least 1.5 ms.
Although normally a ZigBee network cannot achieve this data rate according to other researchers’ calculation and observations [24].

I find that if I continually keep feeding the ZC packets every 7.5 ms the SBC time stamps keep pace and no packets are lost. Thus, the time it takes the SBC to process a packet is at most 7.5 ms and it is able to keep up with the maximum data rate of the network.

If we assume all the ZEDs and ZRs report once every minute and we have in total n nodes (ZEDs and ZRs) in a this ZigBee Network, which means the ZC and the SBC expect to see an average of n reports from the network every 1 minute. This leads to the conclusion that the ZC and SBC can only handle n = 133/sec * 60 sec = 7980 Nodes. This limit is under the limit of 65536 nodes defined by ZigBee protocol (section 4.1.1). If we need the network to sleep 99% of the time and to finish all data reporting in just 0.6 second (1% of the time), then we can handle at most n = 133/sec * 0.6sec = 80 Nodes. If we need the network to sleep only 95% of the time and to finish all data reporting for every minute in 3 seconds (5% of the time), then we can handle at most n = 133/sec * 3 sec = 399 Nodes.

What we also noticed and learned from this is that if we gather multiple reports from one node and put them together into one single message (less than 104 bytes) and transmit them out at one time, with only one header of 33 bytes instead of having one header for each report, we should increase the throughput of the network, thus allowing for a larger network, or larger sleep time for a given network size.
4.1.3 Network Bandwidth Limit

Since on any given channel, in a given location, only one radio may be transmitting at one time, a dense network with lots of traffic could end up interfering with itself. The total network traffic at any time should not exceed 250K bits/sec. As mentioned before, in the ZigBee protocol, an overhead of 33 bytes (264 bits) is added to every message sent out over-the-air when the message itself is under 104-byte size limit (832 bits) [12]. The overhead includes all over-the-air headers and information [12]. Assuming the data reports from ZEDs and ZRs are 14-byte each, then the actual packet sent out over-the-air is 47 bytes each. According to the limit constrained by the process ability of the ZC and SBC (in section 4.1.2), the ZigBee network can only handle 133 reports per second which is $133 \times 47 \times 8 = 50K$ bits/sec. Because this is under the bandwidth limit of 250K bits/sec, the bandwidth will probably not be a bottleneck of this specific ZigBee Network built in this project.

4.1.4 Network Latency Tolerance

For the following four reasons, we do not expect the data from the reports of all nodes are real-time:

1) the Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) which is used in ZigBee protocol for devices to share the channel in a ZigBee Network;

2) some data packets are relayed by the ZigBee Router and it causes more delay;

3) Timestamps are labeled to data after the data are received by the SBC. Even if a ZED transmits directly to the ZC, the data are time-stamped more than 7.5 ms late by the SBC. This time delay varies with different network topologies.
4) It takes time for the ZED or ZR to create the data packet after measuring.

As the depth of the network increases, it takes more time for the data to be relayed from measuring nodes to the ZigBee Coordinator. The question now is how much Network Latency we can endure. I explain how network latency tolerance constrains the size limit of the network with the following simple example.

If we assume a network with the topology shown in Figure 4-2, each ZR has two ZED children and all the ZRs are in a linear topology (single branch tree), the depth of the network is $m$. The furthest ZEDs are the two with the ZR labeled with $(m-1)$. Assuming each ZED and each ZR reports once every minute, then the total number of hops $n$ should take place in 1 minute for all the reports get to the ZC should be:

$$n = (2 \times 2 + 1 \times 1) + (2 \times 3 + 1 \times 2) + \cdots + [2 \times (m-1) + 1 \times (m-2)]$$

$$+ [2 \times (m) + 1 \times (m-1)]$$

$$= 2[2 + 3 + \cdots + (m-1) + m] + [1 + 2 + \cdots (m-2) + (m-1)]$$

$$= 2 \left[ \frac{m(m+1)}{2} - 1 \right] + \frac{(m-1)m}{2}$$

$$= \frac{2m^2 + 2m - 4}{2} + \frac{m^2 - m}{2}$$

$$= \frac{3m^2 + m - 4}{2}$$

We do expect this kind of scenario will happen: more than one hop happens at the same time in a network at difference data links out of radio range with each other.
Actually, in a practical ZigBee Network, all the Nodes are normally spread out relatively far from each other to cover more space and so that every device has only a few other devices within range but transmissions from a single device do not blanket the entire network and devices out of range of each other may carry on independent communications, thereby promoting spatial reuse [25]. So, here, if we guess a 60% of the hops will happen singly and will actually consume time, and according to a conservative time estimation of 15ms/hop [26]; if we want to maintain a maximum network latency no more than 5% of 1 minute (report frequency) which is 3 seconds, in other words we need to get all the data hops done within 3 seconds. Then we have

\[
\frac{3m^2 + m - 4}{2} \times 60\% \times 15 \text{ ms} \leq 3 \text{ s}
\]

The result is m should not be larger than 14. If we need to get all the transmission done within 0.6 seconds (1% maximum network latency), we have m should not be larger than 6.

The preceding is only a simple example of a specific network topology with assumptions and guesses. In general, the performance of the network depends strongly on the exact network topology and node placement, including collisions, and in particular collisions of packets from nodes which cannot see each other but are both connected to the same router. More precise estimates of network performance will require computer simulations and verification of these computer simulations with packets sniffers. Here it only shows how the network latency will affect the network size limit.
A total number of \( \frac{3 \cdot m^2 + m - 4}{2} \) hops for the whole network

\[
\begin{array}{c}
1 \\
2 \\
\cdots \\
m-2 \\
m-1
\end{array}
\]

Figure 4-2: An example of a practical ZigBee Network

When the network size is large, there are more obstacles for a report to be transmitted to the ZigBee Coordinator fast enough to maintain the precision of the timestamps added, thus more network latency is expected. As a result, when the network size is large, timestamps should be considered to be prepared and added to the data at each node once the data are ready to be sent out to alleviate Network Latency. Adding timestamps at each node in the network is actually a timing synchronization issue. Several protocols or algorithms have been developed explicitly for Wireless Sensor Networks (WSNs) in recent years, such as the Reference Broadcast Synchronization (RBS) [27], Timing-sync Protocol for Sensor Networks (TPSN) [28], and the Flooding Time Synchronization Protocol (FTSP) [29, 30].
4.2 ZigBee Network lifespan Limits

4.2.1 Power Consumption

Table 4.1 is a list of all the devices we used in this wireless sensor network with their power consumptions and a description of their functionalities. Detailed hardware descriptions were presented in Chapter 3.

Table 4-1: A list of the hardware used in the project

<table>
<thead>
<tr>
<th>Items</th>
<th>Quantity</th>
<th>Power consumption</th>
<th>Duty cycle</th>
<th>Function or purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Busy (mA)</td>
<td>Idle (mA)</td>
<td>(%)</td>
</tr>
<tr>
<td>ZigBee MC13213-SRB boards(ZED)</td>
<td>4</td>
<td>16@9V</td>
<td>10@9V</td>
<td>1.5% Network End Devices</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Not actually in deep sleep)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZigBee MC13213-NCB board(ZC)</td>
<td>1</td>
<td>61@9V</td>
<td>No Idle</td>
<td>1 ZipBee Coordinator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZigBee MC13213-NCB board (ZR)</td>
<td>1</td>
<td>57@9V</td>
<td>No Idle</td>
<td>1 Network Router</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single Board Computer</td>
<td>1</td>
<td>89@12V</td>
<td>No Idle</td>
<td>1 Data Storage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External Sensor boards</td>
<td>4</td>
<td>1@9V</td>
<td>0</td>
<td>1.5% Measure temperature and relative humidity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tower with 5 thermocouples</td>
<td>1</td>
<td>37@+/-9V</td>
<td>0</td>
<td>20% Measure Air Turbulences</td>
</tr>
</tbody>
</table>

For now, we haven’t tried to implement putting the entire network into sleep, so the ZC and the ZR are running with radio left on all the time with around a 60 mA current drawing from their power supply. I prepared several battery packs with 6 parallel connected 9V batteries each. One 9V battery normally has a capacity of 600 mAh.

Or 9V*600mAh = 5400mWh.

Or 5400mWh * 3600 sec/hr *(1W/1000mW) = 19440 Ws = 19440 J.
So, six of the 9V batteries together should have 19440 J*6 = 116640 J.

The energy used by a ZC or a ZR per day is then 9V *60mA*86400 sec/day = 46656 J/day. So the pack of 6 9V batteries should allow a ZC or ZR to operation for about 116640/46656 = 2.5 days.

When a ZED is idle it draws about 10 mA (Table 4-1). When a ZED is Idle, it still draws about 10 mA. That is because the ZEDs are not in deep sleep. The ZED will not go to deep sleep if there is any timer still running. According to the data sheet provided by Freescale, when the ZED is in deep sleep, it needs a so called Event Timer to wake it up, not normal timers. But I have been unable to find a way to implement event timers, so in this work I use only regular timers and light sleep. When a ZED is in light sleep, the radio is turned off and the whole board consumes a 10 mA current. When the ZED wakes up, it will transmit out a report from last ADC readings and perform another ADC reading and go to light sleep again, and the process continues. When the ZED is up, the radio is turned on, with a 16 mA current drawing from its power supply. The ZED is also programmed to drive a Digital I/O (DIO) high to turn on the external sensor board (refer to Section 3.3.1). When the external sensor board is turned on, it consumes a current of 1mA. The ZED waits approximately 150 ms to allow the external sensor board to stabilize, and then after the ADC is done reading results ZED goes back to light sleep. If we set the ZED to report every 10 seconds, the duty cycle of the ZED and the external sensor board is 1.5%. If the ZEDs are set to report every 1 minute, the duty cycle is 0.25%. If we can implement a deep sleep, the ZEDs should in principle be able to operate for years without replacement.
If we set the ZEDs to report every 1 minute with a duty cycle of 0.25%, without deep sleep, one ZED with one external sensor board spends 86400 sec/day * 0.25% = 216 seconds a day in busy mode (16mA + 1mA = 17mA). The energy used by a ZED together with an external sensor board for data reporting per day is then 9V * 17mA * 216 sec/day = 33 J/day. The energy used by the ZED together with an external sensor board for staying idle (without deep sleep) for one day is then 9V * 10mA *(86400-216) sec/day = 7756 J/day. So, one ZED together with the external sensor board uses up 33 + 7756 = 7789 J/day. A battery pack of three 9V batteries parallel connected together should work with them for about 19440*3 / 7789 = 7.5 days. If the ZED can go to deep sleep, report once every 1 minute with a duty cycle of 0.25%, the ZED with the external sensor board will only use 33 + [9V*2uA*(86400-216)] = 33 + 1.55 = 34.55 J/day. Then the ZED with the external sensor board should run together for about 19440*3/34.55 = 1688 days with a battery pack of three 9V batteries.

For the SBC, we use a 12V car battery as a power supply. The car battery is a deep-cycle marine battery. The capacity stated on the battery is 85 Ah. But with only a trickle 89 mA current draw (compared to car current usage), we would expect the capacity is more than 85 Ah. Assuming total energy it has is 12V*85000 mAh*3600sec/hr*(1W/1000mW) = 3672000 Ws = 3672000 J. If the SBC keeps running all the time, the energy that the SBC consumes is 12V*89mA*86400sec/day = 92275 J/day. So, the SBC can run with the 12V car battery at least for 3672000/92275 = 39 days. This is a lower limit. The car battery should allow the Single Board Computer to run way more than 39 days.
For the Thermocouple Tower, we use a battery pack with six 9V batteries as power supply, three batteries parallel connected together as +9V and the other three parallel connected together as -9V. If the Thermocouple Tower is powered on with a duty cycle of 20%, the energy used by the tower for one day is then 9V \( \times 2 \times 37mA \times 86400 \times 20\% = 11508 \text{ J/day} \). With a battery pack of six 9V batteries, it can run for about \( \frac{19440 \times 6}{11508} = 10 \text{ days} \).

4.2.2 Memory Consumption

In this Wireless Sensor Network, we use a 2G Bytes SD card on the SBC to store the data reported from all Nodes. Data from different nodes are saved in different files in text format. If instead, we write files for data in binary format, memory space will be saved.

With text file format, each report from a ZED takes a 4-byte memory space for the Temperature Sensor (thermistor based) reading and another 4-byte for the relative humidity sensor. For each report from an End Device, we use 8-byte memory space plus a 16-byte space for the timestamp which is together 24-byte memory space. Given that the four End Devices report once every 10 seconds each, for 24 hours, we have 810K bytes of data stored to the SD card on the SBC. Each report from the Thermocouple Tower by the Network Router takes a 4-byte memory space for each thermocouple reading. For 5 thermocouples, each report from the tower, we use 20-byte memory space plus a 16-byte space for the timestamp which is together a 36-byte data. Given that the Router reports data at the rate of 30 Hz, for 24 hours, we have 91M Bytes of data stored to the SD card. Altogether, we have about 92 M bytes data for 24 hours of recording. For
a 2G Bytes SD card, roughly 1.5G Bytes available, we can run the Network for about 16 days, assuming battery life is long enough.

If instead, we write data in binary format, and assuming the tower is transmitting reports at 30 Hz with 12-byte data for each report from the Thermocouple Tower, 30 Mbytes memory space for a day. Also, if we write binary format files for the ZEDs, it is a 6-byte data for each report for a ZED. Assuming each of the four ZEDs reports once every minute. Then we have 34 k bytes for four ZED recording for a day (12 Mbytes for a year). So, together with the memory space used for storing reports from the Thermocouple Tower, we have around 31 Mbytes for a day. For a 2G Bytes SD card, roughly 1.5G Bytes available, we can run the Network for about 49 days, assuming battery life is long enough.

If we set the ZR to process the data from thermocouples to 1-minute Root Mean Square (RMS) values and to transmit that to the ZC instead of having the complete data set transmitted, assuming writing data in file with binary format, it only takes 17K bytes of memory space to store the data from the Thermocouple Tower for a day and only about 6M bytes for a year. So far we have stored the complete data set in the event because we might need it in the future for analysis. Since computing RMS is how we usually post-process these data anyway, in order to conserve power, bandwidth, and memory it may become necessary to store compressed data products, for example 1-minute RMS values.

So, if we set the ZEDs to report every 1 minute and transmit only the 1-minute RMS value of the thermocouple readings from the ZR and write data in binary format, it
takes only about 18M bytes of memory space to store them for a whole year. Moreover, both network bandwidth and power consumptions are also saved from this action which means we can potentially make the network much larger, or save power and storage to allow the network to operate for a longer time.
CHAPTER 5 EXPERIMENT AND RESULTS

This chapter presents the experiment carried out in the Junction Cave at El Malpais National Monument, New Mexico. Results from data collected by the Wireless Sensor Network are also presented.

5.1 Experiment Setup in the Junction Cave

The experiment was carried out in the Junction Cave at El Malpais National Monument, New Mexico. Data were collected for over 24 hours, from the evening of July 10, 2009 to the evening of July 11, 2009. The Junction Cave is part of a lava tube that was created by volcanic forces probably thousands of years ago [31], resulting in a dead ended cave, that is more than 300 meters long and moderately rugged. Throughout the cave, with the exception of a relatively large room at the end of the cave, there are no spaces in the cave that is taller than 5 meters. A spot, roughly 45 meters from the entrance, on the left hand side of the tunnel (looking into the cave), was selected to place the ZigBee Coordinator (ZC), the SBC and their batteries, shown in Figure 5-1. Together they are considered to be the Base Station of the ZigBee Wireless Sensor Network. The spot for the ZC was selected also because it is on top of a rock pile with a view of a large section of the cave which made it easier for us to place the ZEDs and the Thermocouple Tower with line-of-sight to the ZC. The Thermocouple Tower was place approximately half way between the entrance and the base station, which was roughly 23 meters from
the entrance. The location for the Thermocouple Tower was selected for two reasons: 1) this was the tallest spot in the first few hundred feet from the entrance; 2) We wanted to put the tower not too far from the entrance in order to measure the expected greater turbulence near the entrance. The tower is shown in Figure 5-2.

![Image](image1.png)

**Figure 5-1:** The ZigBee Coordinator, the Single Board Computer and their batteries placed together as the Base Station

![Image](image2.png)

**Figure 5-2:** The tower is placed roughly 75 feet from the entrance. Light from the entrance is visible in this picture.
We also deployed four ZigBee End devices (ZED) in the cave. Each of them has the capability to measure both temperature and relative humidity. All the ZEDs were programmed to report their sensor readings once every 10 seconds. ZED 4 was placed on a rock at a high location inside the cave, close to the cave entrance (about 5 meters to the entrance). ZED 1 was placed on a rock right above the Base Station. ZED 3 was placed in a depression on the right side of the cave (looking into the cave), about 45 meters from the cave entrance and 12 meters from the Base Station. ZED 2 was placed about 15 meters deeper in the cave. ZED 1 did not have direct line of sight to the ZC, but they were placed really close (about 1 meter away from each other). ZEDs 2 and 3 had good line of sight to the ZC. ZED 4 did not have direct line of sight to the ZC and its messages were relayed through the ZR which was attached on the Thermocouple Tower, as shown in Figure 5-2. We will reference ZED 1 location as ‘High’, ZED 2 location as ‘Deep’, ZED 3 location as ‘Low’, and ZED 4 location as ‘Entrance’ in the following sections. The whole network layout is shown in Figure 5-3.

Figure 5-3: Whole network layout in the Junction Cave
5.2 Results and Analysis

5.2.1 Network Performance

The experiment was set up on July 10, 2009. The network started collecting data at about 19:00 that day. Then we left for the whole night. We came back to the Junction Cave around 08:15 in the morning next day. Then we went into the Cave to see if everything was still working. The following are what we found out:

a) All batteries were still in pretty good voltage range, all 9V battery packs still had more than 8.7 Volts, and the 12V battery for the SBC still got enough voltage;

b) Data from the Thermocouple Tower were still coming in to the SBC;

c) The ZC had lost connections with all four ZEDs.

After we realized that all ZEDs were all off line, we went to every ZED location and reconnected them back to the network. The ZEDs had been tested with continuous connection to the ZC for more than 48 hours in the lab. The reason why the ZEDs all dropped out of the network is unknown. But it may because the wireless signals somehow got attenuated too much in a cave environment. In the future, we should implement algorithms in their program to have them reconnect once they lose connection or have them be able to periodically auto-reconnect. After we came back to the lab, I found out these:
a) ZED 1 dropped off line about 3 hours after set up (ZED 1 was the one placed right above the Base Station and did not have a line of sight to the ZC);
b) ZED 2 dropped off line about 4 hours and a half after set up (ZED 2 was the one placed about 50 feet away from the Base Station, further down the tunnel);
c) ZED 3 dropped off line about 3 hours after set up, and after it dropped off, about an hour later, ZED 3 connected back to the ZC again by itself, after that, it dropped off line again 2 hours later (ZED 3 was the one placed in a depression on the right side of the cave (looking into the cave), 40 feet away from the Base Station);
d) ZED 4 dropped off line about 5 hours and 20 minutes after set up (ZED 4 was the one placed near the entrance close to the ceiling of the cave and uses the ZR on the Thermocouple Tower to relay its reports);
e) The ZR that was transmitting the Thermocouple Tower data stayed on line all the time throughout the whole experiment;
f) Since we went into the cave for several times to check if the ZEDs had dropped off line and had them reconnected back in to the network every time they dropped, it seemed they dropped off line within 2 to 3 hours after connected back with good battery conditions.

In section 5.2.2 I discuss the data from the Thermocouple Tower, and in section 5.2.3 I discuss the data from the four ZEDs.

5.2.2 Data from Thermocouple Tower

1-minute standard deviations of the five sensor signals are shown in Figure 5-4, with the time period from 19:23 Jul 10, 2009 to 19:43 Jul 11, 2009. The panels are arranged in same order as the thermocouples arranged on the tower from top to bottom. One thing to mention is that when we came back to the Junction Cave second day in the
morning, we noticed the ground in that area was wet. So it must have rained overnight and apparently just finished not long before we arrived.

Figure 5-4: 1-minute standard deviations of the five sensor signals

The thermocouple sensors were installed in a single row vertically on the Tower.

Table 5-1 gives the exact location of the sensors above the floor of the cave.

Table 5-1: Sensor heights above the cave floor

<table>
<thead>
<tr>
<th>Sensor Number</th>
<th>Height (Feet)</th>
<th>Height (meters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>3.96</td>
</tr>
<tr>
<td>2</td>
<td>10.5</td>
<td>3.20</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>2.43</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1.22</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.61</td>
</tr>
</tbody>
</table>
The top three thermocouples started to see many spikes overnight, whereas the bottom two did not. One likely explanation for this is that because the Junction cave is a dead ended cave and the air temperature in the cave is really low (temperature recordings are shown later), and the bottom two sensors were somewhat sheltered in a depression surrounded by cold stable air, if there is any hot air coming in, hot air causes more turbulence in the upper space than in the lower space. The spikes seen from the top three thermocouples overnight died out around 06:00 AM although they last about one hour longer in the top sensor data. We speculate that the spikes overnight seen from the top three sensors must have something to do with the rain. Then there is a period of very calm condition where the traces are very narrow which can be seen from all five sensors. This is close to sunrise; the low turbulence at sunrise is a well-known phenomenon in the astronomical community. Then turbulence increased again and stayed higher for the rest of the day. There are spikes roughly every two hours, and each corresponds to us coming into the cave to check all the battery conditions, thus passing by the tower twice each time. Also, we can see that at the very bottom two sensors’ area the turbulence level dropped down slowly overnight and came back up slowly during the daytime.

5.2.3 Temperature and relative humidity Data from ZEDs

We placed all the ZEDs alone at a same place pretty close to each other for about 38 minutes at the end of the experiment for calibration. The temperature data acquired during the calibration process are shown in Figure 5-5. The plot shows temperature data from 19:00 Jul 11 to 20:40 Jul 11. We turned off all the ZEDs from
their origin places at around 19:45, and then we placed the four ZEDs closely together on a rock and made all of them joined to the ZC again. As shown in the figure, dash lines stand for no data recording. After the four ZEDs were together, their temperature readings started to move towards the same temperature value (or a small temperature value range). If the readings finally agreed exactly with each other then it means there is no need to calibrate them, of course this is under the assumption that the temperatures of the four sensors are exactly the same. Let’s take a closer look at the calibration tail in Figure 5-6.

Figure 5-5: Temperature data acquired during the calibration process
Figure 5-6: A closer look at the temperature calibration tail

Assuming about 33 minutes after they were put together the temperatures of the four sensors were exactly the same which is at time 20:20. Then I choose the values at 20:20 to evaluate their offsets for calibration. As shown in Figure 5-6, at time 20:20, the sensor with ZED 1 reads 6.73°C, the sensor with ZED 2 reads 6.34°C, the sensor with ZED 3 reads 6.63°C, and the sensor with ZED 4 reads 6.49°C. Temperature readings fall in a 0.39 °C range. The average value is 6.55°C. Since I do not have an absolute temperature reference for them, I choose their average value to be the reference point. Thus the offset for each of them is actually an arbitrary offset. The arbitrary offsets for the four sensors are shown in Table 5-2. The RMS deviation from the mean of the sensors is 0.147 degrees. According to the spec sheet the sensors should be accurate to 0.1 degrees, so the differences we see are almost consistent with what we expect. The remaining small difference could easily be due to inaccuracies in our reference resistors,
and/or due to the discretization of the digitized values (the ADC returns the nearest digital value to the actual voltage value), and the digitization steps are similar in size to the uncertainty from the spec sheet. So this is actually a reasonable result. After 20:25, all four sensors sensed a increased temperature which is due to us sitting around the area near the sensors.

### Table 5-2: Readings from four temperature sensors at 20:20 and evaluated offsets

<table>
<thead>
<tr>
<th>Sensor with ZED</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reading at 20:20 (°C)</strong></td>
<td>6.73</td>
<td>6.34</td>
<td>6.63</td>
<td>6.49</td>
</tr>
<tr>
<td><strong>Offset (°C)</strong></td>
<td>-0.18</td>
<td>+0.21</td>
<td>-0.08</td>
<td>+0.06</td>
</tr>
</tbody>
</table>

Figure 5-7 shows the calibrated data for all four temperature sensors.

![Calibrated data plot](image)

**Figure 5-7: Calibrated data plot for all four temperature sensors**
Detailed locations for these four ZEDs were described at the end of Section 5.1. First of all we see temperature spikes that correspond to our entries to make them rejoin the network, and then temperature readings decayed back to previous temperature readings. Even a relative short entry changed the environment. Temperature reading from ZED 2 shows temperature deep in the cave is surprisingly cold: about 5.2 to 5.8 °C throughout the experiment. Temperature reading from ZED 4 shows that the temperature near the entrance is much warmer, with a range from 12 to 14 °C. The spikes in the black trace ('Entrance') during the daytime might be a reflection of winds with warm air coming in occasionally from outside the cave. All of the temperatures were lower the second day than they were overnight. Temperature near the entrance dropped about 1.5 °C. The other sensors indicate a roughly 0.4 °C lower temperature during the second day than overnight. The green and blue traces ('Deep' and 'Low') are very close to each other most of the time except that they crossed at around 13:00, with a relative stable temperature at the 'Low' location and a temperature decreasing at the 'Deep' location. It is a surprise to see the temperature at the 'Deep' location was actually higher than the temperature at 'Low' during the four hours from 9:00 to 13:00. The temperature drop seen from the blue and green trace a few minutes before 9:00 is because of my handling of the devices which applies to other quick temperature drops as well. Also, the red trace (temperature at location 'High') is 1 °C warmer than the green trace (temperature at location 'Low'). We do expect this to some extent, since hot air rises. Is there more turbulence when the gradient is larger? To answer this question, we probably need to have more data with more variations. Maybe we can find some correlations between temperature gradient changes and the magnitude of the thermocouple RMS.
Figure 5-8 shows a plot of data from all the relative humidity sensors.

![Graph showing relative humidity data from ZED 1, 2, 3, and 4 sensors over a 24-hour period from July 10 to July 11, 2008.]

**Figure 5-8: Data plot for four relative humidity sensors**

After verification, it is too bad to know that the relative humidity sensors with ZED 1, 2, and 4 were broken. Note that the green trace (sensor with ZED 3, the only one that was working) produced a very high relative humidity (close to 100%), which is probably consistent with our observation in the cave. The relative humidity rose up at night which may be consistent with the rain. It also topped out at about 103% which is understandable given that the accuracy of these relative humidity sensors is only 3.5%.

According to the data sheet of the relative humidity sensors, they are not supposed to be exposed to direct bright light otherwise they will be damaged. We speculate that the exposure to direct sunlight right before our experiment being the cause of their malfunctions. Painful lesson learned for the future.
CHAPTER 6 CONCLUSIONS AND FUTURE WORK

In this thesis project I successfully deployed a small-scale ZigBee Wireless Sensor Network in a cave environment which had never been done before. This was a proof-of-concept experiment which proved the idea of using a ZigBee WSN-based monitoring system for cave environments. The main contribution of this thesis work to the field of cave environment monitoring is that I proposed and tested a ZigBee WSN-based platform for a cave environment monitoring system as an alternative to a traditional data logger-based platform. This thesis work opened a window to a new study and development area for cave environment monitoring system infrastructure and also a new practical field of ZigBee WSN applications. However, a number of follow-on adjustments are needed both to sensors and to the ZigBee network in order to make it practical efficient, and reliable to deploy on a large scale. For now, the system does not have the capability to run for years. The estimated life span of the proof-of-concept system is less than a week.

The preliminary experiment was carried out in the Junction Cave lava tube at El Malpais National Monument in New Mexico. Data were collected for over 24 hours. Air turbulences at five different altitudes inside the cave were recorded throughout the experiment with a thermocouple tower. Temperatures from four different locations inside the cave were recorded for about 12 hours separated in several segments during the experiment. Only one relative humidity sensor recorded reasonable data, the other three
relative humidity sensors were damaged possibly because of exposure to direct sunlight right before the experiment. The ZigBee WSN worked well for 24 hours with the exception that the ZigBee End Devices all dropped off the network a few times. As a result, we lost a good portion of the temperature readings overnight.

The data from the thermocouple tower showed some interesting phenomena. This includes different air turbulence at different altitudes inside the cave; stronger air turbulence near the roof of the cave overnight which appeared to coincide with a period of rain. There was also a clearly noticeable increase in air turbulence when we entered the cave and were in the vicinity of the tower. The experiment demonstrated that the thermocouple-based sensors are quite useful for measuring air turbulence in a cave environment. Moreover, the thermocouple-based sensors showed us that they are very sensitive; they respond to our brief entries and passages close to the tower.

Both the temperature and thermocouple data shows that even very short entries into the cave modified the environment. In other words, at some level we disturbed the cave environment. This also means that in order to preserve a pristine and fragile cave environment and to collect precise cave environmental data we should minimize the time humans spend in the caves when a cave environment experiment is in progress. This is also another reason why we need the monitoring system to run as long as possible, the less human intervention the better.

In this experiment I did not implement any algorithms to trigger specific actions of the monitoring system based on real-time data. As mentioned in chapter 1, these actions may include: 1) trigger a higher sampling rate of the sensors to record more
detailed information about certain interesting events; or 2) trigger a camera to record
certain outside weather changes or human intrusions to the cave. However, given that the
ZigBee WSN-based monitoring system has the capability to respond to real-time data and
the capability to perform information exchanges among sensor nodes, it is possible to
implement these functions in future.

After the preliminary experiment in Junction Cave, there were many lessons to
learn. Hardware wise I experienced difficulties both working with the bulky
thermocouple tower and the sensor boards. Fast and easy connections between all the
hardware components are essential. As for the thermocouple tower, a lighter, more rigid
material is needed. A tower that is capable of standing on any terrain easily and stably
would be quite useful for experiments in a cave environment. The tower should also have
flexible height adjustments. A solution to protect the sensor boards and/or sensitive
fragile sensors against all conditions of the cave environment is also necessary. With
respect to the ZigBee WSN, more robust and reliable programming for wireless
communications is needed. For example, implementing the capability to detect network
connection failures and launch automatic reconnection attempts is needed.

Some suggestions for future work: 1) prepare customized PCB boards which hold
all the circuits for sensors and the ZigBee chips, within a small footprint with reliable
enclosures; 2) implement deep sleep on the end devices; 3) make the entire network go to
sleep for most of the time with all the radios in the network turned off in synchronized
time intervals to save power; 4) transmit multiple measurements at once when network is
awake, in one packet rather than transmitting in separate packets, to minimize
transmission overhead; 5) implement calculation of RMS values of sensor readings on
each node. For some measurements such as air turbulence data from thermocouples, sending only the RMS values instead of the whole data set might be sufficient, and that can help greatly reduce network traffic and improve network-wide performances, making it easier to deploy a larger scale network.

As for the ZigBee development hardware and tools, it is possible to try to get the ZigBee chips from other vendors than Freescale, such as Atmel where the software is free. Obtaining licenses for the Freescale development tools proved to be a hassle and the software was also very expensive. On the other hand, the current ZigBee chips from Freescale are extremely small which makes it possible to make very small sensors. But this also proved a disadvantage as the small chips are difficult to work with in a electronics laboratory environment. The ZigBee chip from Atmel can be obtained both as conventional surface mount packages and DIP packages which are easier to work with on a prototyping board.
APPENDIX A: THE C PROGRAM FOR THE SBC TO LOG

DATA

/*To compile, link and run this program on the TS-Linux,
 * Download package for the arm architecture gcc cross-compiler
 * [yang@stork ~]$ export PATH=/PATH:/usr/local/opt/crosstool/arm-linux/gcc-3.3.4-glibc-2.3.2/bin
 * [yang@stork ~]$ arm-linux-gcc -Wall -o armlogcom2 armlogger.c
 * [yang@stork ~]$ scp armlogcom2 yang@192.168.0.50:/home/yang/
 * [yang@stork ~]$ ssh yang@192.168.0.50
 * yang@ts7000:~$ ./armlogcom2
 * THIS IS THE FINAL VERSION
 * */

#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <fcntl.h>
#include <termios.h>
#include <termios.h>
#include <unistd.h>
#include <time.h>

int read_serial(int, int, const void *);
int DATA_SIZE = 14;/*Data package size*/
int kz;/*used to loop through buffer buf*/
long int sec;/*Time tag: second*/
int ms;/*Time tag: m second*/
int bytes;
struct timeval tv;
struct timezone tz;

int main()
{
    int fd = open("/dev/ttyAM1", O_RDWR); // | O_NOCTTY | O_NDELAY);

    time_t now;
time(&now);
    gettimeofday(&tv, &tz);
    sec = (long int)(tv.tv_usec);
    ms = (int)((tv.tv_usec)/1000);

    fp1 = fopen("N1.dat", "a");
    fp2 = fopen("N2.dat", "a");
    fp3 = fopen("N3.dat", "a");
    fp4 = fopen("N4.dat", "a");
    fp5 = fopen("N5.dat", "a");
    fp6 = fopen("N6.dat", "a");
    fp7 = fopen("N7.dat", "a");
    fp_backup = fopen("backup.dat", "a");
    fp_log = fopen("log.txt", "a");
fprintf(fp1,"\n\nData Section taken on: %s, (Seconds:%ld.%ld)d,ctime(&now),sec,ms);  
fprintf(fp2,"\n\nData Section taken on: %s, (Seconds:%ld.%ld)d,ctime(&now),sec,ms);  
fprintf(fp3,"\n\nData Section taken on: %s, (Seconds:%ld.%ld)d,ctime(&now),sec,ms);  
fprintf(fp4,"\n\nData Section taken on: %s, (Seconds:%ld.%ld)d,ctime(&now),sec,ms);  
fprintf(fp5,"\n\nData Section taken on: %s, (Seconds:%ld.%ld)d,ctime(&now),sec,ms);  
fprintf(fp6,"\n\nData Section taken on: %s, (Seconds:%ld.%ld)d,ctime(&now),sec,ms);  
fprintf(fp7,"\n\nData Section taken on: %s, (Seconds:%ld.%ld)d,ctime(&now),sec,ms);  
fprintf(fp_log,"\n"+"+\n",ctime(&now),sec,ms);

printf("fd is %d\n", fd);

unsigned char buf[DATA_SIZE];
int datalose = 0;

if (fd == -1){
    printf("Open serial port erro. ");
    fprintf(fp_log,"Open serial port erro. TIME:%s, (Seconds:%ld.%ld)d\n",ctime(&now),sec,ms);
} else{
    printf("Open serial port succeeded\n");
    fprintf(fp_log,"Open serial port succeeded. TIME:%s, (Seconds:%ld.%ld)d\n",ctime(&now),sec,ms);
}

struct termios options;
fcntl(fd, F_SETFL, 0); /*read function normal (blocking) behavior*/
tgetset(fd, &options); /* Get the current options for the port */
cfsetspeed(&options, B115200); /*set the input baud rate to be 115200*/
cfsetspeed(&options, B115200); /*set the output baud rate to be 115200*/
options.c_iflag &= ~PARENB; /*No parity check*/
options.c_iflag &= ~CSIZE; /* Mask the character size bits */
options.c_iflag &= ~(ICANON | ECHO | ECHOW | ISIG); /*raw data input*/
options.c_iflag &= NOFLSH;
options.c_iflag |= CS8; /* Select 8 data bits */
options.c_iflag |= (LOCAL | CREAD); /* Enable the receiver and set local mode */
options.c_iflag |= CRTSCTS; /*Disable hardware flow control*/
options.c_cc[VMIN] = 14; /*Force read function to read 14 bytes at a time*/
options.c_cc[VTIME] = 0; /*in tenths of seconds*/
tgetset(fd, TCSANOW, &options); /* Set the new options for the port */

fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
fclose(fp5);
fclose(fp6);
fclose(fp7);
fclose(fp_backup);
fclose(fp_log);

while (1) {
    fp1 = fopen("N1.dat", "a");
    fp2 = fopen("N2.dat", "a");
    fp3 = fopen("N3.dat", "a");
    fp4 = fopen("N4.dat", "a");
    fp5 = fopen("N5.dat", "a");
    fp6 = fopen("N6.dat", "a");
    fp7 = fopen("N7.dat", "a");
}

76
fp_backup = fopen("backup.dat", "a");
fp_log = fopen("log.txt", "a");
if(read_serial(fd, DATA_SIZE, buf)<0){return -1;}
time(&now);
gettimeofday(&tv, &tz);
sec = (long int)(tv.tv_sec);
ms = (int)((tv.tv_usec)/1000);

for (k=0;k<DATA_SIZE;k++){
    printf("The %d th element in buf is %x \n",k,buf[k]);
    printf("%x \n",buf[k]);
}

if (buf[0] == 66) { //A 0x42
   if (buf[1] == 66) { //A 0x42
      switch (buf[3]) {
      case 1: fprintf(fp1, "%n%d,%d,%ld,%d",
          buf[4]*256+buf[5],
          buf[6]*256+buf[7],sec,ms);//Node1, 0001 T,H
          break;
      case 2: fprintf(fp2, "%n%d,%d,%ld,%d",
          buf[4]*256+buf[5],
          buf[6]*256+buf[7],sec,ms);//Node2, 0002 T,H
          break;
      case 3: fprintf(fp3, "%n%d,%d,%ld,%d",
          buf[4]*256+buf[5],
          buf[6]*256+buf[7],sec,ms);//Node3, 0003 T,H
          break;
      case 4: fprintf(fp4, "%n%d,%d,%ld,%d",
          buf[4]*256+buf[5],
          buf[6]*256+buf[7],sec,ms);//Node4, 0004 T,H
          break;
      }
   }
   else{
      datalose++;
      fprintf(fp_log, "Package missed: +1 TIME:%s,
          (Seconds:%ld,%d)n",ctime(&now),sec,ms);
      read_serial(fd, 13, buf);
   }
}
else if (buf[0] == 65) { //A 0x41
   if (buf[1] == 65) { //A 0x41
      switch (buf[3]) {
      case 5: fprintf(fp5, "%n%d,%d,%d,%ld,%d,%d",
          buf[4]*256+buf[5],
          buf[6]*256+buf[7],
          buf[8]*256+buf[9],
          buf[10]*256+buf[11],
          buf[12]*256+buf[13],sec,ms);//Node5, 0005
          break;
      case 6: fprintf(fp6, "%n%d,%d,%d,%ld,%d,%d",
          buf[4]*256+buf[5],
          buf[6]*256+buf[7],
          buf[8]*256+buf[9],
          buf[10]*256+buf[11],
          buf[12]*256+buf[13],sec,ms);//Node6, 0006
          break;
      }
   }
   else if (buf[0] == 64) { //A 0x40
      if (buf[1] == 64) { //A 0x40
         switch (buf[3]) {
         case 7: fprintf(fp7, "%n%d,%d,%d,%ld,%d,%d"n",
             buf[4]*256+buf[5],
             buf[6]*256+buf[7],
             buf[8]*256+buf[9],
             buf[10]*256+buf[11],
             buf[12]*256+buf[13],sec,ms);//Node7, 0007
             break;
         }
      }
   }
}

Router1, 8 channels

break;

Router2, 8 channels

break;

Coordinator

break;
```c
}
else{
    datalose++;
    fprintf(fp_log, "Package missed: +1
(Seconds:%ld.%d\n",ctime(&now),sec,ms);
    read_serial(fd, 13, buf);
}

 else{
    datalose++;
    fprintf(fp_log, "Package missed: +1
(Seconds:%ld.%d\n",ctime(&now),sec,ms);
    read_serial(fd, 1, buf);
}

 printf("Package missed: %d in TOTAL\n", datalose);
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
fclose(fp5);
fclose(fp6);
fclose(fp7);
fclose(fp_backup);
fclose(fp_log);

} // end while (1)...
if (close(fd) == -1) {printf("error when closing the serial port");} 
else {printf("Serial port closed \n");
printf("End of data recording");
exit(0);
}
// end int main(){...

/*Function read_serial
 *
 * Used to read serial port
 *
 */
int read_serial(int fd, int n, const void *buf){
    int m, j = 0;
    if (fd < 1){
        printf(" port is not open\n");
        return -1;
    }
    while (j < n){
        ioctl(fd, FIONREAD, &bytes);
        m=read(fd, &((unsigned char *)buf)[j,n-j]);
        if (m>0){
            printf("m = %d\n",m);
            j+=m;
            if(j==n){
                break;
            }
        }
    }
    return 0;
}
```
APPENDIX B: A LETTER FROM DR. RICHARD SONNENFELD

Dear John and Anders,

I enjoyed the defense today. Anders did a good job finding interesting work for his student.

Just for fun I started thinking about what it would take to make measurements in a very deep very static cave.

Assuming a space which is 30 meters cubed, that's roughly 27,000 cubic meters at 1.3 kg/m^3 and 1000 J/kg K for specific heat of air at constant pressure. That's 35 million Joules to heat the space by 1 Kelvin, or 350,000 Joules to heat it by 0.01 Kelvin, and that applies to a well mixed space. If you don't even want to perturb the bottom 10% of the space by 0.01K, you are down to 35,000 Joules -- and if you want to emit less than that energy in a day, that corresponds to 35000/86400 = 4 tenths of a Watt average power consumption. Even at that there is a real risk of generating a thermal plume with your instrument. To run a microprocessor at all it would have to probably have a pretty low duty cycle -- particularly if you want to run several microprocessors. This makes the whole problem good fun because it's not impossible -- it would just take some real effort and imagination -- not to mention that in this case the effect of a 60 Watt human body is overwhelming.
REFERENCES


[17] ZigBee Alliance, available online at http://www.zigbee.org


