Development of a long range wireless sensor platform

Daryoush Bayat

Edith Cowan University

DEVELOPMENT OF A LONG RANGE WIRELESS SENSOR PLATFORM

By

DARYOUSH BAYAT

Master of Engineering Science

Edith Cowan University

Joondalup, WA

2011

Submitted to the Faculty of the
Computing and Health Science of the
Edith Cowan University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF ENGINEERING SCIENCE

ABSTRACT

Wireless Sensor Networks have emerged as an exciting field in recent years. There have been numerous studies on how to improve and standardise different aspects of wireless sensor networks. This paper aims to develop a wireless sensor network suitable for environmental monitoring applications. More specifically this paper aims to address the limited communication range of the existing wireless sensor technology.

In order to achieve the desired objectives, we have initially developed a hardware platform and then integrated the hardware with a long range RF radio module to achieve the goals. The system is further enhanced with mesh networking capabilities to increase the communication range and overall reliability of the network. The developed wireless sensor network is composed of sensors, microcontroller, RF radio module, antenna and expansion connectors for additional sensors and peripheral devices.

The developed wireless sensor network has been rigorously tested under three different scenarios to ensure the correct operation of the mesh network, communication range and effect of environmental obstacles such as vegetation and trees.

The developed wireless sensor network has been proven to be a suitable platform for environmental monitoring applications and the modular design has made it very easy to optimise it for different applications.

STATEMENT OF ORIGINALITY

I certify that this thesis does not, to the best of my knowledge and belief:

I. incorporate without acknowledgment any material previously submitted for a degree or diploma in any institution of higher education;

II. contain any material previously published or written by another person except where due reference is made in the text of this thesis; or

III. contain any defamatory material.

ACKNOWLEDGMENT

I would firstly like to thank my supervisor Prof. Daryoush Habibi for his guidance, encouragement and good advice. This thesis is a much work better thanks to his supervision.

My thanks must also go to Dr. Iftikhar Ahmad my co-supervisor, who, in spite of having practically no spare time, still managed to find time to provide help and advice.

Finally, I would like to thank my parents, wife and sister for their support during my studies.

TABLE OF CONTENTS

LIST OF FIGURES

List of Acronyms

ADC	Analog to Digital Convertor
EEPROM	Electrically Erasable Programmable Read Only Memory
PCB	Printed Circuit Board
PDA	Personal Digital Assistant
PDIP	Dual in Line Packaging
RF	Radio Frequency
TCP	Transmission Control Protocol
TQFP	Thin Quad Flat Pack
USART	Universal Synchronous Asynchronous Receiver/Transmitter
WSN	Wireless Sensor Network

CHAPTER 1. INTRODUCTION

Wireless communication has become an integral part of our society and new technologies are increasingly evolving around it. Recent advancements in wireless communication have led to the implementation of wireless data communication in laptops, PDAs and many other devices. In early days of development, one of the major limitations of wireless technology was that the end devices had to be in direct radio communication with the controller or base station. As a consequence the wireless networks were confined to small geographical areas.

In order to address this limitation, routing algorithms were implemented so that messages could be transmitted to the base station through several relay nodes. This networking technology, also called ad-hoc network has resulted in expansion of the network and has led to the development of many interesting applications and has created some challenges that have to be tackled. An ad-hoc sensor network is shown in Figure1.

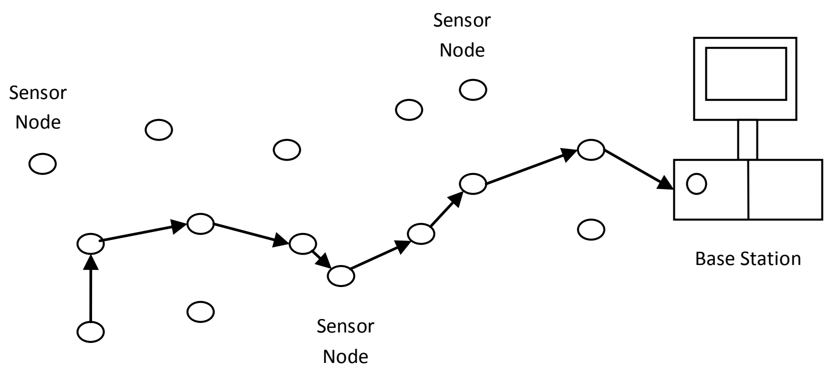


Figure 1. Ad-hoc Sensor Network

One of the major constrains of wireless communication that hasn't been adequately addressed to this date is power consumption. For indoor and other similar applications this is not much of an issue, because power is readily available. But for environmental monitoring applications power consumption is one of the main design issues. Wireless sensor network, an emerging technology has been able to address this issue through the use of smart networking technologies and advancements in microelectronics. Wireless Sensor Network (WSN) is a new generation of networking technology where a collection of smart nodes co-operatively monitor some parameters such as temperature, humidity, gas concentration, etc. A typical wireless sensor node has processing, communication and sensing capabilities. Wireless sensor networks have certain features that differentiate them from other networks. WSN has very stringent power consumption requirements and has very limited computational and storage capacities. A typical wireless sensor network should be able to function unattended for extended periods of time, usually months with say, two AA batteries.

The history of wireless sensor network dates back to the SmartDust project in 1998. This project proposed development of tiny devices in millimetres dimension that have sensing capability and are wirelessly connected. Although the SmarDust project ended early, it motivated many researchers to develop some interesting ideas in this field. The challenges involved in the design of WSN are how to minimize power consumption and increase the computational efficiency. Most of the power in WSN is consumed during data communication. It is therefore very important to implement efficient routing algorithms.

Another important factor in the development of wireless sensor networks is the design of the node itself. Most of the works [**1**], [**2**], [**3**] make optimistic assumptions about the environmental conditions. Many things can go wrong during the deployment of wireless sensor nodes. For example during the deployment of sensor nodes in inaccessible locations, they are usually scattered by an airplane. The problem is that we won't know the exact landing location of the sensor nodes due to changing wind direction and the nodes should be designed to stand the landing impact.

Some of the interesting applications of wireless sensor networks include habitat monitoring [**4**], environmental monitoring [**5**], and biomedical applications [**6**]. The design choice of a wireless sensor network is very application dependant. For example for environmental monitoring applications we ideally need a wireless sensor network that can cover large geographical areas whereas in biomedical applications, the wireless sensor network should be tiny and accurate. Nevertheless, the design of the wireless sensor networks should be modular to some extent in order for them to be economically viable. We certainly don’t want to develop a wireless sensor network solution from scratch for every single application.

The disadvantage of the currently developed sensor nodes is that they can only cover small geographical areas due to their limited transmission range. For applications such as environmental monitoring, the coverage area needs to be at least a few kilometres.

One example of environmental monitoring application of WSN is bushfire monitoring. A bushfire monitoring system should cover tens of square kilometres in order to be effective. Even though the short transmission range of existing wireless sensor networks can be compensated by incorporating a mesh algorithm, it would be economically infeasible to have thousands of densely deployed sensor nodes to monitor bushfire. In fact one of the quoted advantages of WSN is its low deployment cost.

1.1 ORGANISATION OF THIS THESIS

The aim of this thesis is to develop a wireless sensor network with long transmission range to be used for environmental monitoring applications. In order to achieve this we will investigate the following:

1. The development of the node hardware.
2. The development of the node software.
3. The integration of a long range communication device into the node.

The hardware of the sensor node is composed of a microcontroller, sensors, communication device, antenna, battery and expansion connectors for additional peripheral devices. These components are integrated on a PCB.

The software controls how these components interact with each other and runs in the microcontroller. The limited memory and computational capacity of the sensor nodes should be taken into account when developing the software.

The communication device or RF radio module interfaces directly with the microcontroller. In other words, messages are directly sent to the radio module, and after that it is the job of the radio module to transmit, find routes, send and receive acknowledgments, etc.

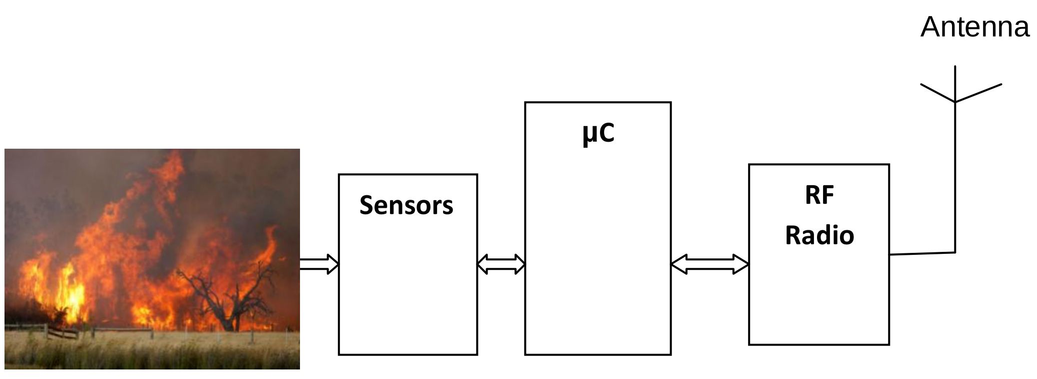


Figure 2. Environmental Monitoring Using WSN

As shown in Figure 2, in an environmental monitoring application, the sensors capture information from the environment in a measurable way such as voltage or current. The captured signal is sent to the microcontroller for processing and decision making. If there is any alarm, the microcontroller sends a warning message to the radio module where it is transmitted to the base station possibly through several other relay nodes. This is the principle behind the operation of WSN.

In CHAPTER 2, we conduct a literature review of the current state-of-the-art technologies in WSN. Topics will include the general architecture of a wireless sensor network, existing wireless sensor networks, possible networking protocols and the development platforms.

In CHAPTER 3, we will discuss the hardware of the developed wireless sensor node. Topics will include comparison of hardware components and why they have been chosen. We will also describe the functionality of ADC, UART, radio module and how they interact with each other.

CHAPTER 4 explains the software implementation of the developed wireless sensor node. Topics include component initialisation, description of registers and flow charts. The source code is given in APPENDIX A.

CHAPTER 5 explains the operation of the 9-Xtend RF radio module. Topics include the hardware of the radio module and the implementation of networking features.

In CHAPTER 6 we will show the results, field tests and the performance of the developed wireless sensor network under different settings.

CHAPTER 7 summarises the project and explains what needs to be done in future to enhance the performance of the developed wireless sensor network.

CHAPTER 2. LITERATURE REVIEW

INTRODUCTION

In this chapter we will have a brief overview of wireless sensor technology, its applications and the existing wireless sensor node technology. The main points of this literature review are as follows:

- Firstly we will explain the hardware architecture of a typical wireless sensor node and its components. This section is particularly important, because a major part of this thesis is the development of a new wireless sensor network hardware platform.
- Secondly we will look at some of the environmental monitoring applications of wireless sensor networks.
- Thirdly we will closely examine the existing popular wireless sensor networks and their shortfalls.
- Finally we will compare network stacks of OSI, Internet, and wireless sensor networks.

2.1 ARCHITECTURE OF A WIRELESS SENSOR NODE

A sensor node is a small battery powered device that is capable of processing sensory information and communication with other nodes in the network [7], [8]. The typical architecture of a sensor node is shown in Figure 3. In this section we will have a detailed look at components of wireless sensor nodes.

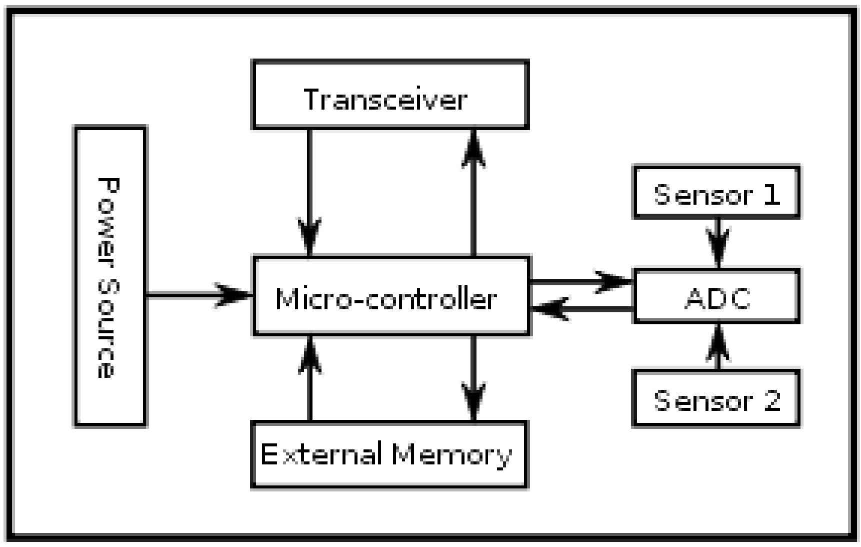


Figure 3. Typical Architecture of a Wireless Sensor Node

2.1.1 MICROCONTROLLER

The microcontroller executes instructions, processes data and controls the correct operation of other peripheral devices in the sensor node. There are also other alternatives that can be used as the controller such as desktop microprocessor, digital signal processors and FPGAs. A microcontroller is used in many embedded systems including sensor nodes due to low cost, easy programming, ease of interfacing with peripherals and low power consumption. General purpose microprocessors usually have high power consumption compared to microcontrollers; therefore they are not a suitable choice for sensor nodes. Digital signal processors offer better signal processing capabilities than microcontrollers, but due to easier signal processing and modulation required by sensor nodes, the superior processing of digital signal processors is not important.

2.1.2 TRANSCEIVER

Transceiver is where the data communication and networking takes place. Transceivers can operate in three modes; transmit mode, receive mode and sleep mode. Most of the power consumption in the sensor node is during data transmission. The possible choices of transmission medium are radio frequency (RF), optical communication (laser) and infrared. Lasers have low power consumption, but need line-of-sight for communication and are sensitive to atmospheric conditions. Infrared, like lasers, needs no antenna but it is limited in its broadcasting capacity. Radio frequency based communication is the most widely used communication medium in wireless sensor network applications. Wireless sensor networks tend to use license-free communication frequencies: 173, 433, 868, 915 MHz and 2.4 GHz.

Transceivers operating in the idle mode have power consumption similar to receive mode; therefore it is best to completely shut down the transceiver when not in use rather than leave it in idle mode. Most of the transceivers today have built in state machines that make transitions between operation states automatic.

2.1.3 SENSORS

Sensors are physical devices that produce electrical signals in response to physical changes in the surrounding environment. They measure environmental parameters such as temperature, humidity, gas concentration and light intensity. The analog signal produced by the sensors is fed into an analog digital converter (ADC) and then the digital values are sent to the controller. Inside the microcontroller, the digital values are calibrated by an already defined calibration equation to make them meaningful. Most of the microcontrollers have built in ADC unit; therefore there is no need for external ADC module. Sensors used in sensor nodes need to be small in size, autonomous and have extremely low power consumption.

2.1.4 EXTERNAL MEMORY

Most of the sensor nodes rely on on-chip memory inside the microcontroller due to energy concerns. The most popular external memories used are flash and EEPROM due to their low cost and storage capacity. Off-chip RAM is rarely if ever used. Memory requirement of a sensor node is entirely application dependant. Memories are used for two purposes: storing application related or personal data and program memory used for programming the sensor node.

2.1.5 POWER SOURCE

In a sensor node, power is consumed for sensing, data processing and communication. Most of the power is consumed in data communication. The most popular power sources used in sensor nodes are rechargeable batteries. Recently solar panels are being used with rechargeable batteries to eliminate the need for replacing batteries.

2.2 ENVIRONMENTAL APPLICATIONS OF WIRELESS SENSOR NETWORKS

Wireless sensor networks have a broad spectrum of applications. Due to their low cost and size and flexibility to expand, they can be utilised in a wide range of applications. Some of the applications include environmental monitoring, biomedical applications, and habitat monitoring and battlefield management [**9**], [**10**], [**11**], [**12**]. For example they can be used to detect and control the speared of forest fire. In this section we will focus on environmental monitoring applications.

2.2.1 VOLCANIC ACTIVITY MONITORING

One of the interesting environmental monitoring applications of wireless sensor networks is the detection of volcanic activity. In a detailed study at Volcan Tungurahua, an active volcano in central Educator [**7**], the possibility of utilising wireless sensor network for detection of volcanic activity has been investigated.

Wireless sensor networks have the potential to improve the quality of monitoring of volcanic eruptions. In the traditional method volcanologists use an array of seismometers and acoustic microphones to monitor the volcanic activity. The collection of these sensors cooperatively determines the source and location of the eruption and differentiates the true eruption from noise and other unwanted signals such as mining activity. The arrays of sensors used in traditional method are interconnected with cables and are distributed in a small area of less than 100 m^2. The data captured by sensors are stored in a hard drive or flash memory and has to be retrieved manually. Implementation of such system requires high power consumption which requires large solar panels, has small coverage area (less than $100m^2$) and is time consuming due to manual retrieval of data.

Implementation of a wireless sensor network can address all the above mentioned issues. A typical implementation is shown if Figure 4. The sensor nodes are MICAz motes equipped with three infrasonic microphones transmitting data to a base station node. The base station node relays data over a 9 kilometre wireless link to a remote laptop located at volcanic observatory. A separate GPS receiver is used to synchronise the operation of infrasonic sensors.

The advantages of using wireless sensor network in detection of volcanic activities include:

- Low power consumption which eliminates the need for large solar panels
- Larger areas can be studied compare to 100m2 in traditional method
- No need for manual retrieval of data
- Availability of real time data

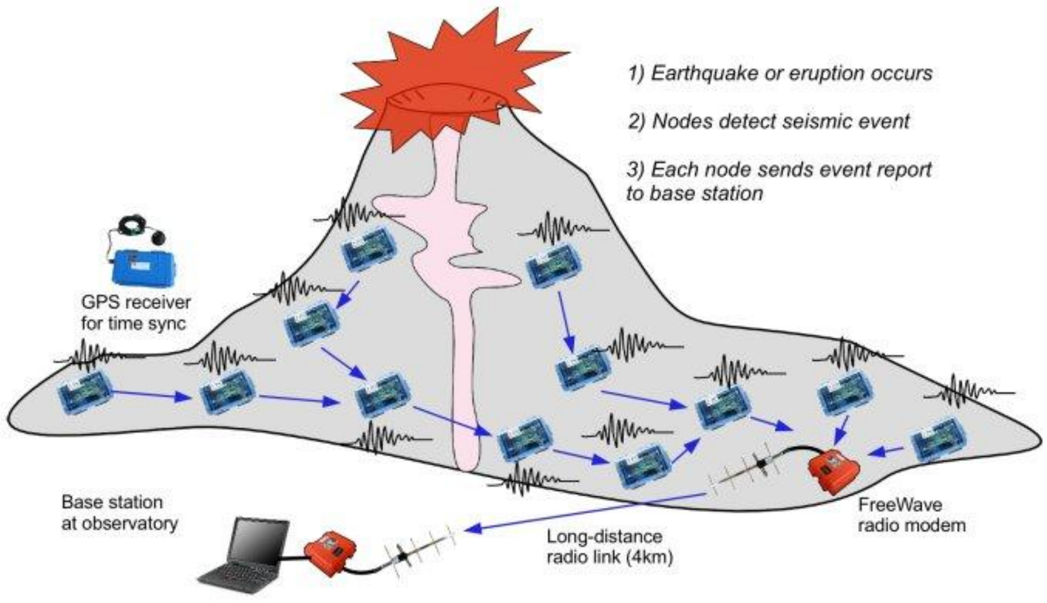

1) Earthquake or eruption occurs
2) Nodes detect seismic event
3) Each node sends event report to base station
GPS receiver for time sync
Base station at observatory
Long-distance radio link (4km)
FreeWave radio modem

Figure 4. Implementation of WSN in Detection of Volcanic Activity [7]

2.2.2 HABITAT MONITORING

In the field of environmental science, researchers rely on manual investigation of environment and collection of data. This method is very limited and very error prone. The deployment of wireless sensor networks for real-time data acquisition can significantly improve the quality of environmental investigation and help environmental scientists to study and cover larger areas for their targeted study. Researchers have already started implementing wireless sensor networks in real world environments, e.g. environmental monitoring [**8**] [**9**] [**10**] and habitat monitoring [**11**].

In a study conducted in Skomer Island **[**12**]**, a UK national reserve, the implementation of a wireless sensor network to monitor sea birds has been investigated. The system is composed of 20 battery powered sensor nodes. There is a dedicated sink node in the network. The sensor nodes are water-proof and placed next to the monitored burrows as shown in Figure 5. Sensors are placed at the entrance of the burrow and the following parameters are monitored:

- Temperature and humidity inside and outside of the burrows
- Movement at the entrance to the burrow
- Identity of the individual birds using RFID tags
- Weight of the individual birds using a small scale

The data collected by sensor nodes are processed and sent to the sink node. In the traditional method of habitat monitoring small data loggers with sensors are attached to the animals and then retrieved manually periodically. This limits the number of animals studied and is very time consuming and inefficient. Using a wireless sensor network provides more extensive and reliable information to environmental researchers.

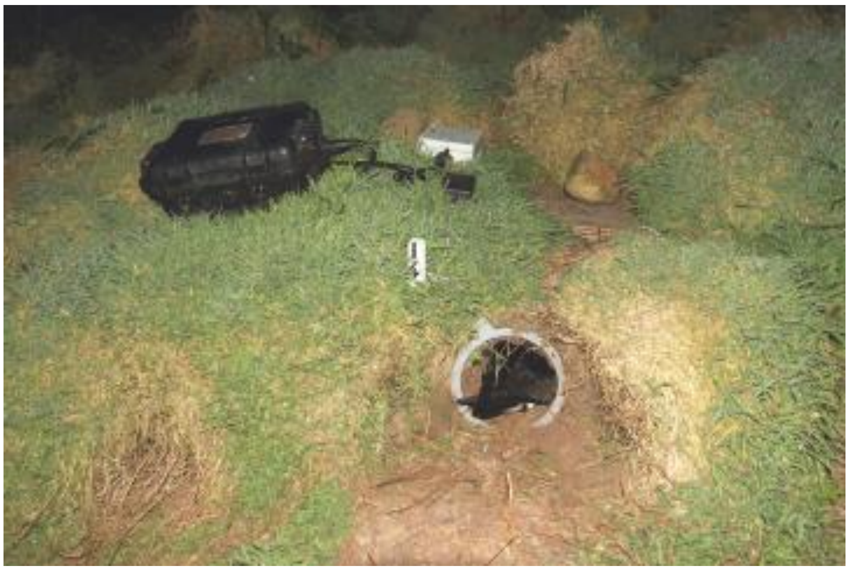


Figure 5. Sea Bird Habitat Monitoring Using WSN [12]

2.2.3 BUSHFIRE MONITORING

Another environmental monitoring application of wireless sensor networks is bushfire detection. In order to prevent bushfires from happening or minimise their damage, they have to be detected early. Today there are many different alarm systems to detect and report bushfire including satellite images [**13**] , infrared cameras and watchtowers [**14**]. The implementation of these alarm systems is very expensive and requires human interaction in case of fire watchtowers. Implementation of wireless sensor networks to detect bushfire provides robust and autonomous monitoring systems.

A typical bushfire detection system based on wireless sensor network is composed of collection of sensor nodes equipped with appropriate sensors scattered in the forest. A small bushfire detection system [**15**] developed by researchers at University of Adelaide achieved this using GSM technology. The architecture of this system is shown in Figure 6.

The system is composed of humidity and temperature sensors connected to the microcontroller. There is a GPS module interface attached to the microcontroller to locate the sensor node and a GSM module for communication of the captured data from sensors. The alarm messages are then collected by a mobile phone or SMS server for further action.

The humidity and temperature sensors regularly take readings and send the captured signals to the ADC module inside the microcontroller. If the readings exceed a predefined threshold, the microcontroller reads the location from GPS receiver and controls the GSM module. The collected temperature and humidity readings along with the position information are sent in SMS format to a mobile phone or SMS server.

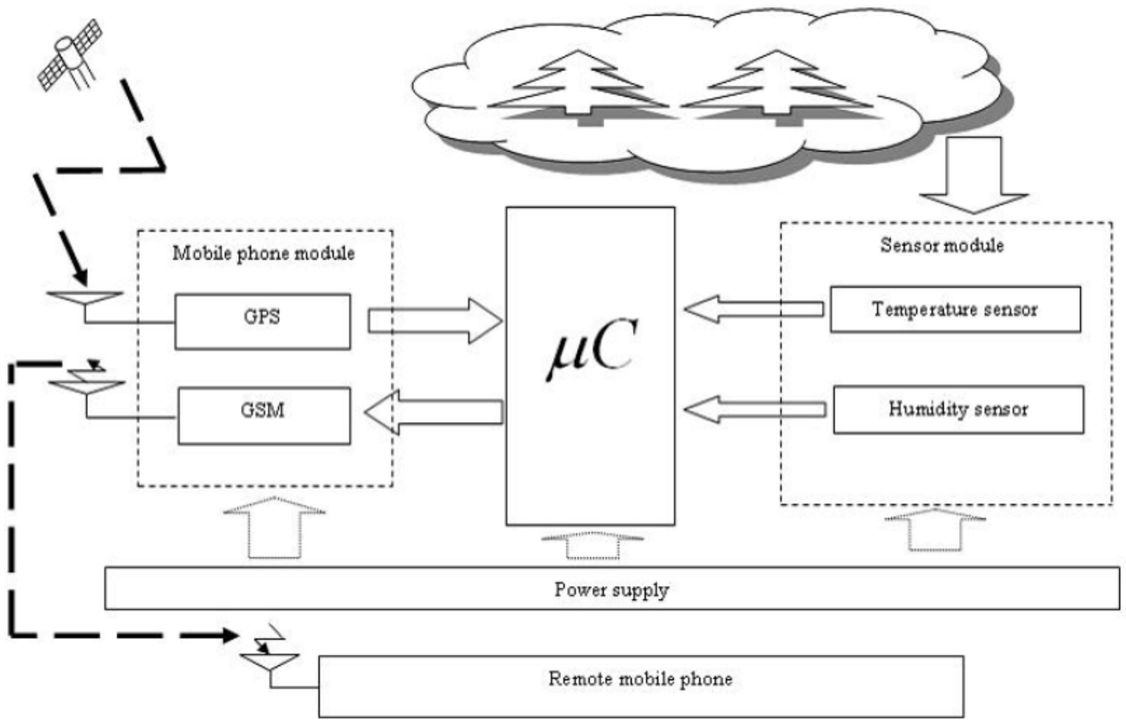


Figure 6. Architecture of a Bushfire Detection System [15]

2.3 EXISTING WIRELESS SENSOR NODES

Recently there have been many small companies that design wireless sensor nodes. Design of a wireless sensor node is very application dependant. For example, some applications require sophisticated signal processing capability while in other application such as environmental monitoring, transmission range is critically important. Most of the designed sensor nodes are targeted for low power, short range applications. There is still a big gap between real world monitoring applications and the existing wireless sensor technology. In this section we will have a detailed look at the existing wireless sensor nodes.

2.3.1 TELOSB

TelosB [**16**] is an ultra low power wireless sensor node designed for quick application prototyping. TelosB has built in light, temperature and humidity sensors and complies with IEEE 802.15.4 standard. The sensor node has USB interface for loading programs and communication with PC. TelosB is great platform for research purposes due to TinyOS support and interoperability with other IEEE 802.15.4 devices. The disadvantage is that it has short transmission range (less than 300m).

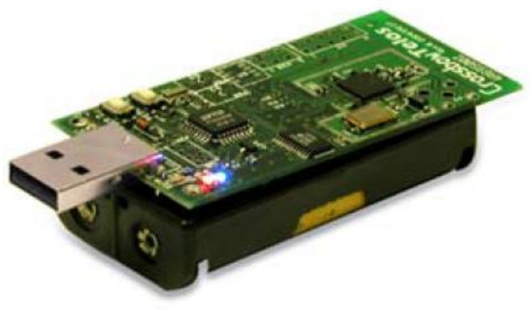


Figure 7. TelosB Wireless Sensor Node [16]

Some of the key features of TelosB include:

- TinyOS support : mesh networking and communication implementation
- Interoperability with other IEEE 802.15.4 devices
- Integrated light, temperature and humidly sensors
- Programming and data collection via USB

2.3.2 MICAZ

MICAz [**17**] is an ultra low power wireless sensor node designed for quick application prototyping. MICAz has expansion connectors for temperature, humidity, magnetic, pressure sensors and additional sensor boards and complies with IEEE 802.15.4 standard. The sensor node requires programming board for loading programs and communication with PC. MICAz is great platform for research purposes due to TinyOS support and interoperability with other IEEE 802.15.4 devices. The disadvantage is that it has short transmission range (less than 300 meters).

Figure 8. MICAz Wireless Sensor Node [17]

Some of the key features of MICAz include:

- TinyOS support : mesh networking and communication implementation
- Interoperability with other IEEE 802.15.4 devices
- Expansion connectors for additional sensors
- Plug and play

2.3.3 WASPMOTE

Waspmote [**18**] is an ultra low power wireless sensor node designed for quick application prototyping. One of the advantages of Waspmote is its modular design. For example, six different types of communication devices can be installed on the mote depending on the application. Furthermore, Waspmote has software and hardware interfaces for a GPS module, sensor boards etc. The communication range achieved depends on the communication devices, but the maximum range is a couple of kilometres. There is also a dedicated socket for solar panel input. This option is useful for environmental monitoring applications where solar energy is readily available. Some of the key features of Waspmote include:

- Long range compared to competing motes
- Modular design – easy prototyping
- Low power consumption
- User friendly hardware and software interface
- Solar panel interface
- IEEE 802.15.4 compatible

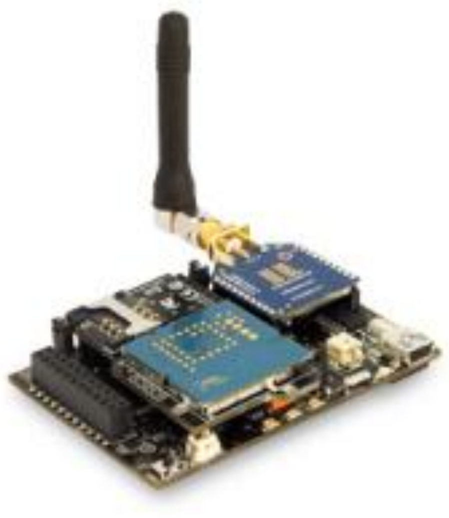


Figure 9. Waspmote Wireless Sensor Node [18]

2.4 NETWORK STACK

The operation of network devices is usually represented by a network stack. There are two network stacks; Open Source Interconnection (OSI) and Internet stacks which are compared in Figure 10.

Communication systems are divided into layers where each layer includes functions with similar logic. Each layer provides services to the upper layer and requests information and services from the layer below. The OSI model is composed of seven logical layers. The purpose of OSI model is to standardise networking technologies and make the development of networking technologies easier [**25**]. For example, someone using the application layer need not know about data link or physical layer and vice versa. Internet model aka TCP/IP is a simplified version of OSI model consisting of four logical layers. In the following sections we will discuss the functionality of different layers in detail.

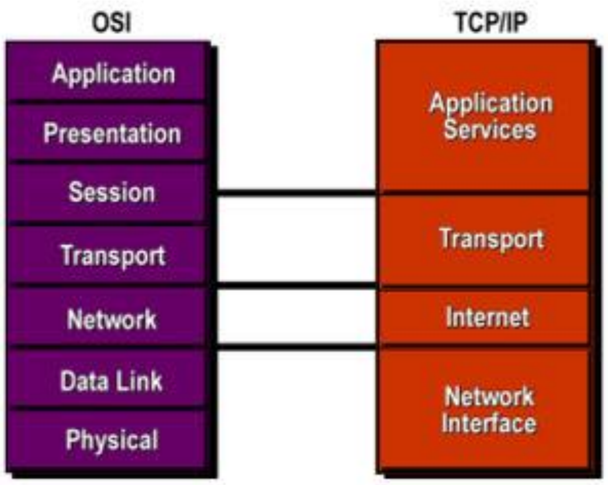


Figure 10. Comparison of OSI and Internet Models [19]

2.4.1 OSI STACK

There are seven layers in OSI model and the functionality of each layer is described below:

- The physical layer specifies electrical and physical specifications for communication systems. It is mainly consists of hardware of the transceivers and the transmission medium. Things like pin voltages, timing of signals, cable specification, hubs, repeaters and anything to do with hardware falls in physical layers. It also defines transmission medium such as copper, optical fiber, RF and etc.
- Data link layer has two main roles; error correction and medium access control. The error correction detects and probably corrects errors that might have occurred in the physical layer. Medium access control establishes connection between devices and makes sure all devices are given equal opportunity to access the communication medium.
- Network layer mainly takes care of routing of messages between end devices. It makes sure that messages arrive at their destinations by utilising some sort of acknowledgment and retransmission mechanism. Network layer also manages data traffic in the network and finds alternative routes when links are congested or broken.
- Transport layer takes care of reliable and in order delivery of messages between communication devices and performs error correction as well.
- The session layer takes care of connection between application processes. It establishes, manages and terminates the connections between applications.
- The presentation layer takes care of delivery and formatting of data to the application layer for further processing or display.
- Application layer interfaces with the user software. Some of the roles of application layer include identifying communication partners, determining resource availability, and synchronising communication.

2.4.2 INTERNET STACK

The internet model aka TCP/IP provides a communication framework for internet. The network stack is somewhat the simplified version of OSI model with similar functionalities.

Network interface layer does the same job as physical and link layers. Internet layer is the same a network layer, transport and application layers have the same function as their OSI counter parts.

2.4.3 WIRELESS SENSOR NETWORK STACK

According to Akyildiz, [**19**] the wireless sensor network stack is very similar to OSI/Internet protocol stack as shown in Figure 11. In addition to the usual layers, sensor networks have three more planes: power management, mobility management plane and task management plane.

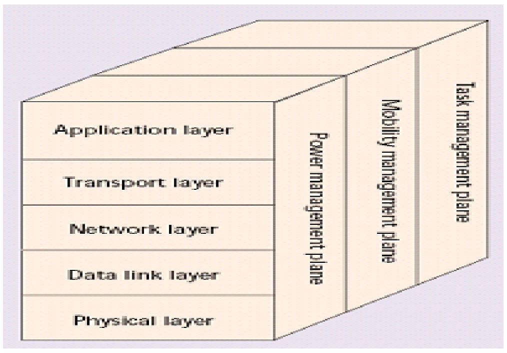


Figure 11. Wireless Sensor Network Stack [19]

The power management plane monitors power consumption of the wireless sensor network and takes actions to conserve power. Mobility management plane takes care of mobile nodes and makes sure there are available routes through neighbouring nodes. Task management plane controls scheduling and processing of information given to a region. Sometimes several nodes cooperatively monitor one given task. In this thesis we will be mainly be concerned with the application layer.

2.5 CONCLUSION

In this chapter we have covered the basics of wireless sensor network technology, specifically the hardware platform and its components. We briefly outlined some of the interesting environmental monitoring applications of wireless sensor networks. Several of the most popular wireless sensor platforms were evaluated and we concluded that a common shortfall of all the existing wireless sensor nodes is the limited transmission range. Finally we discussed network stack models including OSI, Internet and wireless sensor networks. We noted that wireless sensor networks have three additional planes in the network stack that take care of power management, task scheduling and mobility management.

CHAPTER 3. HARDWARE DESIGN

INTRODUCTION

This chapter will explain the development of the wireless sensor platform from the hardware point of view. We will explain how the components work and interact with each other.

In section 3.1 we will discuss the architecture of the developed wireless sensor platform and its components.

In section 3.2 we will talk about the microcontroller or central processing unit and its components such as UART and ADC.

In section 3.3 we will evaluate three different transceivers and explain why we choose 9-Xtend as our transceiver.

In section 3.4 we will evaluate several wireless networking protocols and their advantages and disadvantages.

Section 3.5 describes the working of the sensor module, its electrical connections, calibration, etc.

In section 3.6 we will explain the expansion connectors (SPI and I^2C) and how these protocols work.

Section 3.7 explains the operation of the external memory.

Finally, section 3.8 explains the PCB design of the complete system.

3.1 ARCHITECTURE OF ECUMOTE

The developed wireless sensor platform is composed of the following components: microcontroller, transceiver, temperature and humidly sensors, external EEPROM and expansion connectors for additional peripheral devices. The interconnection of these components is shown in Figure 12. In this chapter we will explain the operation of these components and how they are integrated on the board.

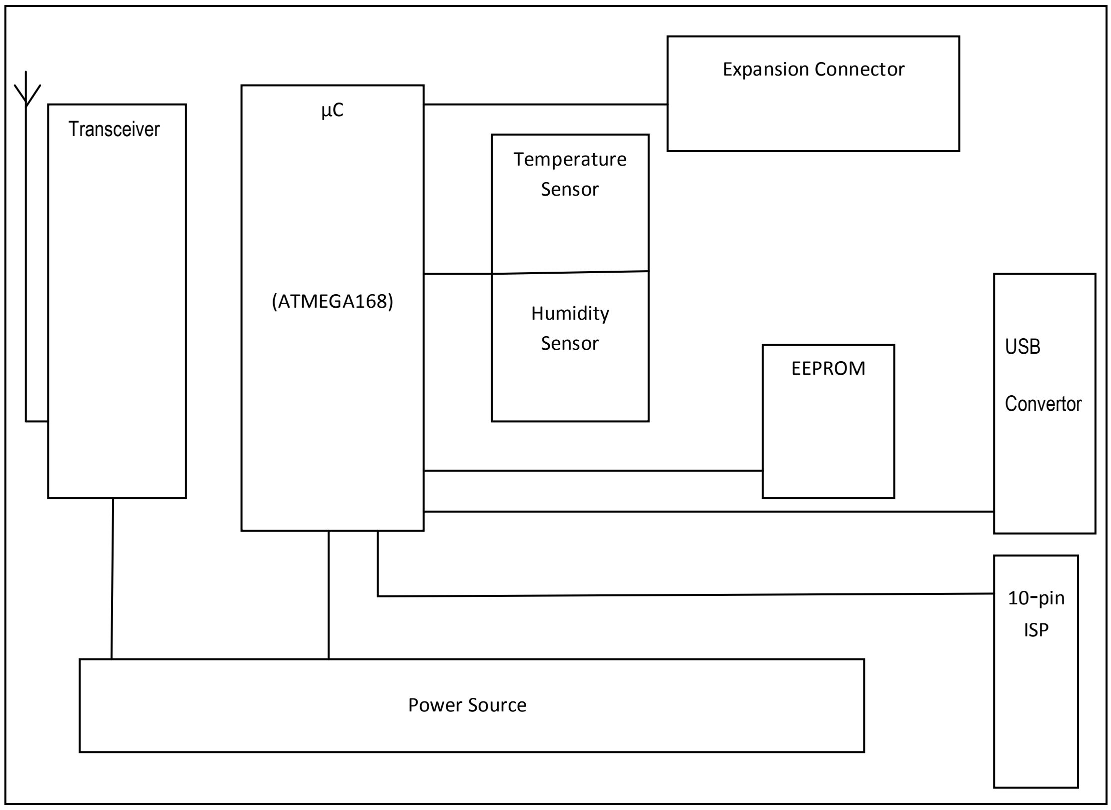


Figure 12. Architecture of the Developed Wireless Sensor Platform

3.2 CENTRAL PROCESSING UNIT

Microcontroller or central processing unit is the heart of the wireless sensor node. Although microcontroller's main purpose is processing of data, today's microcontrollers come with many additional features such as USART module, built in ADC, I^2C, SPI and memory. The integration of these features inside the microcontroller makes the development much more compact and easier. The chosen microcontroller for the wireless sensor node is ATmega 168 [**20**], due to the author's familiarity and experience with ATmega family of microcontrollers. In this section we will describe different features of the microcontroller and how they are used in our wireless sensor node. As shown in Figure 13, ATmega 168 is a 28 pin microcontroller with built in USART, 10-bit ADC, SPI and memory. The key features of this microcontroller that are related to our wireless sensor node include:

- Low power consumption
- Built in 10 bit ADC with 6 ADC channels
- Built in SPI, USART
- Support for C programming

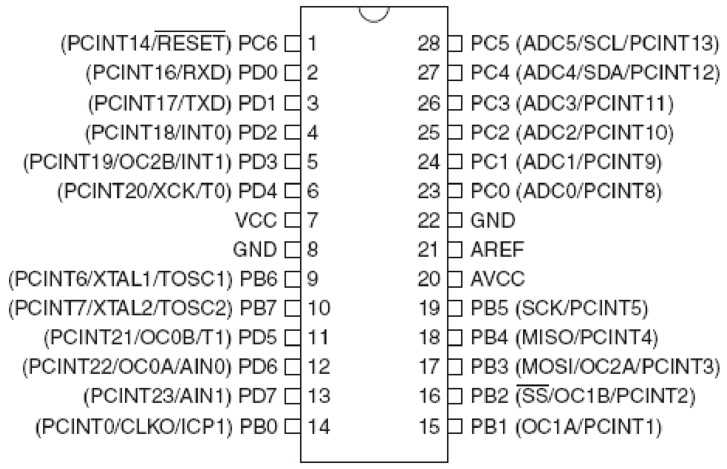


Figure 13. ATmega 168 Pin out Diagram

In the following subsections we will describe the hardware side of ADC and USART module and the I^2C and SPI interfaces are described in sections 3.6.1 and 3.6.2.

3.2.1 ANALOG TO DIGITAL CONVERTOR

ATmega 168 comes with 10 bit built in ADC. There are six ADC channels that are multiplexed to the ADC unit inside the microcontroller as shown in Figure 14.

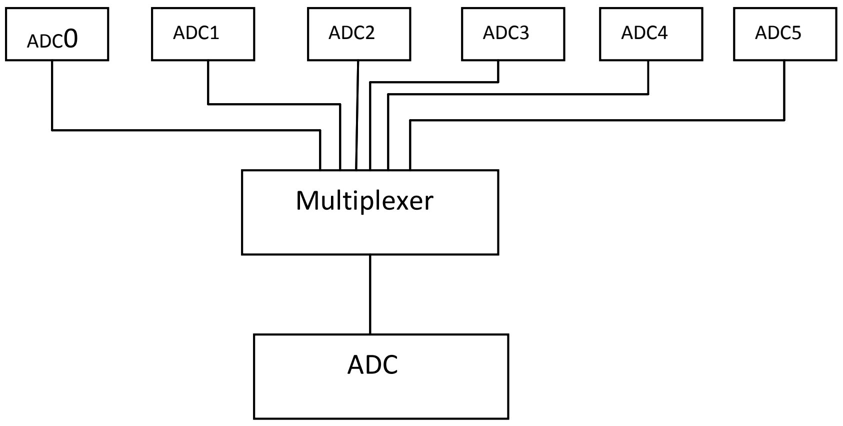


Figure 14. ADC Module

The ADC module needs a separate analog voltage supply (AVCC pin) and a clock signal to operate. The clock signal defines the sampling rate of ADC and the AVCC powers up the ADC circuitry. There is a prescaler inside the ADC module to bring down the CPU frequency to the desired frequency required by the ADC module (clock signal). The input analog signal to the ADC is converted to a 10 bit digital value, and the maximum input voltage is bounded by the reference voltage (AREF) which is set before using the ADC module. Therefore the minimum digital value produced by ADC is zero (ground) and the maximum value is 1023 (2^{10}-1) which corresponds to input voltage of AREF. This section just described the general operation of the ADC module from hardware point of view. Details of implementation such as register setting and initialisation will be described in CHAPTER 4.

3.2.2 UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSCEIVER

ATmega 168 comes with built in USART module. USART is a simple serial communication protocol that is used in our wireless sensor node for communication of data between the microcontroller and PC/Transceiver. As shown in Figure 15, microcontroller and PC/Transceiver are connected through TX (transmit) and RX (receive) pins. The USART module inside the microcontroller takes care of formatting and synchronisation of data. In order to communicate two devices using USART both of them have to have the same frame format and baud rate. Frame format includes the number of bits, stop bits and parity bits. We will be using 8-N-1 format (8 data bits, no parity and one stop bit). The baud rate is the rate of data communication between two devices in symbols per second. It is important for the baud rate to be less than the RF data rate of the transceiver in order to avoid packet loss and smooth communication between sensor nodes. In section 4.3 we will describe the software implementation of UART such as initialisation and baud rate calculation.

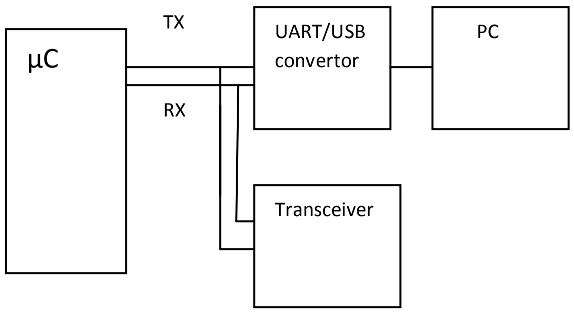


Figure 15. UART Module

3.3 TRANSCEIVER

Several communication devices were evaluated including Xbee-PRO, 9-Xtend RF radio module by Maxstream and RMX-232 by Embedded Communication Systems [21] [22] [23]. The evaluation process was based on transmission range, size, connectivity and power consumption. Transmission range was critically important due to the fact that the developed wireless sensor platform will be used for environmental monitoring applications that require long transmission range.

3.3.1 XBEE-PRO

The Xbee-PRO has the following specifications:

- **Transmission Range:** 9.6 km
- **Connectivity:** USART
- **Power consumption:** 954mW
- **Size:** 3.29mm x 2.44mm x 0.546mm
- **Cost:** $63 (AUD)

Xbee-PRO offers good transmission range and low power consumption at the same time. It is small and has UART connectivity which means that it can be directly connected to the microcontroller. Xbee-PRO also has advanced built in networking capabilities. The downside is that the transmission range is still not sufficient for environmental monitoring applications.

Figure 16. Xbee-PRO RF Radio Module [21]

3.3.2 RMX-232

The RMX-232 has the following specifications:

- **Transmission Range:** 2 km
- **Connectivity:** serial interface
- **Power consumption:** 630 mW
- **Size:** 60.5mm x 36.5mm x 5.1mm
- **Cost:** $230(AUD)

RMX-232 offers relatively good transmission range and low power consumption at the same time. The disadvantages are serial connectivity which means that we will need a level shifter to connect the radio module to the microcontroller which adds to the complexity and power consumption of the sensor node. RMX-232 is also very expensive compared to 9-Xtend and Xbee-PRO.

Figure 17. RMX-232 RF Radio Module [23]

3.3.3 9-XTEND

The 9-Xtend has the following specifications:

- **Transmission Range:** 64 km
- **Connectivity:** USART
- **Power consumption:** 3650mW

- **Size:** 3.65mm x 6.05mm x 0.51mm
- **Cost:** $150 (AUD)

9-Xtend was the chosen communication device due to its superior transmission range and built in advanced networking capability. It has USART connectivity which means that it can be directly connected to the microcontroller. The downside is its high power consumption. In environmental monitoring applications, solar power can be utilised to compensate for higher power consumption of the 9-Xtend radio module.

Figure 18. 9-Xtend RF Radio Module [22]

3.4 NETWORKING

In this section we will have a review of different wireless networking technologies that are suitable for implementation in wireless sensor networks. We will discuss the advantages and disadvantages of these networking protocols and particularly focus on long-range and power efficient protocols. Most of the wirelesses networking protocols are targeted for short-range applications and the ones that offer long transmission range are very complex and usually require monthly charges.

3.4.1 IEEE 802.11

IEEE 802.11 also known as Wireless Local Area Network (WLAN) or WI-FI is a group of wireless protocols used in computer communication. WLAN operate at 2.4 GHz and 5 MHz frequency bands. WLAN provides physical and data link layers [**24**] and can be formed into an ad-hoc network if routing algorithms are implemented in the upper layer. Table 1 summarises the four most important IEEE 802.11 standards in use.

Table 1. Comparison of Different Versions of IEEE 802.11 Protocols

802.11 protocol	Freq (GHz)	Data rate per stream(max) Mbit/s	Modulation	Indoor range (m)	Outdoor range (m)
a	5	54	OFDM	35	120
b	2.4	11	DSSS	38	140
g	2.4	54	OFDM	38	140
n	2.4/5	150	OFDM	70	250

The IEEE 802.11a standard uses Orthogonal Frequency Division Multiplexing (OFDM) in the physical layer. It operates at 5 GHz band with a maximum data rate of 54 Mbit/s, plus error correction code and maximum outdoor range of 120m.

Since 5GHz frequency band is relatively unused compared to the heavily used 2.4 GHz, IEEE 802.11a has better interference protection compared to IEEE 802.11b/g. The high frequency, however, results in shorter frequency which means that waves are absorbed more easily by walls and obstacles.

The IEEE 802.11b standard uses Direct Sequence Spread Spectrum (DSSS) in the physical layer. It operates at 2.4 GHz band with a maximum data rate of 11 Mbit/s, plus error correction code and maximum outdoor range of 140m.

IEEE 802.11b suffers interference from other devices operating at 2.4 GHz. Devices operating at 2.4 GHz frequency band include cordless telephone, Bluetooth devices and microwave ovens.

IEEE 802.11g standard uses OFDM in the physical layer. It operates at 2.4 GHz band with a maximum data rate of 54 Mbit/s, plus error correction code and maximum outdoor range of 140m. IEEE 802.11g hardware is fully compatible with IEEE 802.11b standard. IEEE 802.11g like IEEE 802.11b suffers interference from other devices operating at 2.4 GHz.

The IEEE 802.11n standard uses OFDM in the physical layer. It operates at 2.4 and 5 GHz frequency band with a maximum data rate of 150 Mbit/s, plus error correction code and maximum outdoor range of 250m.

IEEE 802.11n features multiple-input multiple-output antennas (MIMO) to improve communication range and throughput. MIMO offers significant increases in data throughput and range without additional bandwidth or transmission power. It achieves this by higher spectral efficiency and link reliability or diversity.

3.4.2 IEEE 802.15.4

IEEE 802.15.4 is communication standard that specifies physical layer and Medium Access Control (MAC) layer. IEEE 802.15.4 is targeted for low data rate and low power wireless sensor networks applications that require inexpensive transceivers. An example is Chipcon CC2420 [**25**] transceiver used in MICAz wireless sensor node which is produced by Crossbow Technology Inc.

IEEE 802.15.4 standard uses DSSS and orthogonal QPSK in the physical layers. It operates in 2.4 GHz, 915 MHz and 868 MHz with data rates of up to 250kbps. The output power of the transceiver is 1mW which results in a transmission range of 10-75m. The standard uses Carrier Sense Multiple Access (CSMA) in the MAC layer. IEEE 802.15.4 is suitable for applications that require low data rate and low power consumptions such as industrial control and indoor applications. Due to its short transmission range IEEE 802.15.4 is not a suitable choice for our wireless sensor node.

3.4.3 BLUETOOTH

Bluetooth is a global wireless standard aimed to connect laptops, mobile phones and commuters to form Personal Area Network (PAN) [**33**]. Bluetooth devices operate in 2.4-2.5GHz ISM band and can achieve a transmission range of 10-100m depending on power mode of the device. Bluetooth uses fast hopping CDMA (FH-CDMA) which provide protection to interference in heavily used ISM band. It can achieve data rate of up to 723.3 kbit/s

Bluetooth is a connection-oriented standard, meaning that it should establish communication channel before starting data communication. Due to its very short transmission range Bluetooth is not considered a suitable choice for our wireless sensor node.

3.4.4 ZIGBEE

ZigBee is a group of high level communication protocols that use IEEE 802.15.4 transceivers. According to ZigBee alliance [**26**] ZigBee is intended for use in mall embedded applications that require low power consumption.

ZigBee standard specifies three kinds of nodes: ZigBee Coordinator (ZC), ZigBee Router (ZR) and ZigBee End Device (ZED). ZC is the most advanced node sitting at top of the network connecting two networks. ZR simply acts as a relay between ZED and ZC to route packets. ZED is the simplest node with reduced functionality.

The differences in the functionality among nodes makes the set up of ZigBee networks difficult compared to homogenous networks. Due to the short transmission range of ZigBee standard, it is not considered a suitable choice for our wireless sensor node.

3.4.5 LONG RANGE NETWORKS

There are many long range wireless networking protocols, but most of them operate in licensed bands and require monthly charges. Additionally, most of these protocols are very complex and expensive to implement and they generally have high power consumption. Examples of long range wireless networking protocols are WIMAX which is defined by IEEE 802.16 [**27**] , third generation (3G) mobile telephone technology and DIGIMESH protocol which will be discussed in the following section.

In wireless sensor networks, most of the applications require low data rate. For example, in bushfire detection application most of the nodes will be in idle mode and transmit only a few short alarm messages during a forest fire. Therefore, the high bit rate provided by most of the long range wireless networks is not an advantage for environmental monitoring applications in wireless sensor networks.

In wireless senor network applications and more specifically environmental monitoring applications, the networking protocols should have the following properties:

- Low power consumption
- Low cost
- Small size transceivers
- Long transmission range
- Mesh or ad-hoc networking support
- Data rate is not an important factor
- No periodic charges after initial deployment

3.4.6 DIGIMESH

DIGIMESH is [**28**] a proprietary mesh network protocol and is the networking protocol used in our sensor node. In a mesh network, messages are routed through several nodes to reach the final destination. Other than the increased range, DIGIMESH offers a unique set of capabilities that makes it suitable for our sensor node.

Some of the important features of DIGIMESH include:

- Self healing
- Flexibility to expand network
- No need for expensive gateway routers
- Reliability

Self-healing means that during node failures, the network can find alternative paths to the destination. The flexibility to expand the network is due to the fact that any node can be added and removed from the network without affecting the functionality of the network as a whole. The homogenous nature of the network means that all the nodes have equal functionality; therefore there is no need for any gateway node with enhanced functionality. This makes the configuration of the network substantially easier. And finally, reliability is achieved through acknowledgments and retransmissions.

The routing algorithm in DIGIMESH is very similar to AODV (Ad-hoc On-demand Distance Vector) algorithm. There is an associative routing table for each node that maps the destination address to its next hop address. Therefore a message from a source node will go through routing nodes until it reaches the destination node. When the source node doesn't have a route for the specified destination it will initiate a route discovery process. During the route discovery process, the source node broadcasts a Route Request message. Upon reception of the Route Request message, the intermediate nodes rebroadcast the Route Request message if they don’t have a better route back to the source node, otherwise they drop the message. When the destination node eventually receives the Route Request message, it unicasts a Route Reply message back to the source node. The source node might receive multiple Route Reply messages; it will choose the one with the best round trip route quality. The destination address of the source nodes has to match with the source address of the destination node.

At the other end of the network, the destination node is connected to a PC, where contents of the packets can be viewed using the X-CTU software.

3.5 SENSOR MODULE

Sensors are an integral part of wireless sensor networks. They are physical devices that capture information from the environment, such as pressure, gas concentration temperature and humidity. The choice of sensor is entirely application dependant. Precision, power consumption and interfacing options are the three most important factors for choosing a sensor. In case of wireless sensor networks, we would ideally choose sensors with extremely low power consumption due to power constraints. There are two categories of sensors to choose from analog sensors and digital sensors. Analog sensors are components that measure actual values from environment and translate it into voltage or current that can be measured by electronic circuits. The voltage or current output of the sensor is fed into the ADC (analog digital convertor) module inside the microcontroller and then the digital values are processed by the user program for decision making. Digital sensors have built in ADC inside them and output digital values. Digital sensors have also built in logic circuits that enable the microcontroller to control the operation of the sensor. The most popular interfaces for digital sensors are I^2C and SPI and other proprietary two wire interfaces.

There are two analog sensors used in our long range wireless sensor nodes: humidity and temperature sensors. These sensors were chosen because temperature and humidity are two parameters that are used in many applications. There are also expansion connectors for additional I^2C, SPI and analog sensors.

3.5.1 HUMIDITY SENSOR

HIH-4030 [**29**] from Honeywell was chosen as the humidity sensor. HIH-4030 is an analog humidity sensor that can be connected directly to the microcontroller due to its near linear output. The special packaging provides protection to environmental hazards such as condensation, dirt, dust and other chemicals.

According to the datasheet, the output voltage and relative humidity are related according to the following equation:

$$V_{out} = V_{cc}\,(0.0062\ RH + 0.16) \quad (1)$$

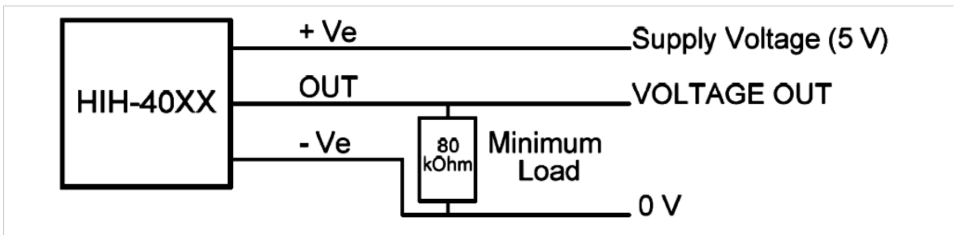


Figure 19. HIH-4030 External Interface [29]

where Vout stands for output voltage of the sensor, RH is relative humidity in percentage and Vcc is the supplied voltage. The output of the sensor is connected to one of the ADC channels of the microcontroller via an 80kΩ resistor as shown in Figure 19.

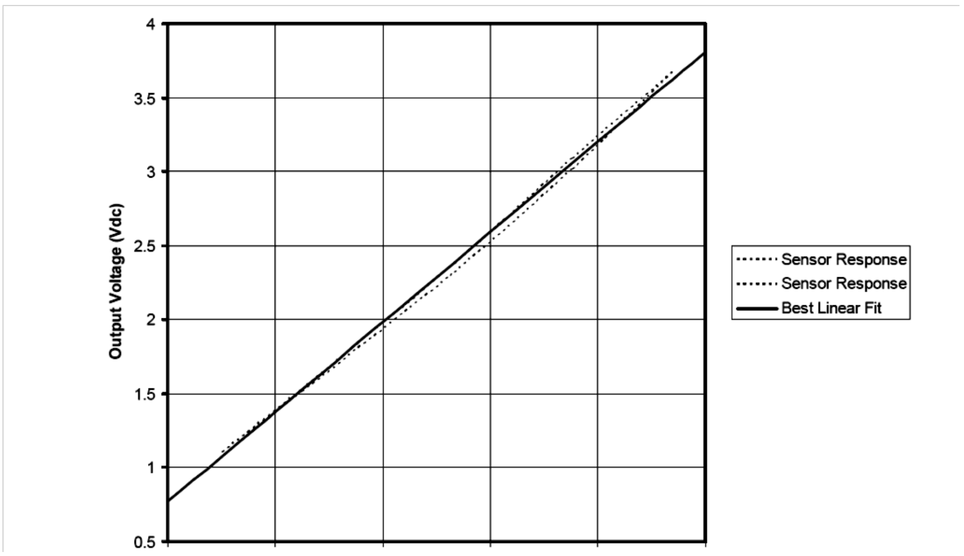


Figure 20. Output of HIH 4030 at Room Temperature and 5volt Supply Voltage [29]

The output of the sensor is radiometric to the supply voltage. The following graph shows the output of the sensor at 25 degrees Celsius and 5 volts supply voltage.

In order to compensate for temperature changes which affect the precision of the sensor the following equation is used:

$$\text{True RH} = \frac{\text{Sensor RH}}{1.0546 - 0.00216\text{T}} \quad (2)$$

where True RH stands for compensated relative humidity and T is the temperature in Celsius.

3.5.2 TEMPERATURE SENSOR

LM335 [**30**] from National instruments was chosen as the temperature sensor. LM335 is an analog temperature sensor that can be easily calibrated and operates over a -55 degrees Celsius to 150 degrees Celsius. It can be directly connected to the microcontroller with a 1 kΩ pull-up resistor as shown in Figure 20.

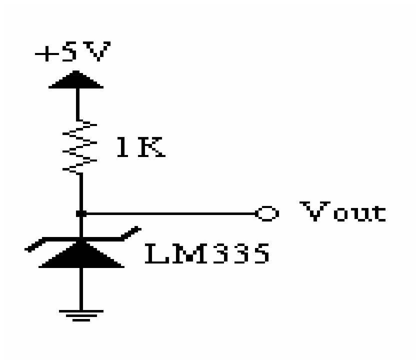


Figure 20. LM335 Circuit Diagram [30]

According to the datasheet the calibrated output of the sensor and temperature in Celsius are related according to the following equation:

$$T(C) = (T_0(K)\ (V_{outT}/V_{outT0}) - 273 \quad (3)$$

where T(C) stands for temperature in degrees Celsius, $T_0(K)$ is a reference temperature in Kelvin, V_{outT0} is the output voltage at the reference temperature and V_{outT} is the output voltage at any temperature. We calibrated the sensor at 25 degrees Celsius and the output voltage was 2.940 volts at that temperature.

3.6 EXPANSION CONNECTORS

3.6.1 I²C

Inter Integrated Circuit Bus (I^{2}C) is a low speed, two wire, half duplex, master/slave local area network protocol designed for interconnecting electrical subsystems on a PCB. An example would be the popular I^{2}C EEPROMs used in most PC memory modules for storing BIOS. Today a wide variety of devices, especially sensors, come with the I^{2}C interface. Furthermore, most small embedded microcontrollers come with the I^{2}C interface as well. Some of the devices that usually have the I^{2}C interfaces include:

- Analog digital convertors
- LCD drivers
- Environmental monitoring sensors
- Memory chips (EEPROM)

I^{2}C offers unique advantages which make it attractive in embedded systems. Simple two wire interfaces minimises interconnections so ICs have fewer pins and there are less PCB tracks which results in smaller and less expensive PCBs. ICs can be removed from the bus without affecting the functionality of other devices on the bus which simplifies configuration of the system easier. All devices connected to the bus are plug and play which means there is no need for external circuitry such as address decoders.

As shown in Figure 21, I^{2}C interface is composed of two devices; master and slave which are connected together using two wires; clock line (SCL) and data line (SDA). Master device (microcontroller) controls other devices on the bus telling them when to listen and when to talk. Slave devices have little control on the bus and are entirely controlled by the master device.

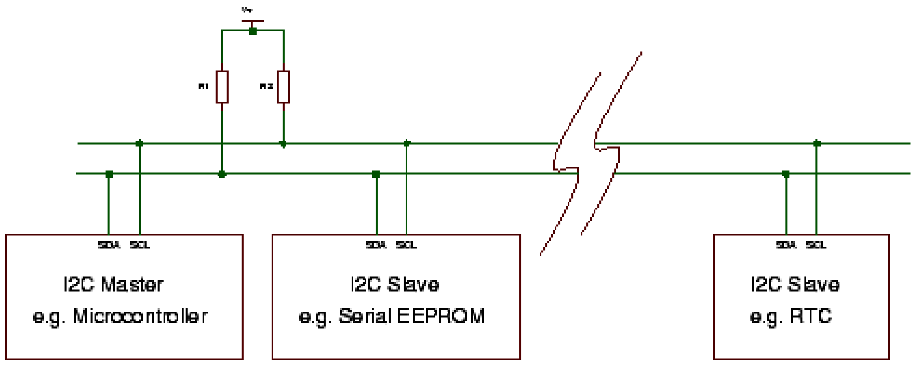


Figure 21. I^{2}C Interface

Master device which is usually the microcontroller transmits the address of the slave device being spoken to. Master generates the clock to synchronise the serial data and defines the direction (read and write) from/to the slave device. It initiates data transfer using stop and start conditions on the data line and at any time can be a transmitter or a receiver of data to/from the slave device. Slave device on the other hand synchronises itself to the clock generated by the master device. It responds only when addressed by master which means it cannot initiate data transfers. Slave device supplies data during read operation and stores data during write operation as defined by the direction of data transfer indicated by the master.

The addressing system in I^{2}C is very simple which eliminates the need for external address decoders. Each device on the bus is allocated a 7 or 10 bit address to uniquely identify it. A portion of the address is hardwired to the peripheral device by the manufacturer. Addresses are allocated by a central governing body based on technological groupings. For example I^{2}C standard defines Group A devices in the range [1010xxx] as memory type devices.

I^{2}C protocol has compact and simple hardware specification which means most of the complexity of the design is done in software. Software implementation of I^{2}C will be explained in CHAPTER 4.

3.6.2 SPI

Serial Peripheral Interface (SPI) is a serial data communication protocol that can operate as receiver and transmitter of data at the same time. Devices operate in master/slave mode and the master initiates the data transfer. The following are some of the devices that have SPI interface:

- Flash memory and EEPROM
- ADCs
- Digital sensors
- LCD displays

SPI protocol has unique advantages compared to I^{2}C protocol. Some of the advantages include:

- Higher throughput than I^{2}C
- Arbitrary choice of packet size
- Low power consumption

As shown in Figure 22, SPI protocol is composed of a master device (microcontroller) and one or more peripheral devices. The system bus is composed of four wires; SCLK (clock signal), MISO (master in slave out); MOSI (master out slave in) and SS (chip select signal). The fact that SPI can act as transmitter and receiver at the same time is due to two dedicated data lines MISO and MOSI. Dedication of two separate data lines results in higher data throughput but also results in an increase in the number of pins. Generally SPI protocol is suitable for applications that require fast data communication such as flash memory etc.

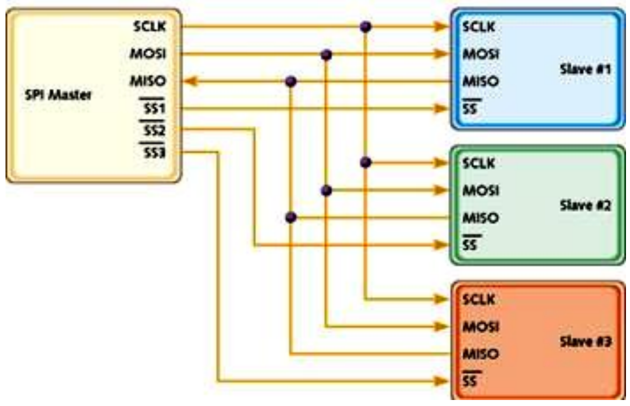


Figure 22. SPI Interface

Data communication on SPI is as follows:

1. Master device generates clock frequency which should be less than or equal to the maximum frequency of the slave device.
2. The master selects the slave device by pulling the chip-select signal low.
3. The master sends a bit via the MOSI line and the slave device reads it from the same line.
4. The slave sends a bit on via the MISO line and the master reads it from the same line.

The slave devices that have not been chosen by the master device should ignore the incoming clock signals. The master can only select slave devices one at a time.

There are two shift registers; one in the master device and one in the slave device that have the same size and are connected in a circular ring format. Data is usually shifted out of the register with the most significant bit first, while shifting in new data into the same register. After that the register has been shifted out, the master and slaves have exchanged register values. Then each device takes that value and does something with it, such as writing it to memory. If there are more data for communication, the same procedure is repeated.

3.7 EXTERNAL MEMORY

Our wireless sensor node comes with built in 8kbytes of external EEPROM memory. The use of this additional memory is entirely application dependent. In this section we describe the operation of the memory chip and how it is interfaced to the microcontroller and the software implementation will be discussed in CHAPTER 4.

As shown in Figure 23, the EEPROM is connected to the microcontroller via I^{2}C interface. In this configuration EEPROM acts as the slave device and the microcontroller as the master device.

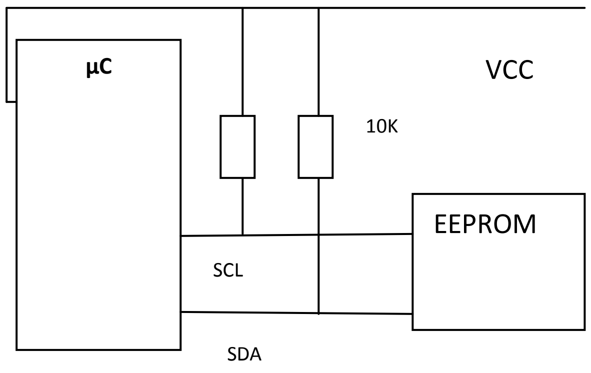


Figure 23. EEPROM Implementation

Before any data communication can take place the master (microcontroller) must put a start condition on the bus (SDA and SCL pins). The start condition is defined as the falling edge of the serial data line (SDA) while the Serial Clock (SCL) is stable at high state.

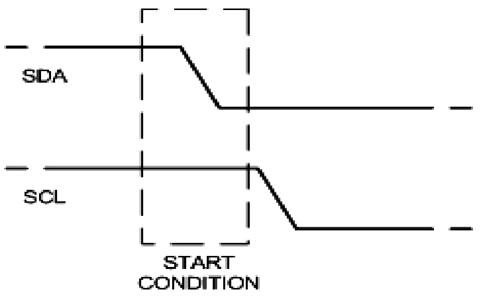


Figure 24. Start Condition

After the start condition has been put on the bus, the master sends the device select code on SDA line followed by R/W (read or write) signal in the 8th bit. The device select code is a 7 bit address which identifies the memory chip on the bus. The most significant 4 bits in the address identify the type of slave device which in our case is 1010 (memory devices) defined by the I^{2}C standard. The least significant 3 bits identify the exact device being addressed. This portion of address is hardwired to the slave device which in our case is 000. If there is a match, the EEPROM (slave device) acknowledges by pulling the SDA line low during the 9th bit time.

After the start condition has been put on the bus and the memory chip has been selected, depending on the R/W bit value the master writes data to the EEPROM if R/W is 0 and reads from memory when R/W is 1.

During write operation the byte address of the location in the EEPROM that will be written is transmitted by the master. Each byte in the EEPROM has a unique 16 bit address, so the most significant 8 bits are sent first followed by the least significant 8 bits. The memory chip sends ACK for each address bye received. Once the slave device has received the byte address, the master (microcontroller) sends the data byte and the slave device sends ACK after reception of the data byte. Immediately after receiving ACK, the master device puts stop condition on the bus which is defined by, as rising edge of SDA line while the SCL line is stable at high state. After the stop condition, the slave devices increments the address counter to point to the next byte address. The sequence of write operation is shown in the following figure.

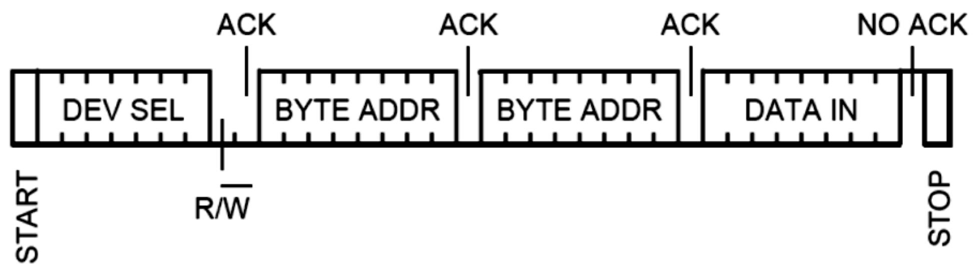


Figure 25. Write Cycle

Read operation is very similar to write operation except that the R/W bit is set to 1. There are three modes of read operation: Random Address Read, Current Address Read and Sequential Address Read.

During a Random Address Read, a dummy write operation is performed but without a stop condition to load the address counters of the EEPROM with the desired byte address. After the dummy write operation, the master puts start condition on the bus with the device select code and the R/W bit set to 1 (read operation). The slave device (EEPROM) acknowledges and puts out the data byte. The master device shouldn't respond to the ACK sent by EEPROM and terminate communication by putting a stop condition on the bus. The sequence of Random Address Read operation is shown in Figure 26.

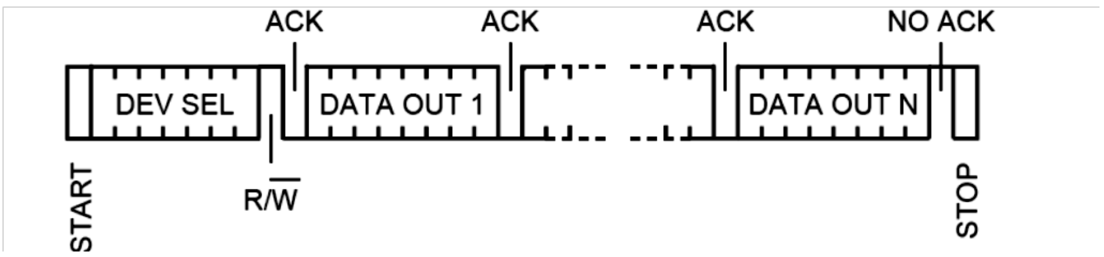


Figure 26. Random Address Read Cycle

During a Current Address Read operation, after the start condition, the master just puts the device select code and the R/W bit set to one. The slave device acknowledges and puts out the byte associated with the current address saved in its address counter and then the counter is incremented. The master does not acknowledge the data byte and terminates the data communication by putting a stop condition on bus. The sequence of Current Address Read operation is shown in Figure 27.

Sequential Address Read operation is very similar to Current Address Read operation except that the master does acknowledge the reception of data bytes and

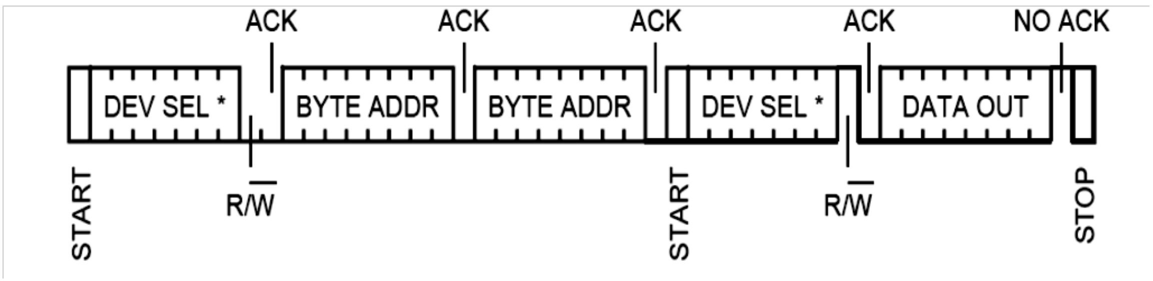


Figure 27. Current Address Read Cycle

continues reading data bytes sequentially.

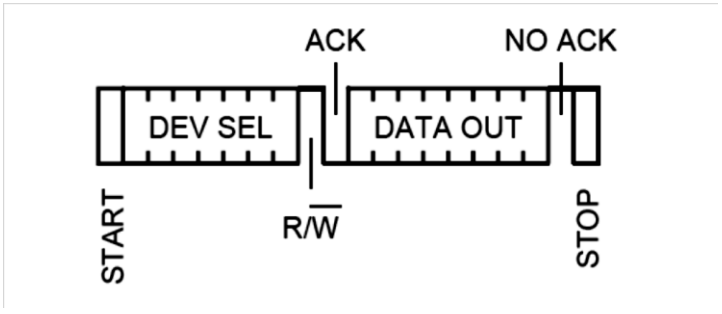


Figure 28. Sequential Read Cycle

3.8 PCB DESIGN

A PCB (Printed Circuit Board) for the complete design including microcontroller, transceiver, sensors and other peripherals was designed using ExpressPCB [**31**] and is shown in Figure 29. The PCB is double sided and the dimension is 80mm X 70mm. A 5 volt battery is housed underneath the PCB.

It should be noted that the board size can become smaller if we use a four layer PCB which is more expensive to manufacture. There is also a small circuit below the red LED which is the power supply circuit. This circuit is used to regulate the unregulated power supply. It is a low pass filter coupled with a 5V voltage regulator. In sensor nodes there are lots of power fluctuations from the battery or solar panel, and some of the devices on the board such as transceivers need a minimum voltage to operate properly. It is therefore essential for the supply voltage to be reliable to avoid system failures.

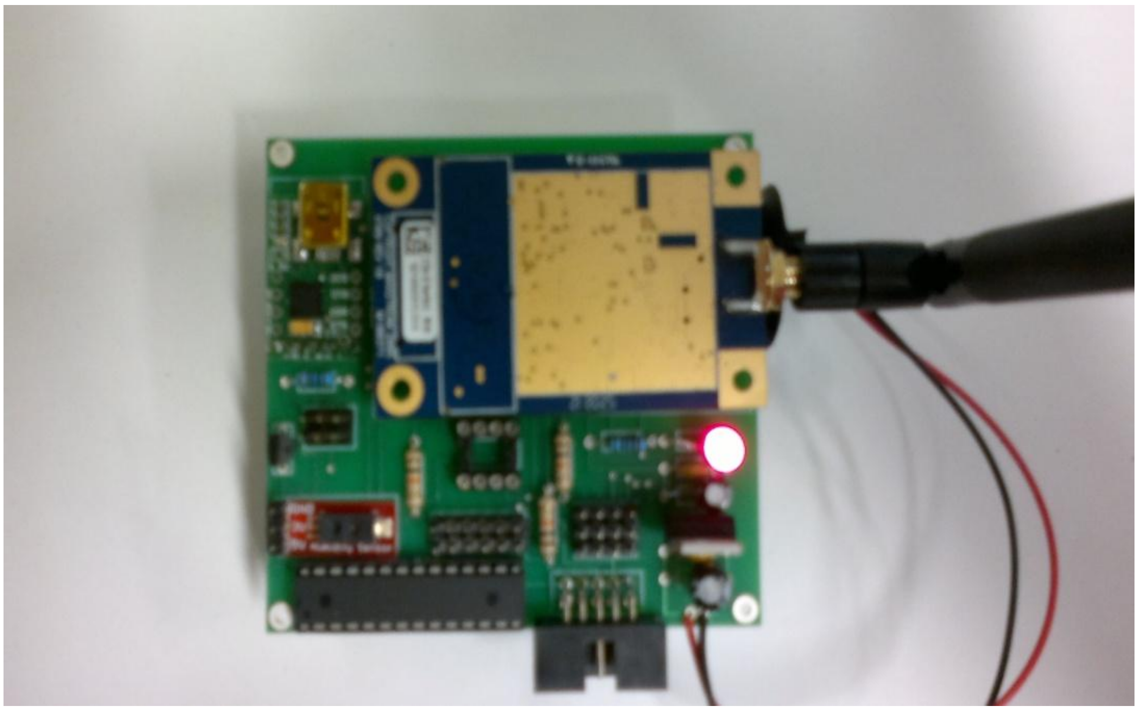


Figure 29. Complete PCB Design

CONCLUSION

In this chapter we have explained the development of wireless sensor platform from hardware point of view. We initially explained the architecture of the developed wireless sensor platform and its components and then we explained each component in more detail. We evaluated several RF transceivers and networking protocols. We explained the operation of SPI and I^2C expansion connectors in detail and the operation of the external memory which is connected to the microcontroller via I^2C interface. Finally we explained the design of the PCB for the complete system.

CHAPTER 4. SOFTWARE IMPLEMENTATION

INTRODUCTION

In CHAPTER 3, we explained the implementation of the developed wireless sensor platform from hardware point of view. Like any other embedded system, we will need to write a logic or software in order to control and interact with the hardware. In this chapter, we will explain the software implementation of the developed wireless sensor platform. We will only explain the algorithm and pseudo-code in this chapter. For source code please refer to APPENDIX A

In section 4.1 we will briefly talk about the software development platform, the procedure to write and compile the source code and how to load the logic or program into the microcontroller’s flash memory.

In section 4.2 we will explain the implementation of analog to digital convertor (ADC). Topics will include explanation of registers associated with configuring and controlling the ADC and then we will show how to initialise and use the ADC.

In section 4.3 we will explain the implementation of USART, including initialisation transmitting and receiving data over USART.

In section 4.4 we will explain how to configure and change parameters in our radio module using the X-CTU software.

In section 4.5 we will explain how to read from the sensors.

In section 4.6 we will explain the implementation of EEPROM. Topics will include the procedure to read and write from/to the EEPRIM chip using I^2C protocol.

4.1 DEVELOPMENT PLATFORM

AVR Studio [**32**] is a development platform for creating applications in 8-bit ATmega microcontrollers. It supports programming in both low level (assembly) and high level (C language) with the aid of the GCC [**33**] cross compiler. AVR Studio makes it easy to develop new applications due to its user friendly interface. In this section we will show how to develop a simple application on AVR Studio and then in the subsequent section we will explain the details of application that runs on the developed wireless sensor node.

In order to write applications for Atmel AVR microcontrollers in C language we will need a development platform and a C compiler: AVR Studio and WinAVR. AVR Studio includes an editor, the assembler, and HEX file downloader while WinAVR is a C compiler for AVR. It appears in AVR Studio as a plug-in.

1. Start the AVR studio program from the start menu
2. From the menu select **Project->New Projects**. In the dialogue box (see Figure 30), choose the location where you want to store the application and choose AVR GCC as the project type.
3. During the next step (see Figure 31), select AVR Simulator as the Debug platform and choose ATmega 168 as the microcontroller and then finish.

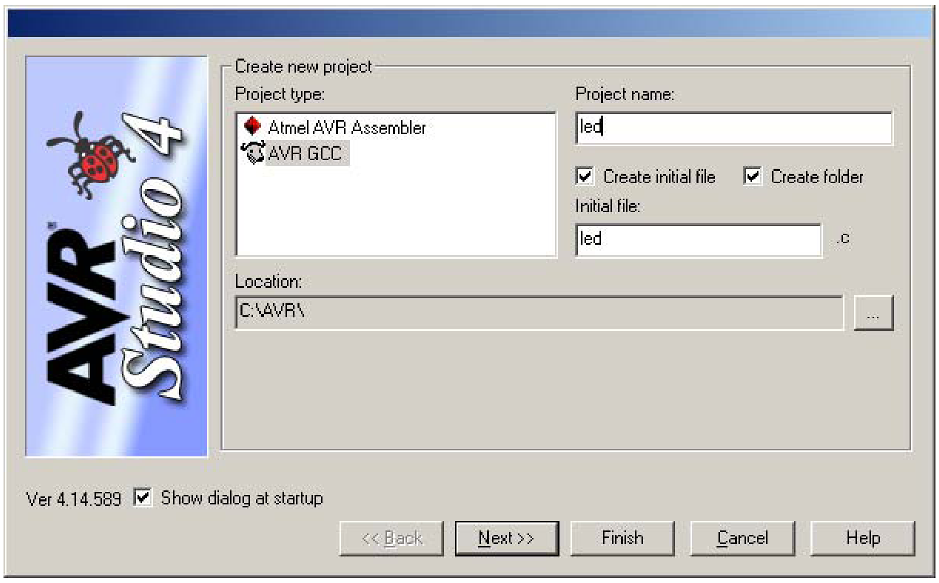

Create new project
Project type: Atmel AVR Assembler, AVR GCC
Project name: led
[✓] Create initial file [✓] Create folder
Initial file: led .c
Location: C:\AVR\
Ver 4.14.589 [✓] Show dialog at startup

Figure 30. Entering Project Type and Location [32]

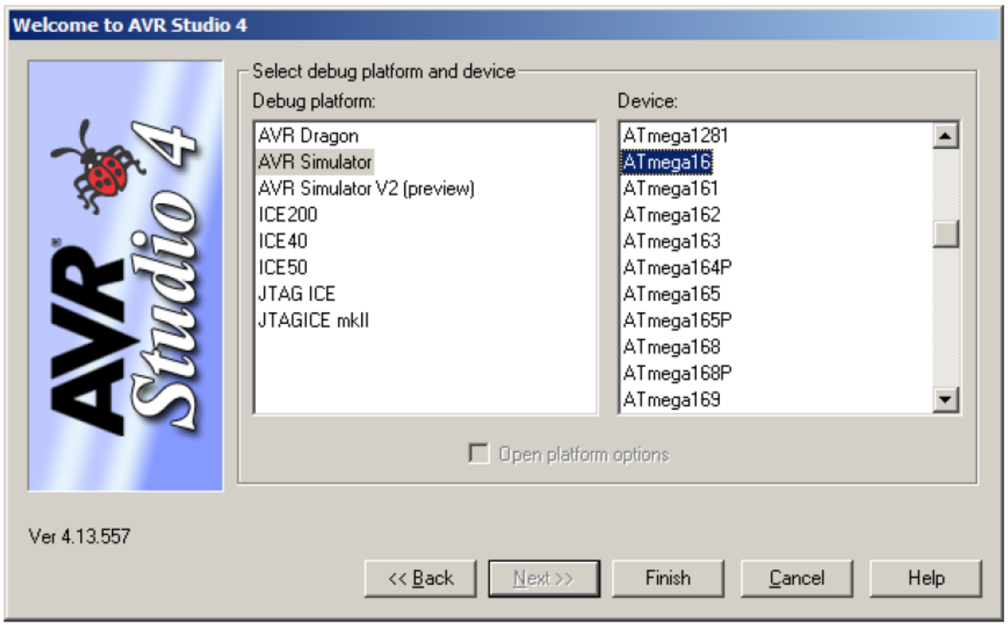


Figure 31. Selecting Debug Platform and Device [32]

After the initial configuration of the new project we can type in our c code in the editor. In order to build or compile the code, from the menu select **Build->Build all**. The machine code or generated HEX file will be in the default directory within the folder where the project is saved.

In order to burn the code to the microcontroller, we will need a programmer which loads the HEX file to the flash memory on the microcontroller. We used USBAsp [**34**] (see Figure 32), a programmer developed for Atmel AVR microcontrollers to upload the HEX file to the microcontroller.

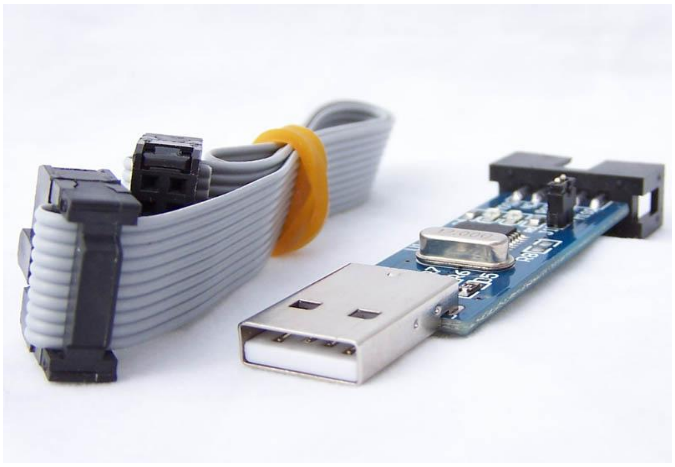


Figure 32. USBAsp Programmer [34]

4.2 ADC IMPLEMENTATION

As explained in section 2.1 ATmega168 has a built in 10-bit resolution ADC with 6 ADC channels multiplexed to it. The ADC channels are located in PORTA of ATmega168 which means 6 analog sensors can be connected to the device at the same time. The ADC can be operated in two modes: single conversion and free running mode. In single conversion mode the ADC does an instantaneous conversion and then stops while in free running mode it is continuously converting. In free running mode, it will immediately start a new conversion after the previous conversion. We will operate the ADC in single conversion mode, because the sensors have to only periodically monitor the environment.

4.2.1 ADC PRESCALER

The ADC unit needs a sampling frequency or clock signal to do its conversion from continuous waveform to discrete value. The input clock to the ADC is generated by the system clock divided into a lower frequency required by the ADC. The ADC requires a frequency between 50 KHz to 200 KHz. If the sampling frequency is high, conversion is fast but at lower sampling frequencies we get more accurate results. The Prescaler unit divides the system clock into the acceptable frequency required by the ADC**.** System clock can be divided by 2, 4,16,32,64,128 by setting the Prescaler.

4.2.2 ADC REGISTERS

Registers provide a communication link between the peripheral devices such as ADC, USART, I^{2}C and the CPU. These registers are used for configuring the ADC and storing the conversion results. There are four registers associated with the ADC unit.

1. ADC Multiplexer Selection Register – ADMUX: This register is used for selecting the channel and reference voltage.
2. ADC Control and Status Register A – ADCSRA: Holds the status of the ADC and is used for controlling it.

3. The ADC Data Register – ADCL and ADCH: The final result of conversion is stored here.

4.2.3 READING

Before using the ADC it has to be initialised. We have to configure the ADMUX and ADCSRA registers before reading the sensors. The ADMUX bits are shown in table 2. The REFS1 and REFS0 bits set the reference voltage and MUX4-MUX0 determine the ADC channel. The possible values of REFS1 and REFS0 and their meanings are shown in table 4.

Table 2. ADMUX Register

Bit No	7	6	5	4	3	2	1	0
Bit Name	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Bit Value	0	0	0	0	0	0	0	0

Table 3. ADCSRA Register

Bit No	7	6	5	4	3	2	1	0
Bit Name	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Bit Val	0	0	0	0	0	0	0	0

Table 4. Reference Voltage Selection

REFS1	REFS0	Reference Voltage
0	0	Vref turned off
0	1	AVCC
1	0	Reserved
1	1	Internal 2.56 volt

We will set the reference voltage to AVCC which is 5 volts. Now we will describe the bits in ADCSRA which control the operation of ADC.

- **ADEN** – Set this to 1 to enable ADC

- **ADSC** – We need to set this to one whenever we need ADC to do a conversion.
- **ADIF** – This is the interrupt bit which is set to 1 by the hardware when conversion is complete. So we can wait till conversion is complete by polling this bit.
- **ADPS2-ADPS0** – These bits select the Prescaler for ADC. We will need to set these bits in a way so that the ADC input clock falls in the 50 kHz-200 kHz range. Given that we have set the system frequency to 1 MHz, we will set these bits to 100 (16 in decimal) so that the input clock becomes 62.5 kHz which falls in the acceptable range. Table 5 shows the possible values of these bits.

Table 5. Division Factor Table

ADPS2	ADPS1	ADPS0	Division Fac
0	0	0	0
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

In order to read an analog value and convert it into a digital value we will take the following steps respectively. The C code is shown in APPENDIX A.

1. Initialise ADC: select operation mode, set the reference voltage, and choose the division factor for prescaler.
2. Select the input channel.
3. Complete the conversion and store it somewhere.
4. Manipulate the result (calibration etc).
5. Delay (application dependent).
6. Go back to step 2.

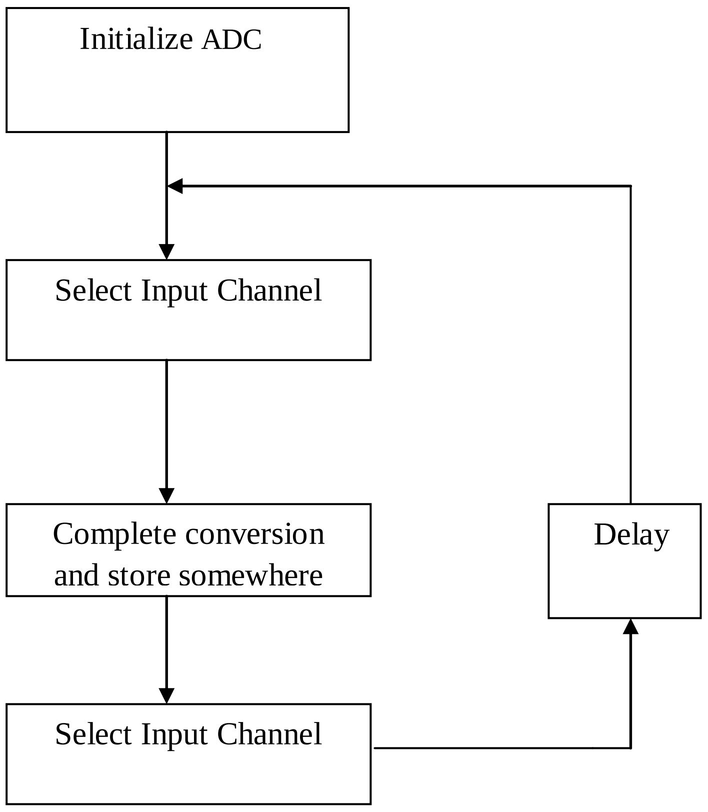


Figure 33. ADC Software Flowchart

4.3 UART IMPLEMENTATION

As mentioned in CHAPTER 2, ATmega 168 has built in USART hardware. The integration of the USART hardware in the microcontroller makes the programming and application development very easy. The user should only initialise the USART according to the desired data rate and frame format and just put the data in the register for transmission. In the following sections we will explain the USARR registers and how to initialise and communicate via USART in software.

4.3.1 UART REGISTERS

As with other peripheral devices USART has dedicated registers that provide the communication link between the USART and CPU (see Figure 34). There are six registers associated with USART which will be described below.

UDR - USART Data Register: as the name suggests, this registered is used for storing the user data. This register can behave in two different ways. When it is read, it will retrieve the data from the receive buffer, and when it is written it will send the data to the transmit buffer.

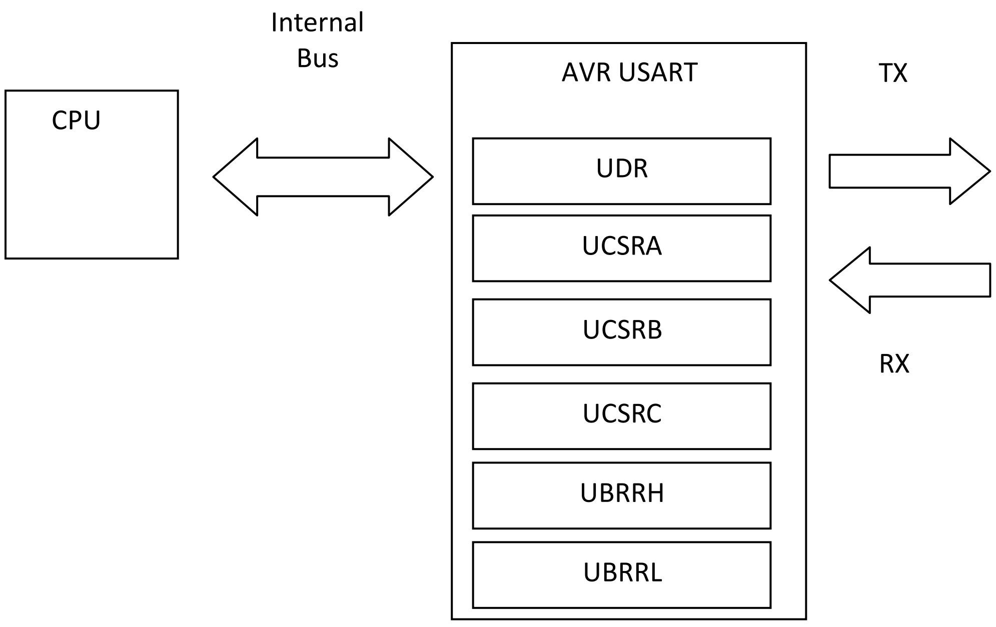


Figure 34. USART Registers

UCSRA- USART Control and Status Register A: This register is used to know whether we have received any data or if there is any data to be transmitted.

Table 6. UCSRA Register

Bit No	7	6	5	4	3	2	1	0
Bit Name	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
Bit Value	0	0	1	0	0	0	0	0

RXC: If this bit is set to 1, it means that the USART has received a complete byte and we can retrieve the received byte from UDR register.

TXC: This bit is set to 1 when USART has completed the transmission of one byte, and we can write new data to the UDR register.

UCSRB- USART Control and Status Register B: This register is used enable interrupts, receiver and transmitter and conFigure frame format.

Table 7. UCSRB Register

Bit No	7	6	5	4	3	2	1	0
Bit Name	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Bit Value	0	0	1	0	0	0	0	0

RXCIE: this bit should be set to 1 if the receive interrupt is to be enabled.

TXCIE: this bit should be set to 1 if the transmit interrupt is to be enabled.

RXEN: receiver enable

TXEN: transmitter enable

UCSZ2: defines frame length (discussed later)

UCSRC- USART Control and Status Register C: This register is used for formatting and configuration of USART.

Table 8. UCSRC Register

Bit No	7	6	5	4	3	2	1	0
Bit Name	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
Bit Value	0	0	1	0	0	0	0	0

URSEL: UCSRC and UBRRH share the same address. In order to distinguish, we set URSEL to 1 if we want to write to UCSRC register otherwise it is written to UBRRH.

UMSEL: if set to 1 means synchronous communication otherwise asynchronous.

USBS: determines the number of stop bits. If it is 1, means 2 stop bits otherwise 1 stop bit.

UCSZ2-UCSZ0: these bits define the length of the data frame. The following table shows the possible entries.

Table 9. Frame Format Selection

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5 bit
0	0	1	6 bit
0	1	0	7 bit
0	1	1	8 bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9 bit

4.3.2 INITIALISATION

The USART has to be initialised before any data communication can take place. The initialisation involves setting the baud rate, frame format (such as length, stop bits) and enabling transmitter and receiver. The first step is to calculate the vales for UBRRH and UBRRL registers. UBRR is calculated according to the following equation:

$$UBRR = F_{OSC} / (16\,(Baud - 1)) \quad (4)$$

We are using bit rate value of 2400 bit/s and the F_{OSC} is 1MHz. Therefore the UBRR come to be 26. This UBRR value gives 0.2% error rate according to the datasheet which is within the acceptable range for reliable communication. The calculated UBRR value should be written to the UBRR register (UBRRL and UBRRH). The following flowchart shows the initialisation steps.

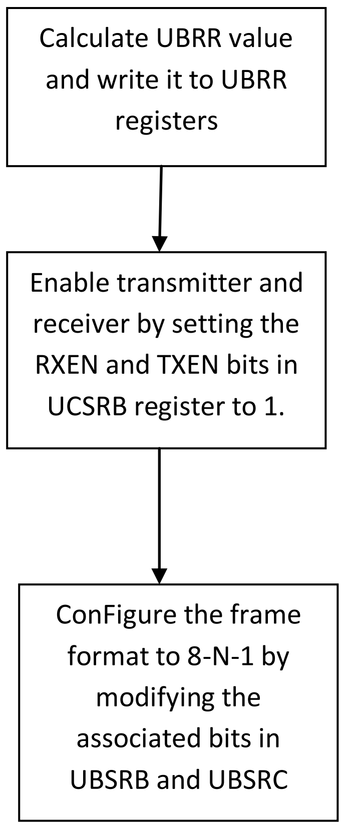

Calculate UBRR value and write it to UBRR registers
Enable transmitter and receiver by setting the RXEN and TXEN bits in UCSRB register to 1.
ConFigure the frame format to 8-N-1 by modifying the associated bits in UBSRB and UBSRC

Figure 35. UART Initialisation

4.3.3 TRANSMISSION AND RECEPTION

After the appropriate initialisation of the USART, data reception and transmission is very simple. For data transmission, we just put the data on the UDR register and it automatically gets transmitted. However before putting the data for transmission we have to wait for transmit buffer to be empty. When there is any received data, it will be put in the UDR register. This is the most basic form of communication over the USART. However in our application we might want to send and receive strings of data with varying types of variables rather than a single byte.

Rather than using these basic communication functions we will be using the printf function from AVR Libc. AVR Libc is a free software project which provides a high quality C library for use with GCC on Atmel AVR microcontrollers. The implementation of printf in AVR is a bit more complicated and different than the standard C printf function, because we have to tell the printf where we want to print the data which in our case is USART.

4.4 X-CTU SOFTWARE

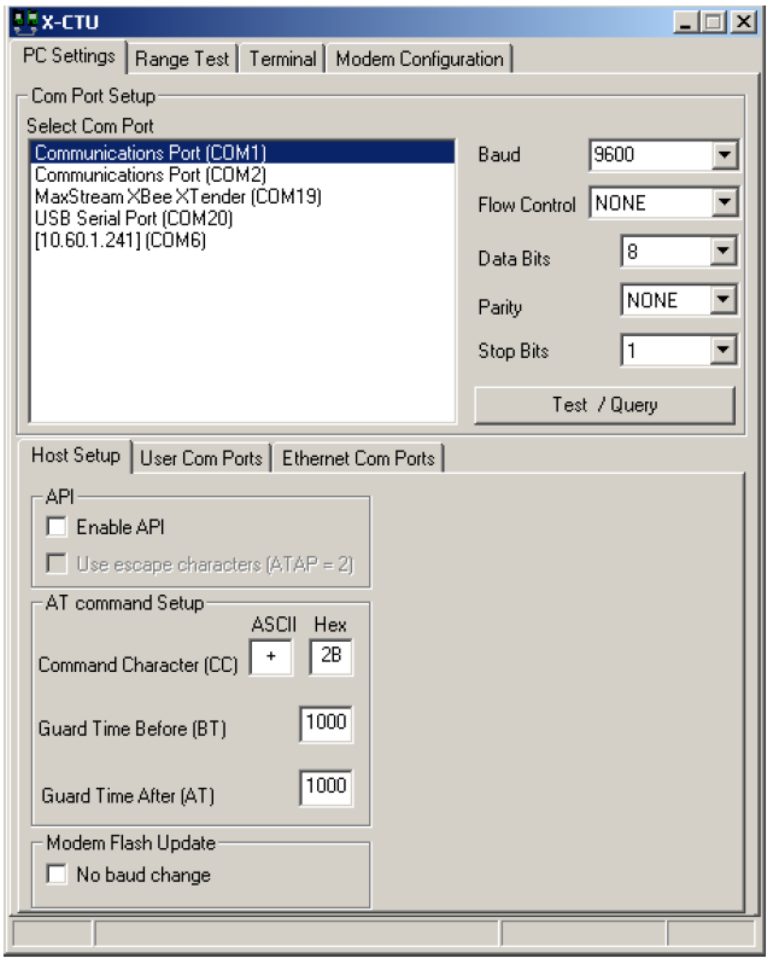
X-CTU is windows based application software developed by Digi for interacting with the RF radio modules, including 9-Xtend radio module. X-CTU has four main functions:

PC Settings: allows choosing the desired COM port and modifying PC parameters such as frame format and baud rate to be compatible with RF radio module.

Module Configuration: allows changing the configuration of RF radio module through a friendly graphical user interface. It can also be used to change the firmware of the radio module.

Range Test: Range test allows testing the communication range between two radio modules.

Terminal: Allows seeing the contents of the received packets.



Com Port Setup
Select Com Port
Communications Port (COM1)
Communications Port (COM2)
MaxStream XBee XTender (COM19)
USB Serial Port (COM20)
[10.60.1.241] (COM6)

Baud 9600
Flow Control NONE
Data Bits 8
Parity NONE
Stop Bits 1

API
Enable API
Use escape characters (ATAP = 2)

AT command Setup
ASCII Hex
Command Character (CC) + 2B
Guard Time Before (BT) 1000
Guard Time After (AT) 1000

Modem Flash Update
No baud change

Figure 36. X-CTU Software

Here we will explain some of the important parameters related to the radio module configuration under the *Module Configuration* tab (see Figure 36). More details can be found in the product datasheet.

Network ID: This parameter defines the network. Only nodes with the same network ID can communicate with each other.

DL and DH: destination address low and high. DL and DH form 64 bit address of the designation node. For mesh networking the destination address of all nodes should match with the source address of the base station.

SL and SH: source address low and high. SL and SH form 64 bit source address of the note. SH and SL are hardwired to the radio module and cannot be changed. Each radio module has a unique source address.

BD, NB, and SB: baud rate**,** party bits and stop bits parameters are used for configuring serial communication between the radio module and PC. PC and radio module should have identical serial communication settings.

PL: TX power level. This parameter controls the output power of the antenna. 9-Xtend radio module is power adjustable from 1mW to 1W depending on the application.

NQ: maximum number of route discovery retries allowed to find a path to the destination node. If NQ is zero, route discovery request will only be sent once.

NH: Number of network hops. This number doesn't limit the number of hops; rather, it is used to calculate maximum network traversal time and must be set the same on all nodes in the network.

HP: Hopping channel. Determines the spread spectrum channel on which module communicates. Separate channels minimise interference between multiple sets of modules operating in the same vicinity.

4.5 SENSING MODULE IMPLEMENTATION

As mentioned in CHAPTER 2 our wireless sensor node has two sensors: temperature and humidity. These sensors output analog signals; therefore we should convert them to digital values before calibration and interpretation. The following flowchart demonstrates the software routine for the sensor module. For source code (C) please refer to APPENDIX A.

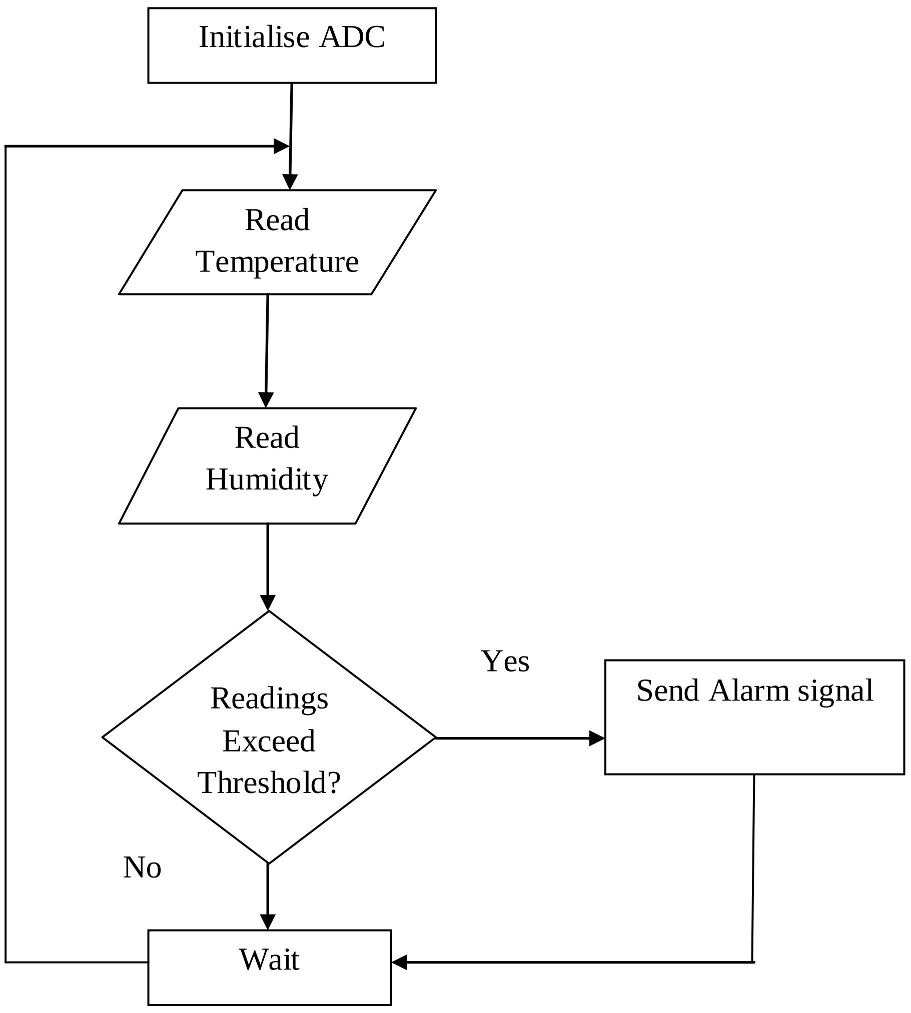


Figure 37. Sensor Module Flowchart

4.6 EEPROM IMPLEMENTATION

As mentioned in CHAPTER 2, EEPROM is connected to the microcontroller via the I^{2}C interface. In this section we will describe the software implementation of reading and writing from/to the EEPROM chip. In order to write to the EEPROM chip, the following steps have to be implemented consecutively.

1. Put start condition on the bus.
2. Transmit the address of the EEPROM and the R/W bit set to 0 (write operation).
3. Transmit the address of the register in the EEPROM we intend to write to.
4. Send the data byte.
5. Continue sending data if there are more than one byte.
6. Put stop condition on the bus.

The read operation is a bit more complicated because we have a combination of write and read cycles in the read operation. The following steps have to be taken consecutively during the read operation.

1. Put start condition on the bus.
2. Transmit the address of the EEPROM and the R/W bit set to 0 (write operation).
3. Transmit the address of the register in the EEPROM we intend to read from.
4. Put another start condition on the bus(repeated start).
5. Transmit the address of the EEPROM and the R/W bit set to 1 (read operation).
6. Read the data byte from the EEPROM.
7. Continue reading if desired.
8. Put stop condition on the bus.

It is important to note that I^{2}C is a byte oriented protocol; therefore any data or address put on the bus should be 8 bits long. This is why we send the device address (7bit) and the R/W bit in one frame to make it 8 bits long. From now on we will refer to the microcontroller as the master and EEPROM chip as the slave. For the source code (C) please refer to APPENDIX A.

Before explaining the details of implementation we will first describe the registers associated with I^{2}C hardware.

TWCR- TWI control register: This register is used for controlling the operation of TWI (two wire interface).

Table 10. TWCR Register

Bit No	7	6	5	4	3	2	1	0
Bit Name	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	__	TWIE
Bit Value	0	0	1	0	0	0	0	0

TWINT- TWI Interrupt Flag: This bit is set to 0 by the I^{2}C hardware whenever it has finished its current operation. This bit must be cleared with writing 1 to it by the application software.

TWSTA: by writing 1 to this bit, the device becomes master and a start condition is put on the bus.

TWSTO: writing 1 to it results in stop condition on the bus

TWEN: enables the I^{2}C

TWEA*:* writing 1 to this bit generates ACK

TWDR- TWI data register: This register is used for storing the data

Table 11. TWDR Register

Bit No	7	6	5	4	3	2	1	0
Bit Name	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD1
Bit Value	1	1	1	1	1	1	1	1

This address contains the next byte to be transmitted during the transmit mode and the last byte received during the receive mode.

TWSR- TWI status register: This register is used for retrieving information about the status of the I^{2}C logic and controlling the bit-rate prescaler.

Table 12. TWSR Register

Bit No	7	6	5	4	3	2	1	0
Bit Name	TWS7	TWS6	TWS5	TWS4	TWS3	—	TWPS1	TWPS0
Bit Value	1	1	1	1	1	1	1	1

TWS7-TWS3: these bits reflect the status of the I^2C logic.

TWS2: reserved.

TWPS1-TWPS0: prescaler bit rate. The possible values are shown below.

Table 13. Prescaler Setting Table

TWPS1	TWPS0	Prescaler value
0	0	1
0	1	4
1	0	16
1	1	64

Now that we know the details of the registers associated with the I^2C, we will explain in detail how to read and write from/to the EEPROM. The logic used here applies to any I^2C compatible slave device and therefore can be reused with minimal modification.

The first thing we need to do is to calculate the SCL frequency. SCL frequency is calculated using the following equation:

$$\text{SCL Freq} = \frac{\text{F\_CPU}}{16+2(\text{TWBR}).(\text{Prescaler Value})} \quad (5)$$

We will operate the I^2C in 100 MHz or less as recommended by the datasheet. F_CPU or CPU frequency is 1MHz and we set pescaler value to be 1 (refer to table 8). We will choose SCL frequency to be 50 KHz; therefore the TWBR comes to be 2. After setting the SCL frequency we are ready to work with the I^2C interface. We will start with explaining the write operation and then the read operation. The algorithm is shown below, for source code (C) please refer to APPENDIX A.

4.6.1 WRITE CYCLE

1. Send START condition, enable the TWI and write 1 to TWINT.
2. Wait for TWINT Flag set. This indicates that the START condition has been transmitted.
3. Check value of TWI Status Register to make sure start condition has been put on the bus.
4. Load the EEPROM address and the W bit into TWDR Register. Clear TWINT bit in TWCR to start transmission of address.
5. Wait for TWINT Flag set. This indicates that the EEPROM address and the R/W has been transmitted, and ACK has been received.
6. Check the value of the TWI Status Register to make sure the ACK has been received.
7. Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data.
8. Wait for the TWINT Flag to be set. This indicates that the DATA has been transmitted, and ACK has been received.
9. Check the value of the TWI Status Register to make sure ACK has been received.
10. Continue sending data if desired.
11. Transmit STOP condition.

4.6.2 READ CYCLE

1. Send START condition, enable the TWI and write 1 to TWINT.
2. Wait for the TWINT Flag to be set. This indicates that the START condition has been transmitted.
3. Check value of the TWI Status Register to make sure the start condition has been put on the bus.
4. Load the EEPROM address and the W bit into te TWDR Register. Clear the TWINT bit in the TWCR to start the transmission of address.
5. Wait for the TWINT Flag to be set. This indicates that the EEPROM address and the R/W have been transmitted, and ACK has been received.

6. Check the value of the TWI Status Register to make sure that ACK has been received.
7. Load the address of the internal register in EEPROM we intend to read from into the TWDR Register. Clear the TWINT bit in TWCR to start the transmission of data.
8. Wait for the TWINT Flag to be set. This indicates that the DATA has been transmitted, and ACK has been received.
9. Check value of the TWI Status Register to make sure that ACK has been received.
10. Send START condition (repeated start).
11. Wait for TWINT Flag to be set. This indicates that the START condition has been transmitted.
12. Check value of TWI Status Register to make sure that the start condition has been put on the bus.
13. Load the EEPROM address and the R bit into the TWDR Register. Clear the TWINT bit in TWCR to start the transmission of address.
14. Wait for the TWINT Flag to be set. This indicates that the EEPROM address and the R/W have been transmitted, and ACK has been received.
15. Check the value of TWI Status Register to make sure that ACK has been received.
16. Read the data from the bus.
17. Continue reading data if desired.
18. Transmit STOP condition.

CONCLUSION

In this chapter we have explained the software implementation of the developed wireless sensor platform. Initially we discussed the development platform and covered topics such as how to compile and load the program into to the microcontroller and etc. Then we explained how to use and configure peripheral devices inside the microcontroller such as ADC and USART. Then we described the implementation of the sensing module and the configuration of the RF radio module. Topics included setting of registers, how to configure the networking parameters, etc. This chapter only explained the high level implementation of the software. For source code please see APPENDIX A.

CHAPTER 5. COMMUNICATION AND NETWORKING

INTRODUCTION

We discussed the hardware and software implementation of the developed wireless sensor platform in the earlier chapters. In this chapter we will cover the communication and networking features of the developed wireless sensor platform. We will start this chapter by explaining the basic hardware and electrical characteristics of the RF radio module and then describe advanced networking features and their implementations.

The communication device used in our system is 9-Xtend RF radio module. The module accepts asynchronous serial data from the host device, communicates within 900MHz unlicensed band, has maximum data throughput of 115 kbps, can achieve 64 kilometres line of sight range and has built in networking functions.

We chose 9-Xtend RF radio module, due to its superior range and built in mesh networking capabilities. Long communication range is essential for environmental monitoring applications which when coupled with mesh networking feature, will let us cover large geographical areas and enhance the reliability of the system as a whole.

5.1 HARDWARE AND ELECTRICAL CHARACTERISTICS

The external interface of the 9-Xtend RF radio module as shown in Figure 38 is composed of an SMA connector for mounting the antenna (not shown in the picture) and 20 equally spaced pins for serial communication and other functionalities.

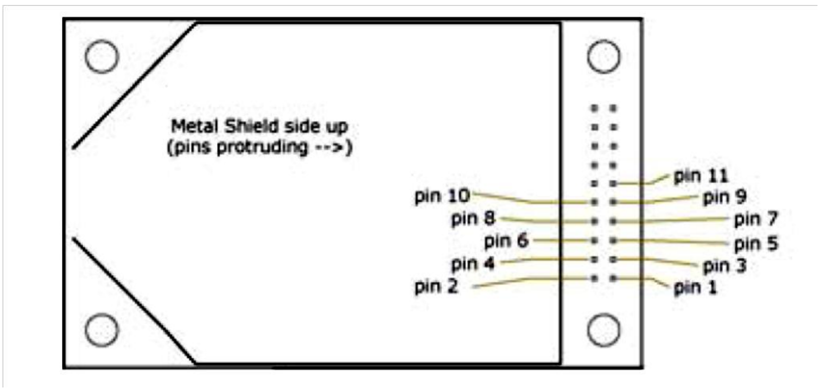


Figure 38. 9-Xtend Pin Numbers [22]

Due to the compact design of RF 9-Xtend radio modules (3.64cm x 6.05cm) they can easily be integrated into small embedded systems without occupying much space. They can communicate directly with any UART enabled device such as a microcontroller and other protocols such as RS-232 using a level shifter. Pin names and their functionalities are shown in the table below.

Table 14. 9-Xtend Pin Descriptions

Number	Name	Input/output	Function
1	GND	N/A	Ground
2	V_{CC}	I	Power supply
5	DI	I	Serial data in
6	DO	O	Serial data out
4	TX_PWR	O	Pulses low during RF data transmission
3	RX_LED	O	Goes high when data is received
7	SHDN	I	Enable pin for shutdown mode
8-20	N/A	N/A	N/A

Data communication starts when the first byte of data appears in DI buffer of the transmitting module. If the 9-Xtend RF module A is not actively receiving RF data, data in the DI buffer are grouped into RF packets and then transmitted over-the-air to 9-Xtend RF module B.

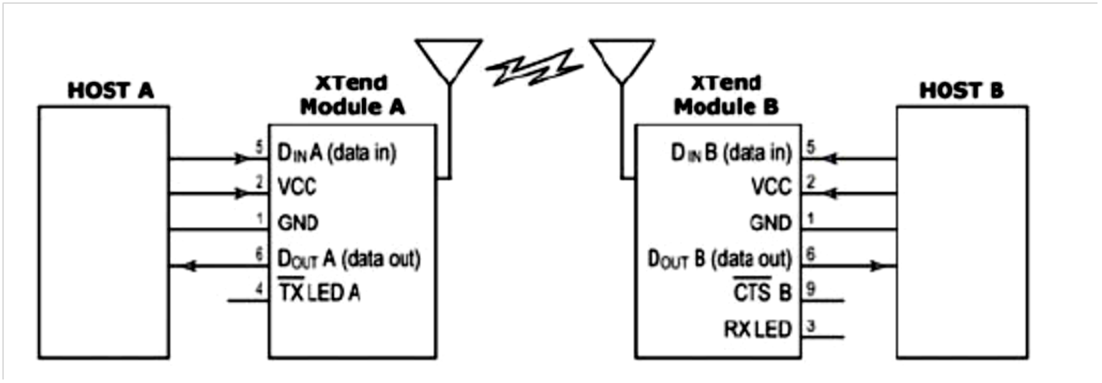


Figure 39. Basic RF Link Between Hosts [22]

As an example, let us assume that we want to transmit a single byte from host A to host B. Host A which in our case is a microcontroller, transmits a single byte to DI pin, which is stored in DI buffer of host A. Once host A packetised and transmitted the message, TX LED A (TX_PWR A) briefly pulses low to indicate data transmission. At the receiving end (module B), once the antenna detects RF activity, RX LED briefly pulses high to indicate something is being received and if the packet passes the CRC check, it is sent out to DO pin of module B. This example demonstrated the most basic RF link between two modules. Later we will see that the received RF packets must meet multiple criteria other than the CRC check in order to be sent out to host B. Also at the transmitting side we can adjust the RF packet length and other criteria that must be met before a transmission can take place.

5.2 SERIAL COMMUNICATION

The RF 9-Xtend module communicates with the host device through a TTL-level asynchronous serial port. Through the serial port, the module can communicate with any UART voltage compatible device or through a level translator to any serial device RS-232 or USB.

Data enters the RF 9-Xtend module through pin 5 (DI) as an asynchronous serial signal. Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). The following Figure illustrates the serial bit pattern of data passing through the module.

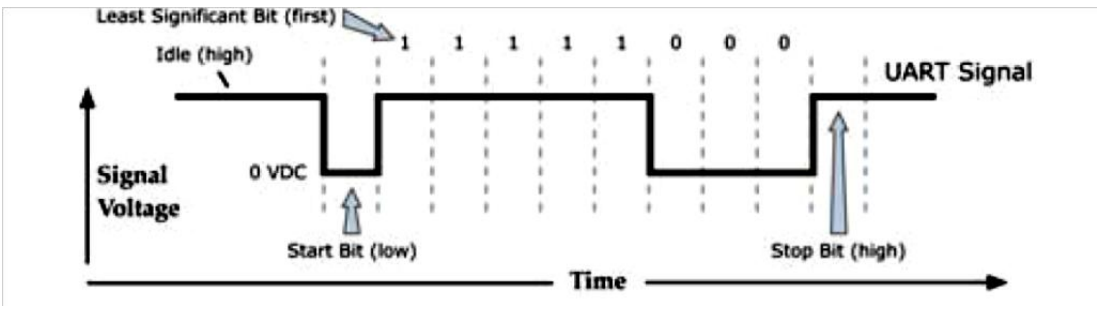


Figure 40. Transmission of Data Packet "31" in 8-N-1 Data Format [22]

It is important to note that the serial communication between the host device and the RF 9-Xtend can only happen if the two devices have the same frame format.

When data enters the module through the DI pin, the data is stored in the data buffer until it can be processed. There are two conditions defined by RB and RO parameters that must be satisfied before the data in the DI buffer is packetised. The RB parameter defines the number of bytes that should be in the DI buffer before packetisation and transmission can take place. The RO parameter defines the duration of inactivity on the serial bus before packetisation and transmission can take place. It is important to note that the two conditions above must not be met simultaneously. Satisfaction of any of them can result in packetisation and transmission of data. The following diagram demonstrates the internal data flow within the RF 9-Xtend radio module.

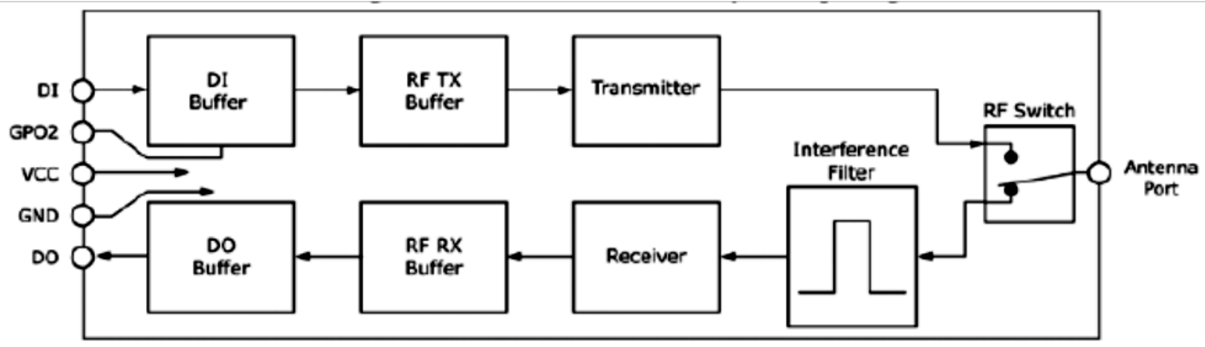


Figure 41. Internal Data Flow Diagram [22]

Data loss can happen if data continuously flows into DI buffer without any transmission taking place. This can happen when the module is actively receiving data and data flows into the DI buffer at the same time. In order to avoid this situation we will set the RF data rate to be higher than the serial data rate.

5.3 MODES OF OPERATION

The 9-Xtend RF radio module operates in six modes (see Figure 42). 9-Xtend RF modules can be in one mode at a time.

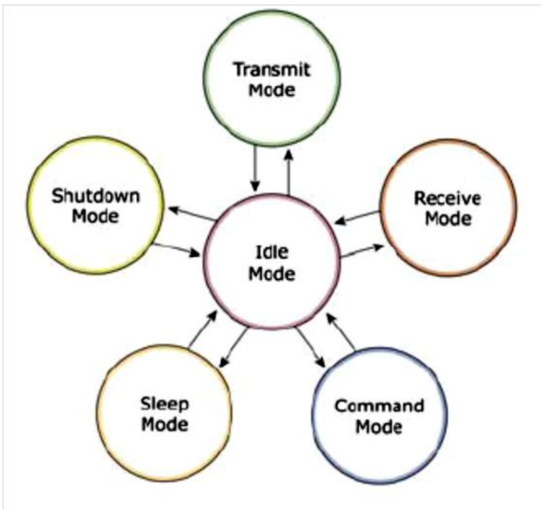


Figure 42. 9-Xtend RF Module's Modes of Operation [22]

When not receiving or transmitting, the RF module is in idle mode. The module shifts into other modes of operation under the following conditions:

- Transmit Mode: Serial data is received in the DI buffer
- Receive Mode: Valid RF data is received through the antenna
- Shutdown Mode: Shutdown condition is met
- Sleep Mode: Sleep mode condition is met
- Command Mode: Command mode sequence is issued

When SHDN pin is driven low, the RF module goes into hardware sleep (shutdown) mode. This mode has the least power consumption. Once in shutdown mode any data reception or transmission will be halted and the stored data in the DI buffer will be lost. Unfortunately the DIGIMESH compatible version of 9-Xtend does not support sleep mode, therefore in our case the RF module never enters sleep mode.

In order to modify or read the RF module's parameters, the module must first enter into Command Mode. Command Mode and RF module's parameters will be discussed in more detail later in this chapter.

5.2.2 TRANSMIT MODE

Once the first byte of data has been received in the DI buffer, the 9-Xtend RF module attempts to shift into transmit mode and start communicating with neighbouring 9-Xtend RF modules. There are two conditions that when either of them is satisfied, the 9Xtend RF module packetises the data and starts transmission. These two conditions are as follows:

1. Packetisation Threshold: RB or packetisation threshold specifies the number of data bytes that must be received in the DI buffer before the 9-Xtend RF module can group them into a single RF packet and transmit. The maximum value of RB parameter is PK (maximum RF packet size).
2. The second condition specifies that at least one character is in the DI buffer and is waiting for transmission and RO times of silence have been observed on the serial line. RO or packetisation timeout parameter defines the duration of inactivity on the serial line before the 9-Xtend RF module can packetise and transmit the data in DI buffer. We can disable the RO parameter by setting it to zero. By doing this, RF transmission does not start unless RB bytes of data have been received in the DI buffer.

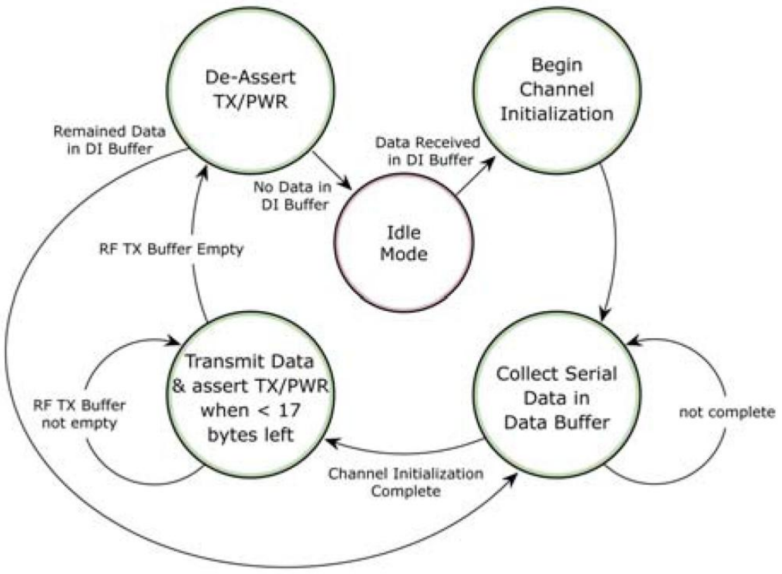

De-Assert TX/PWR
Begin Channel Initialization
Remained Data in DI Buffer
Data Received in DI Buffer
No Data in DI Buffer
Idle Mode
RF TX Buffer Empty
Transmit Data & assert TX/PWR when < 17 bytes left
Collect Serial Data in Data Buffer
RF TX Buffer not empty
not complete
Channel Initialization Complete

Figure 43. Transmit Mode State Diagram [22]

Figure 43 shows the state diagram of the transmit mode. If data is detected in the DI buffer, channel initialisation begins. During the channel initialisation, the receiving module synchronises itself with the transmitting module and incoming data is stored in the DI buffer. Once any of the two conditions mentioned above has been met and channel initialisation is complete, the 9-Xtend RF module starts transmission of data. When there are less 17 bytes left before the packetisation is complete, TX_PWR is inserted to indicate transmission activity. After transmission of the RF packet, the 9-Xtend RF module checks the DI buffer to see if there are more data left. If yes, packetisation and transmission starts again; if no, TX-PWR is de-asserted and the module shifts back into idle mode until data is detected in the DI buffer. Note that transmission cannot start if the module is already receiving RF data.

The contents of an RF packet are shown in Figure 44. The RF initialiser which is transmitted first contains information such as hopping pattern utilised by the transmitter and other information that help the receiving module to synchronise with the transmitting module. RF data is divided into three sections. Payload is the user data, header contains networking and filtering information and CRC (cyclic redundancy check) helps to detect corrupted packets and discard them. The header section of RF data is composed of 5 parameters namely ATMY (source address or transmitter address), ATID (network ID), ATHP (hopping pattern), ATDT (destination address) and packet ID. Note that RF modules must have the same ATID (network ID) and ATHP (hopping pattern) in order to communicate with each other.

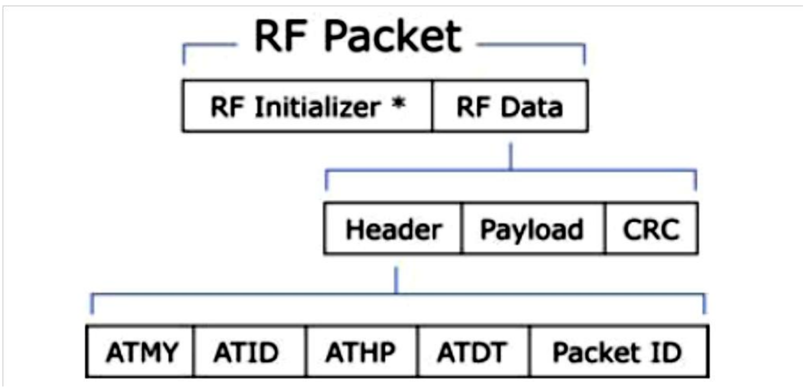


Figure 44. Structure of RF Packet [22]

5.2.3 RECEIVE MODE

Figure 45 shows the state diagram of the receive mode. If data is detected, the RF module shifts into receive mode from idle mode. Once the packet is received, the RF module examines the header. If the ATID and ATHP parameters match with the receiving module's ones, RX_LED is asserted to show reception activity otherwise the packet is ignored and the RF module falls back into idle mode. Then the RX_LED is de-asserted and CRC and address checks are done on the packet. If the packet passes both of these tests, only then the payload is passed to the DO buffer otherwise the packet is ignored and the module falls back into idle mode. The above process is repeated as long as there are data detected. Note that the receive mode and transmit mode's state diagrams described, do not incorporate acknowledgments and retries. These will be covered in the DIGIMESH section.

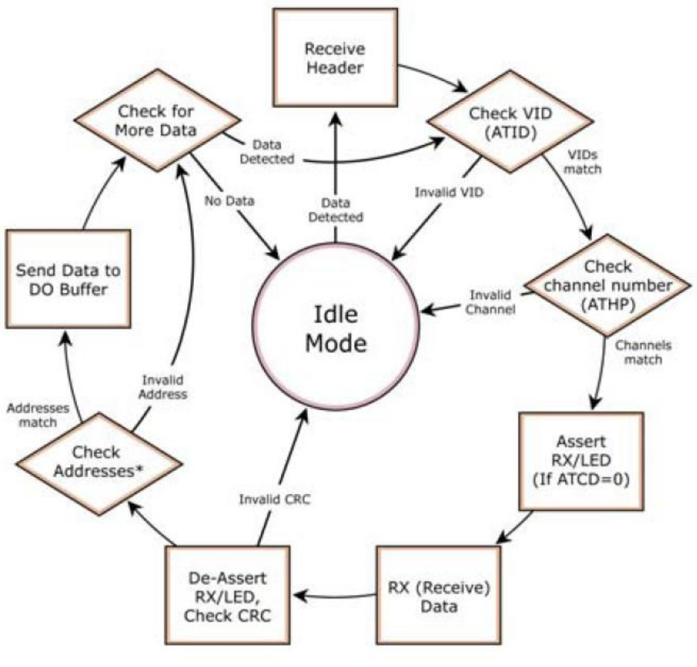


Figure 45. Receive Mode State Diagram [22]

5.3 ADDRESSING

The incoming RF data must be filtered through three stages in order to go through to DO buffer. In the first stage, hopping channel (HP) is compared to the HP of the receiving module, during the second stage network ID is compared and finally in stage three the destination address of the packet is examined. If the packet fails any of these tests it will be discarded.

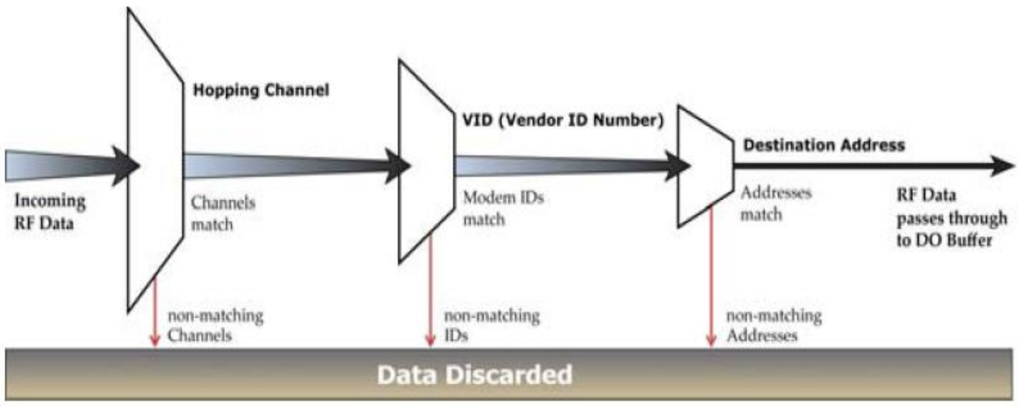


Figure 46. Packet Filtering in 9-Xtend RF Module [22]

Packets can be addressed to a specific module or a group of modules. Packets might pass through several intermediate nodes in order to get to the final destination. Address recognition at the receiving module is according to the following procedure. Initially the destination address of the received packet is examined and compared to RX_MK (address mask of receiving module). If it matches the address mask, then the packet is accepted as a global packet. Global packet means that the packet is not directly addressed to the receiving module, but to a group of modules or the packet must pass through this module in order to get to the final designation. If the destination address and RX_MK are not the same, the module compares the destination address of the received packet with its source address (RX_MY) to see whether the packet is local (addressed to this module); otherwise it discards the received packet.

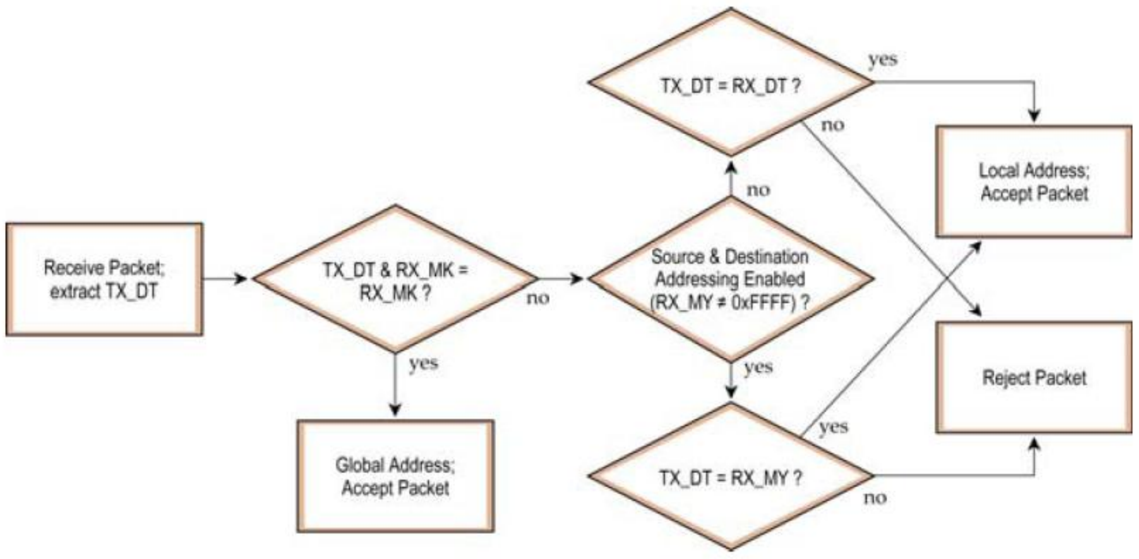

Receive Packet; extract TX_DT
TX_DT & RX_MK = RX_MK ?
yes
Global Address; Accept Packet
no
Source & Destination Addressing Enabled (RX_MY ≠ 0xFFFF) ?
no
TX_DT = RX_DT ?
yes
no
Local Address; Accept Packet
Reject Packet
yes
TX_DT = RX_MY ?
yes
no

Figure 47. Address Recognition at Receiving Module [22]

5.4 DIGIMESH

DIGIMESH, a proprietary mesh network protocol is the networking protocol used in our sensor node. In a mesh network, messages are routed through several nodes to reach the final destination. Other than the increased range, DIGIMESH offers unique set of capabilities that makes it suitable for our sensor node.

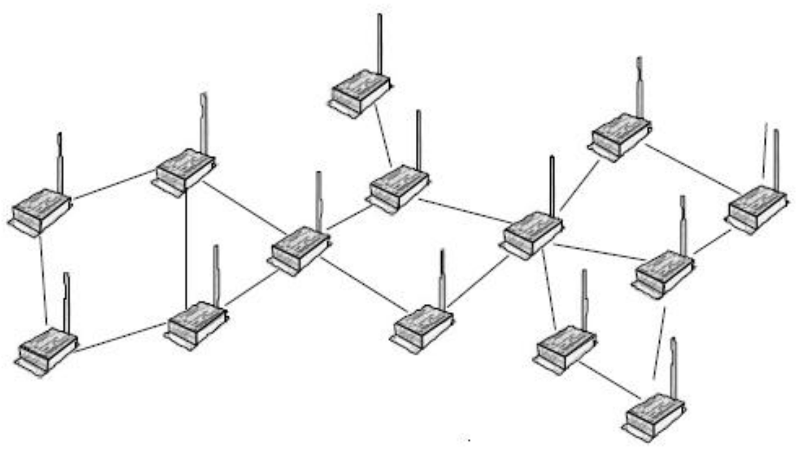


Figure 48. Sample Mesh Network Topology [22]

Some of the important features of DIGIMESH include:

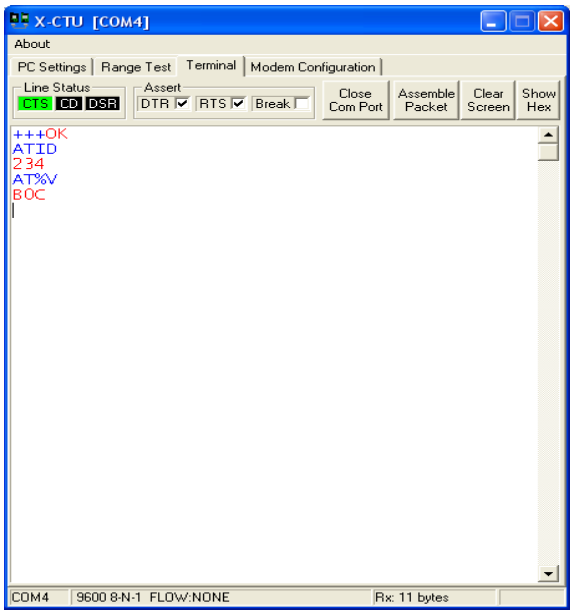
- Self healing
- Flexibility to expand network
- No need for expensive gateway routers
- Reliability

Self-healing means that during node failures, the network can find alternative path to the destination. Flexibility to expand the network is due to the fact any node can be added and removed from the network without affecting the functionality of the network as a whole. The homogenous nature of the network means that all the nodes have equal functionality, therefore there is no need for any gateway node with enhanced functionality. This makes the configuration of the network substantially easier. And finally, reliability is achieved through acknowledgments and retransmissions.

The routing algorithm in DIGIMESH is very similar to AODV (Ad-hoc On-demand Distance Vector) algorithm. There is an associative routing table for each node that maps the destination address to its next hop address. Therefore, a message from a source node will go through routing nodes until it reaches the destination node. When the source node doesn't have a route for the specified destination it will initiate a route discovery process. During route discovery process, the source node broadcasts a Route Request message. Upon reception of Route Request message, the intermediate nodes rebroadcast the Route Request message if they don’t have a better route back to the source node, otherwise they drop the message. When the destination node eventually receives the Route Request message, it unicasts a Route Reply message back to the source node. The source node might receive multiple Route Reply messages: it will choose the one with the best round trip route quality. The destination address of the source nodes has to match with the source address of the destination node. At the other end of the network, the destination node is connected to a PC, where contents of the packets can be viewed using the X-CTU software.

5.5 CONFIGURATION

The RF module can be configured using AT commands entered through terminal tab of X-CTU software as shown in the following Figure.



```
+++OK
ATID
234
AT%V
B0C
```

Figure 49. Terminal Tab of X-CTU Software [36]

The module enters command mode by typing special characters "+++" and pressing enter in the terminal window. System response appears red and user commands are in blue. For example if we want to view the network ID of the module, we type ATID and the system outputs the ID of the network in red. If we want to change a parameter's value, we write the desired value of the parameter next to the command and issue the command ATWR afterwards to write the new value to the non-volatile memory of the RF module. For example let's say we want to change the network ID of the module to 2346. Then we use the following commands.

```
+++
ATID2345
ATWR
ATCN
```

ATCN command shifts the module out of command mode into idle mode. We will now briefly explain some of the important parameters associated with DIGIMESH version of 9-Xtend RF radio module.

Table 15. List of Important AT Commands

Command	Function	Category
PL	Transmitter output power level	RF Interfacing
WR	Write	Special
DH	Destination address high	Networking
DL	Destination address low	Networking
HP	Hopping channel	Networking
ID	Network ID	Networking
NH	Maximum number of network hops	Networking
NQ	Maximum number of Route Requests	Networking
NN	Network delay slots	Networking
NR	Maximum number of retries	Networking
SH	Source address high	Networking
SL	Source address low	Networking
BD	Baud rate (bit rate)	Serial Interfacing
RB	Packetisation threshold	Networking
RO	Packetisation timeout	Networking
SB	Number of stop bits	Serial Interfacing
NB	Number of parity bits	Serial Interfacing

CONCLUSION

In this chapter we have covered the communication and networking features of the developed wireless sensor platform. We started by describing the hardware and electrical characteristics of the 9-Xtend RF radio module. Then we covered serial communication and modes of operation of the module. After gaining detailed insight about the basic operation of the RF module, we described DIGIMESH, an advanced proprietary mesh networking protocol. Finally, we showed how to read and write the configuration parameters of the 9-XTend RF radio module.

CHAPTER 6. TESTING AND RESULTS

INTRODUCTION

In earlier chapters we explained how we developed our wireless sensor platform in terms of hardware, software and fabrication. In this chapter we will conduct several tests on the newly developed wireless sensor platform to verify its limitations and strengths. We will start with the physical layer and demonstrate how the user data is modulated and transmitted. Then we will test the transmission range under different conditions. Finally we will conduct tests on the networking side of the design and verify the correct operation of the DIGIMESH protocol.

6.1 MODULATION AND SPREAD SPECTRUM

The modulation technique used in 9-Xtend RF radio module is BFSK (Binary frequency shift keying). In BFSK digital data is carried using discrete variations in the carrier frequency. In other words we will represent 0s with one frequency and 1s with another frequency. In the time domain the modulated signal would look like something similar to Figure 50.

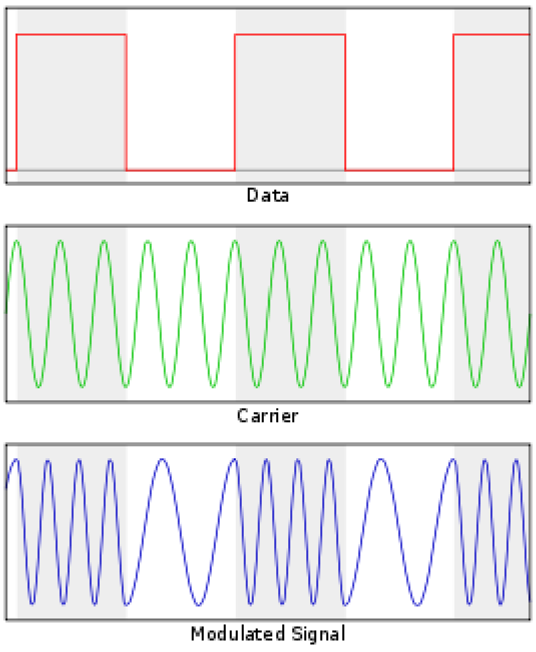


Figure 50. BFSK in Time Domain

The 9-Xtend RF radio module also utilises Frequency Hopping Spread Spectrum (FHSS). In FHSS, the carrier frequency is varied within the available bandwidth according to a pseudo-random function. It means that the carrier signal will change with time and is not fixed anymore. Figure 51 through to Figure 53 demonstrate the frequency response at three different times using a spectrum analyser. In an ordinary BFSK, there will be two fixed peaks in the frequency domain, but because of frequency hopping the carrier signal continually changes. The HP (Hopping Channel) parameter we mentioned in the last chapter defines the hopping sequence used. We also said that the communicating modules must have identical HP values. This is because the receiver should know what hopping security the transmitter is utilising, otherwise it will not recognise which frequency represents 1s and which one represents 0s. This method of frequency hopping also increases the security of the network, because the attacker cannot interpret the signals without knowing the hopping sequence pattern.

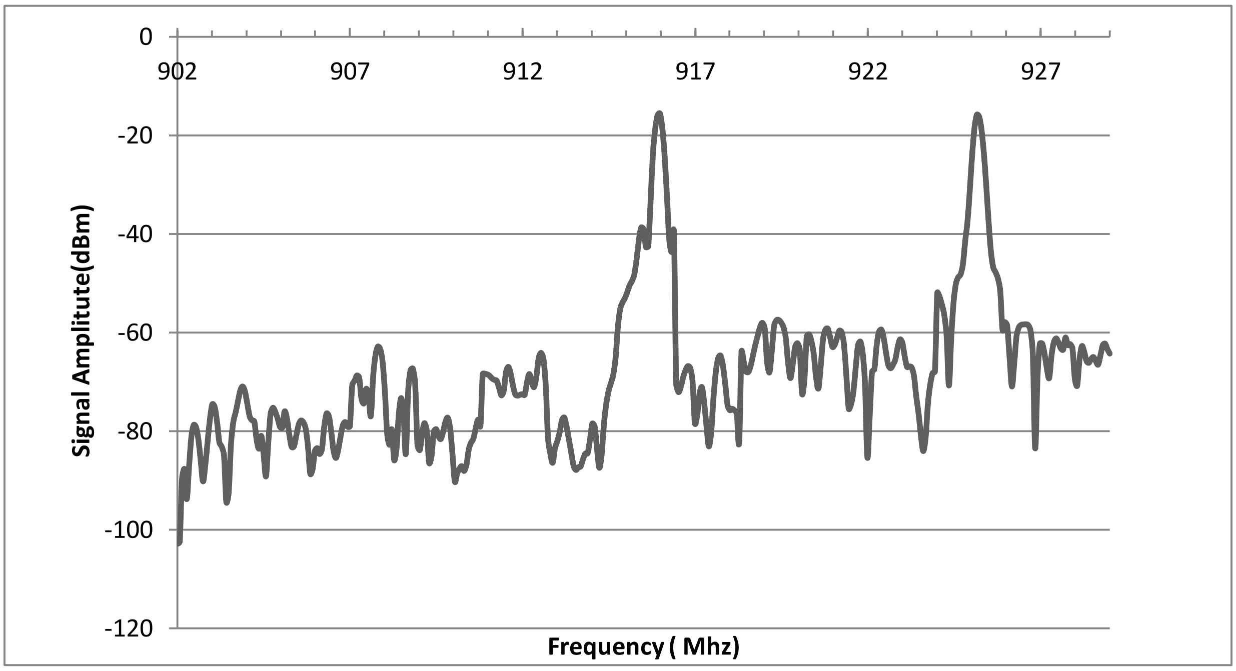


Figure 51. BFSK at Frequency Domain

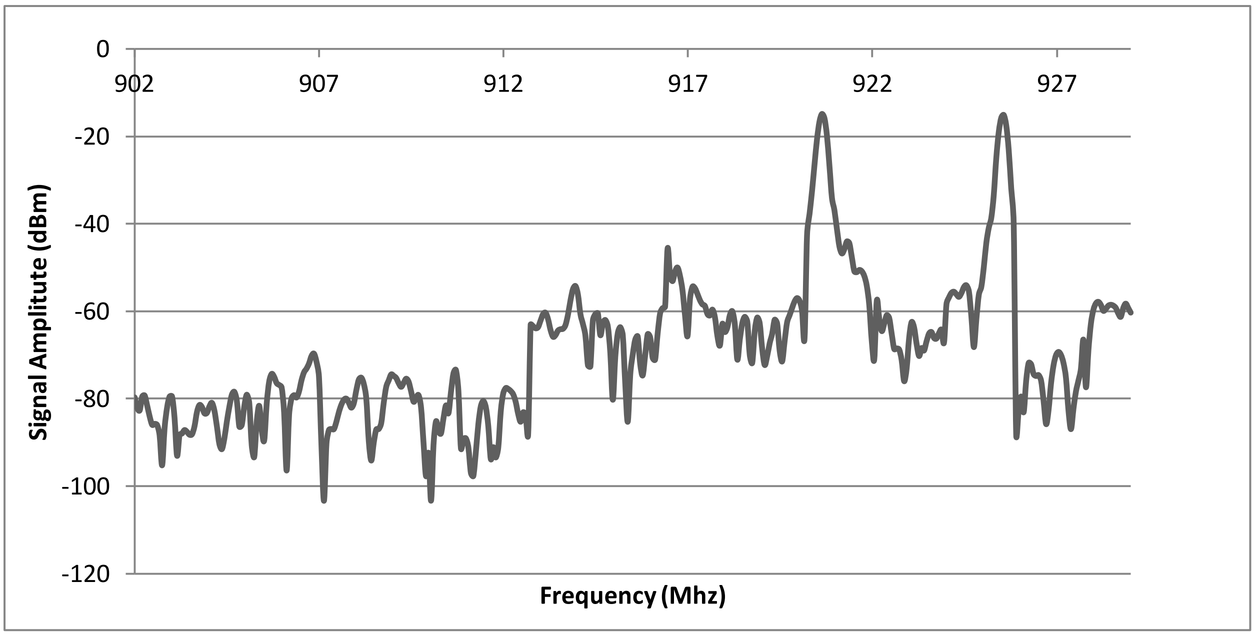


Figure 52. BFSK at Frequency Domain

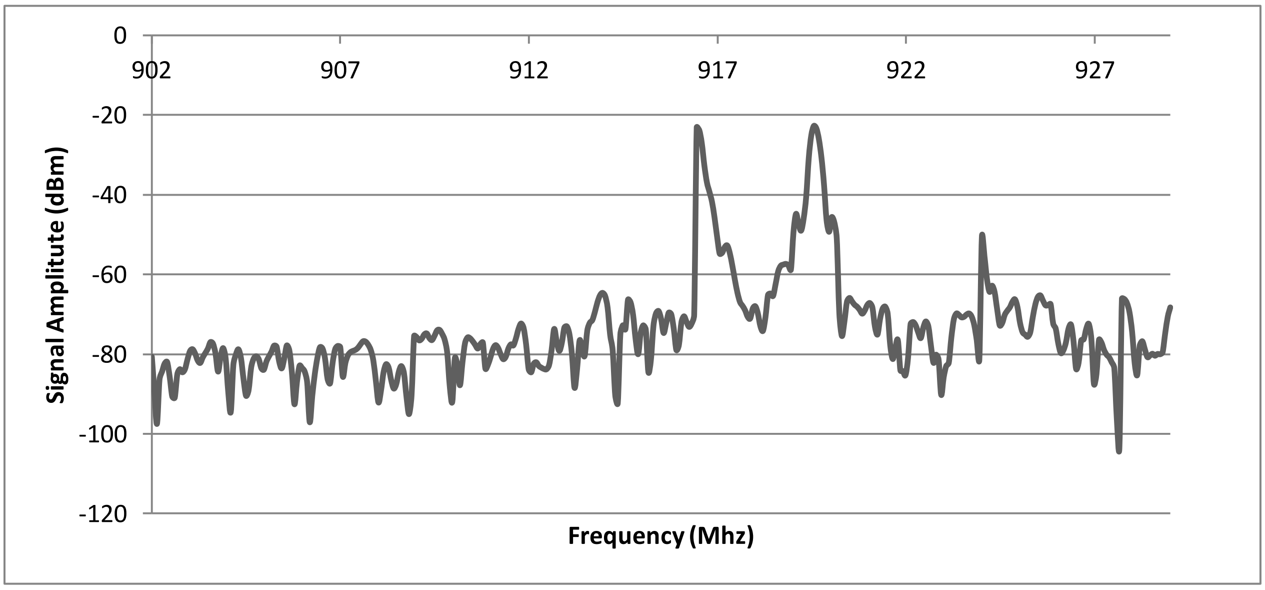


Figure 53. BFSK at Frequency Domain

6.2 NETWORKING AND RANGE TEST

The stated range according to the 9-Xtend RF radio module's datasheet is 64 kilometres. This range can only be achieved when the transmitter and receiver are within line of sight and high gain antennas are used. We have conducted several tests and measurements to verify the communication range of the 9-Xtend RF radio module under different conditions. It should be noted that we have used a dipole antenna (2.1 dBi), therefore the range is reduced. The receiver sensitivity of the 9-Xtend RF radio module is -110 dBm at 9600 bps, which means that the receiver will not detect any useful signal below -110 dBm. The following graph shows the signal strength versus distance plot in a densely vegetated area. The maximum communication range attained was 1100 meters. Similar tests were conducted in urban (see Figure 55) and outdoor (line of sight) conditions. The maximum attained line of sight range was 23 kilometres and 650 meters in residential areas. As can be seen in Figure 55, the signal strength versus distance plot has a steeper slope. This is because in built up areas, concrete walls and other obstacles more quickly reduce signal strength and RF signal cannot penetrate as easy as a vegetated area.

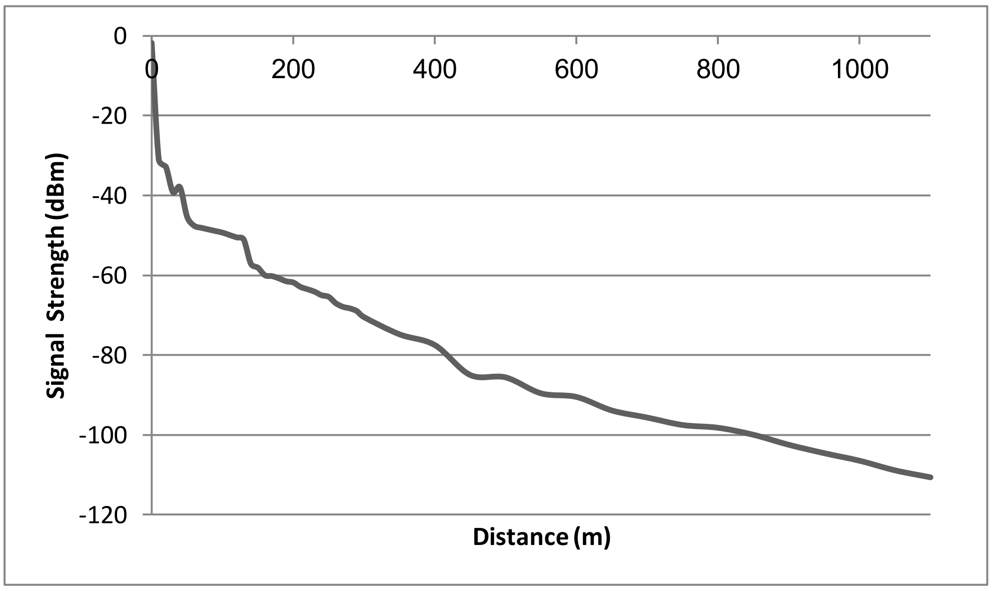


Figure 54. Signal Strength vs. Distance Plot for Densely Vegetated Area

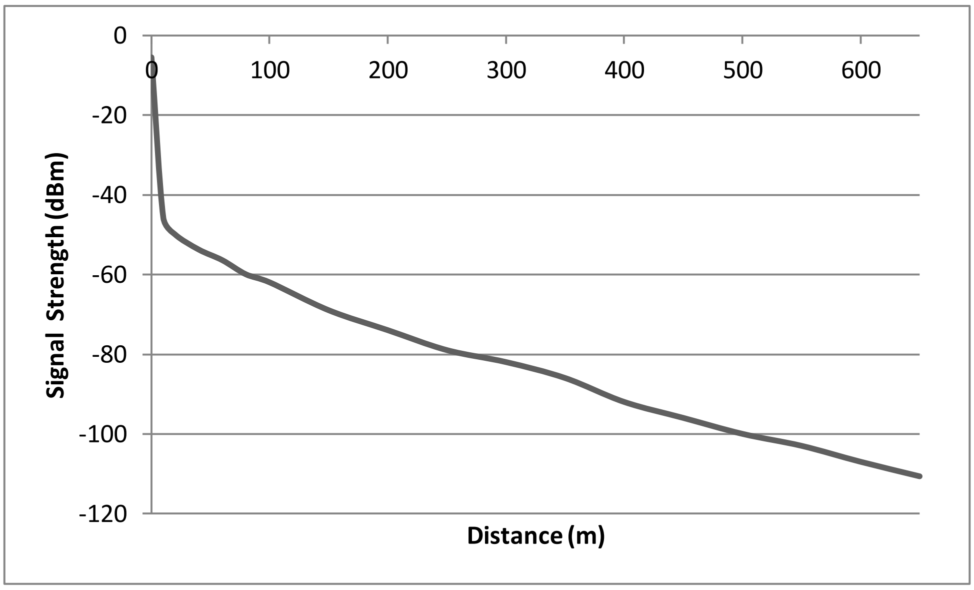


Figure 55. Signal Strength vs. Distance Plot for an Urban Area

It should be noted that the communication range can be affectedby the nature obtacles. For example, type and denstisty of vegeration, materials used in buldings, insulatorts, etc.

As shown in Figure 56, a mesh network consisting of six sensor nodes has been constructed in a test scenario covering a reasonable size urban area in Perth, Western Australia. All the sensor nodes have been uploaded with DIGIMESH firmware. We choose one of the sensor nodes to be the destination node. Once the destination node is chosen, all the other nodes can generate and send packets to the destination node. This prototype network was tested in the Packwood suburb of Perth, Western Australia. The purpose of this test was to verify the correct operation of mesh network and the self healing attribute of the network.

During the first stage of the test, packets are routed in the following order: source> node 1 > node2 > destination. Then, in the second stage node 1 is shut down and node 3 and node 4 are placed in the network, therefore the source node has to configure a new path to the base node which is Source>Node 3>Node 4>destination.

At the destination node, the sensor node is connected to laptop via the USB interface and the received packets can be viewed in the terminal tab of X-CTU software.

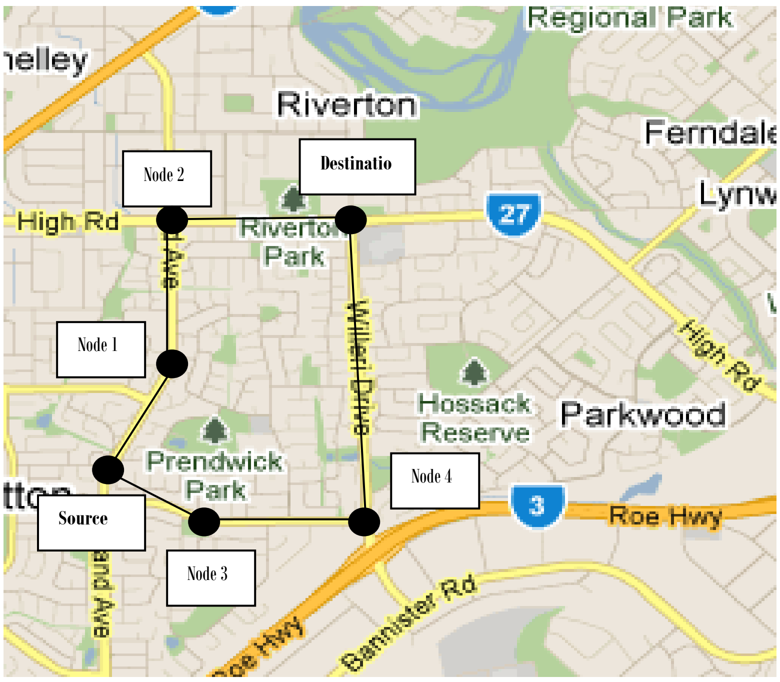


Figure 56. Mesh Networking Test in the Suburb of Parkwood, W.A.

CONCLUSION

In this chapter we have explained the testing procedure of the developed wireless sensor platform and the achieved results. Initially we described the modulation and spread spectrum techniques used in the 9-Xtend RF radio module and verified them using a spectrum analyser.

We conducted several tests for verifying the communication range and networking features of the developed wireless sensor platform. Firstly we conducted three range tests to find the transmission range of the developed wireless sensor platform in residential areas, outdoors (line of sight) and densely vegetated areas. For residential areas the achieved range was 650 meters. For outdoor (line of sight), the achieved range was almost 22 kilometres, and finally, in densely vegetated areas the maximum attained range was 1100 meters. Another test was conducted to verify the correct operation of the mesh network and its features such as self healing and route discovery abilities.

CHAPTER 7. CONCLUSION

This chapter summarises the main outcomes of the thesis and briefly evaluates the achieved results. The FUTURE WORK section explains the future trend in wireless sensor networks and how this research may be extended.

Initially, the main objective of this thesis was to develop a wireless sensor network for environmental monitoring applications. A literature review was conducted to see if the existing wireless sensor products can adequately address the needs of environmental monitoring applications. The most important characteristic we were after was the communication range of the existing wireless sensor nodes. Other important properties such as low power consumption, size and cost were also considered but they were not as critical as the communication range requirement.

In order for wireless senor networks to be feasible for environmental monitoring applications, they need to be able to cover large geographical areas in the order of tens of kilometres. Most of the popular wireless sensor nodes such as TelosB and MICAz have communication range of around two hundred meters. These wireless sensor nodes perform reasonably well for indoor applications. They have very low power consumption, small size, low cost and very user friendly APIs. However, the transmission range for these devices is not sufficient for environmental monitoring applications. Consequently, we decided to develop a wireless sensor network that could address the transmission range shortfall of the existing wireless sensor nodes.

We started the implementation of the wireless sensor node by first designing the hardware platform. Then we integrated an RF radio module to the designed hardware for communication purposes. After the implementation of the hardware, the developed module was programmed, and finally the complete system was tested and the anticipated outcomes were achieved and verified.

The hardware platform of the developed sensor node is composed of a microcontroller, temperature and humidity sensors, transceiver, external memory, expansion connectors for additional sensors, USB interface, antenna and in system programmer (ISP) for the microcontroller. Three RF radio modules were evaluated based on transmission range, power consumption, size and cost. Xbee-PRO, 9-Xtend and RMX-232 were evaluated. The 9-Xtend was the chosen RF transceiver due its superior range (64 kilometres) and support for mesh networking. Temperature and humidity sensors are already integrated on the board and have been calibrated. These sensors were chosen due to their popularity in environmental monitoring applications. However, there are I^2C and SPI connectors for additional digital sensors. There is also a USB interface for connecting the developed wireless sensor node to a PC, and an EEPROM interfaced to the microcontroller via the I^2C interface. A simplified block diagram of the system is shown in Figure 57.

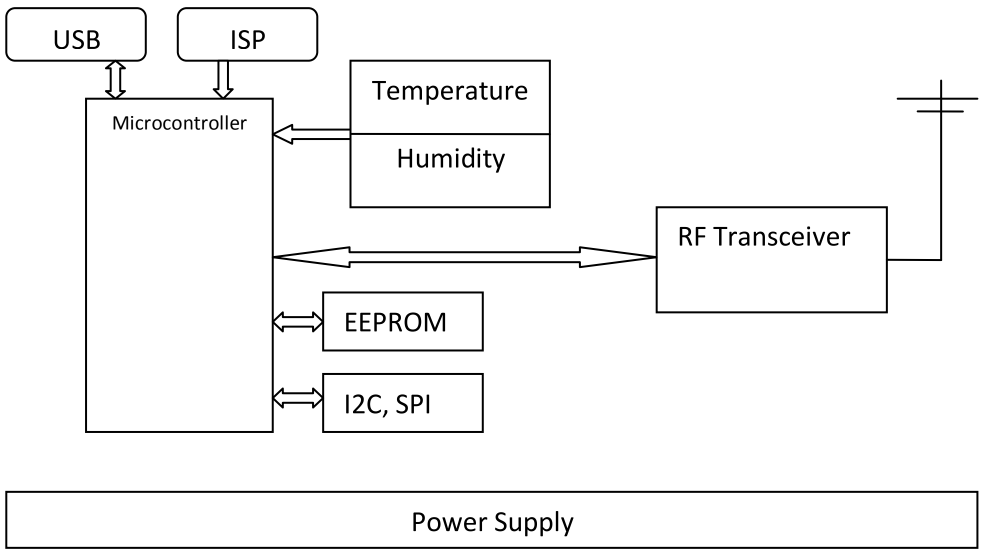


Figure 57. Simplified Block Diagram of the System

The software for the developed wireless sensor platform was written in C using the AVR Studio and GCC complier. The written software first initialises the system and then controls monitoring functions such as sensing and processing. During the initialisation, the peripheral devices such as UART and ADC are enabled and the associated registers are set to appropriate values. After the initialisation of the system, sensors are read and calibrated. If the sensor readings exceed a predefined threshold, the processor generates an alarm signal and communicates the alarm to the neighbouring nodes where it eventually reaches the base station or the destination through the mesh network.

The chosen network topology for communication is DIMIMESH protocol which is a proprietary mesh protocol. The configuration of the networking parameters including the RF radio module settings is done using the APIs in X-CTU software. DIMIMESH offers unique advantages such as self healing, easy configuration, extended range, etc.

We also conducted several field tests to verify the correct operation of the developed wireless sensor node under different environmental conditions.

1. Range test (outdoor) – this test examined the maximum line of sight transmission range of the RF radio module. The achieved range was twenty kilometers.
2. Range test (residential) – this test examined the maximum transmission range of the RF radio module in residential areas where there are buildings and other obstacles. The achieved range was seven hundred and fifty meters.
3. Range test (densely vegetated) – this test examined the maximum transmission range of the RF radio module in forested areas. The achieved range was a little over one kilometer.
4. Mesh network test– The main points of this test were to test the correct operation of the mesh protocol and conform the self healing ability of the network when some of the intermediate nodes ceased functioning due to power loss or other technical issues.

In summary, the developed wireless sensor platform serves as a reliable and feasible choice for environmental monitoring applications. We have adequately addressed the transmission range limitation of the existing wireless sensor network technology and have successfully tested the developed wireless sensor platform.

7.1 FUTURE WORK

In this section we will briefly review the future trend in wireless sensor networks and evaluate the options that will improve the performance and interoperability of the developed wireless sensor platform. There are three main design features that can be improved.

- Integration of a solar energy harvesting mechanism
- Four layer PCB design and use of smaller components to reduce hardware size
- Compatibility with TinyOS

For environmental monitoring applications, power supply is an important concern. This is mainly because nodes are far apart from each other and power consumption is relatively higher due to longer communication range. Given that in most cases solar energy is ready available for outdoor applications such as environmental monitoring, integration of a solar panel and rechargeable battery can eliminate the need for battery replacement and consequently improve the performance of the system as a whole. As shown in Figure 58, the setup is very simple. The current or energy from the solar panel is connected to a voltage regulator and a resistor to control the current and the level of voltage applied to the batteries.

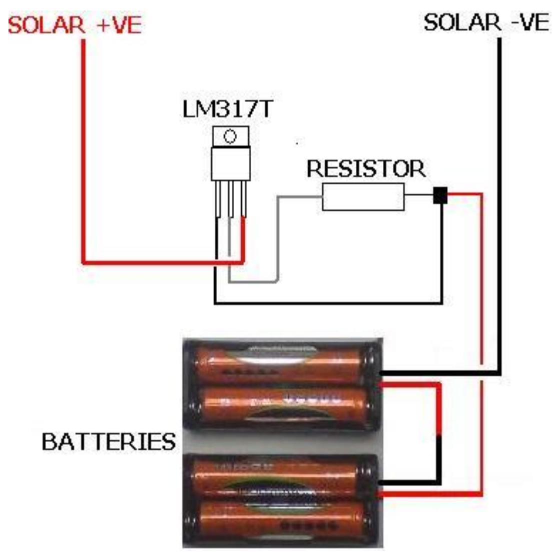


Figure 58. Simple Solar Panel Setup

In terms of PCB size, it can be significantly reduced if we used four layer PCB and smaller components. Four layer PCB means that there are four copper layers which means components can be placed closer to each other on the PCB, but four layer design is also a more expensive option. The PCB size can be further reduced if smaller components are used. For example TQFP packaging for microcontrollers is substantially smaller than the usual PDIP packaging.

The final design aspect which requires the most work is compatibility with popular wireless sensor network development platforms. By compatibility we mean the ability to write applications in popular development platform such as TinyOS as opposed to directly programming the microcontroller. There is nothing wrong with developing applications by directly working with the microcontroller but in terms of ease of application development, compatibility with TinyOS can be an attractive feature due to code reusability and support.

So far, we have seen that wireless sensor networks have become a popular research topic and are gradually finding their way into industry. There is however still a big gap between the research and the commercial availability of wireless sensor networks. One of the best solutions which can be viewed as continuation of our work would be to converge the developed wireless sensor network at the network layer to IPv6. This would attract lots of corporate interests because there is already a huge infrastructure for IP networking that not only spans the internet, but reaches into people's homes. It would also make things much easier for consumers to adopt because it's a familiar protocol. And finally, it would put an end to the eternal protocol wars among manufacturers. Transition to IPv6 would increase power consumption by a small factor due to extended overhead, but a few hours of extra battery life doesn't compare to instant access to billions of consumers.

BIBLIOGRAPHY

[1] K.Jamieson,H.Balakrishnan,R.Morris B.Chen, "Span: An Energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *ACM Wireless Networks Journal*, pp. 481-494, 2002.

[2] JA.Stankovic,C.Lu,T.Abdelzaher T.He, "SPEED: A stateless protocol for real-time communication in sensor networks," in *IEEE Computer Society*, Rhode Island, 2003, pp. 46-55.

[3] A.Abouzeid J.Ali, "Coverage by directional sensors in randomly deployed wireless sensor networks," *Journal of Combinatorial Optimization*, no. 11, pp. 21-41, 2006.

[4] N.Xu. (2003) University of Southern California. [Online]. http://enl.usc.edu/~ningxu/papers/survey.pdf

[5] S.Pratomo,I.Mita R.Wirawan, "Design of Low cost wireless sensor network-based environmental monitoring system for developing country," in *APCC*, 2008, pp. 1-5.

[6] H.Wang,W.Wang,S.Wu K.Hua, "Adaptive Data Compression in Wireless Body Sensor Networks," in *CSE*, 2010, pp. 1-5.

[7] K. Lorincz,M.Welsh,O. Marcillo,J.Johnson,M.Ruiz, and J. Lees G. Werner-Allen, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. 2, pp. 18-25, March 2006.

[8] A. Terzis, S. Ozer, R. Musaloiu-Elefteri,J. Cogan, S. Small, R. Burns, J. Gray, A. Szalay K.Szlavecz, "Life Under Your Feet: An End-to-End Soil Ecology Sensor Network," in *Proceedings of the Third Workshop on Embedded Networked Sensors*, Cambridge, 2006, pp. 51-55.

[9] A.Wood,Q. Cao,T.Sookoor,H. Liu, A.Srinivasan, Y.Wu, W.Kang, J.Stankovic ,D. Young, J.Porter L. Selavo, "LUSTER: wireless sensor network for environmental research," in *Proceedings of the 5th international conference on Embedded networked sensor system*, Sydney, 2007, pp. 103-116.

[10] A. Hasler,S. Gruber,C.Tschudin I.Talzi, "PermaSense: investigating permafrost with a WSN in the Swiss Alps," in *Proceedings of the 4th workshop on Embedded networked sensors*, New York, 2007.

[11] J. Polastre , A. Mainwaring , D.Culler R.Szewczyk, "Lessons From A Sensor Network Expedition," in *in Proceedings of the European Workshop on Wireless Sensor Networks*,

2004, pp. 307-322.

[12] R. Freeman, H. Kirk, B. Dean, M. Calsyn, A. Liers, A. Braendle, T. Guilford, J. Schiller T. Naumowicz, "Wireless Sensor Network for habitat monitoring on Skomer Island," in *in Proceedings of the International Workshop on Practical Issues in Building Sensor Network Applications*, 2010, pp. 882-889.

[13] J. Wang, C. Rizos, G. Baitch, D. Kinlyside Y. Li, "APPLICATION OF DGPS/INS INTEGRATION TECHNIQUE TO BUSHFIRE MONITORING," in *in Proceedings of SSC Spatial Intelligence, Innovation and Praxis*, Melbourne, 2005.

[14] A.McArthur R. Luke, "Bushfires in Austra," in *Dept. Of Primary Industry, Forestry and Timber Bureau. CSIRO Division of Forest Research*, Canberra, 1978, pp. 154-160.

[15] R. Sun, Y.Sun, S. Al-Sarawi L.Liu, "A smart bushfire monitoring and detection system using GSM technology," *International Journal of Computer Aided Engineering and Technology*, vol. 2, pp. 218-233, 2010.

[16] (2011, August) TelosB datasheet. [Online]. http://www.willow.co.uk/TelosB_Datasheet.pdf

[17] (2011, August) MICAz datasheet. [Online]. http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf

[18] (2011, August) Waspmote datasheet. [Online]. http://www.libelium.com/documentation/waspmote/waspmote-datasheet_eng.pdf

[19] W.Su,Y.Sankarasubramaniam I.Akyildiz, "A survey on sensor networks," *Communication Magazine, IEEE*, vol. 40, pp. 102-114, 2002.

[20] (2011, August) Atmega168 datasheet. [Online]. http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf

[21] (2011, August) Xbee-PRO datasheet. [Online]. http://ftp1.digi.com/support/documentation/90000976_C.pdf

[22] (2011, August) 9-Xtend datasheet. [Online]. http://www.digi.com/pdf/ds_xtendmodule.pdf

[23] (2011, August) RMX-232 datasheet. [Online]. http://www.embeddedcomms.com.au/download/rmx232_datasheet.pdf

[24] H.Singh, and A.Chhabra B.Sidhu, "Emerging Wireless Standards - WiFi, ZigBee and WiMax," *Applied Science, Engineering and Technology*, vol. 4, pp. 308-313, 2007.

[25] (2011, August) CC2420 datasheet. [Online].

http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf

[26] (2011, August) ZigBee Alliance. [Online]. http://www.zigbee.org/

[27] (2011, August) IEEE 802.16f-2005: IEEE Standard for Local and Fixed metropolitan area networks – Part 16: Air Interface for Fixed Broadband Wireless Access Systems. [Online]. http://standards.ieee.org/getieee802/download/802.16f-2005.pdf

[28] (2011, August) Digimesh protocol. [Online]. http://www.digi.com/technology/digimesh/

[29] (2011, August) HIH-4030 datasheet. [Online]. http://sensing.honeywell.com/index.cfm?ci_id=140301&la_id=1&pr_id=145616

[30] (2011, August) LM335 datasheet. [Online]. http://www.national.com/mpf/LM/LM335.html#Overview

[31] (2011, August) ExpressPCB. [Online]. http://www.expresspcb.com/

[32] (2011, August) AVR Studio 4. [Online]. http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

[33] (2011, August) GCC compiler. [Online]. http://gcc.gnu.org/

[34] (2011, August) USBAsp datasheet. [Online]. http://www.protostack.com/accessories/usbasp-avr-programmer

[35] (2011, August) IEEE 802.16f-2005: IEEE Standard for Local and Fixed metropolitan area networks – Part 16: Air Interface for Fixed Broadband Wireless Access Systems.

[36] (2011, December) X-CTU User's Guide. [Online]. ftp://ftp1.digi.com/support/documentation/90001003_a.pdf

APPENDIX A

```
#include "ECUMote.h"

static int uart_putchar(char c, FILE *stream);
static FILE mystdout = FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_WRITE);
static int uart_putchar(char c, FILE *stream)
{
   if (c == '\n')
  {
    uart_putchar('\r', stream);
  }
  loop_until_bit_is_set(UCSR0A, UDRE0);
  UDR0 = c;
  return 0;
}
int main(void)
{
   uint16_t uncal_temp;
   uint16_t uncal_humid;
   uint16_t calib_temp;
   uint16_t calib_humid;
   InitADC();//initalize ADC
   USARTInit(MYUBRR);//initalize USART
   stdout = &mystdout;//tells the print function to print to USART
   while(1)
  {
    uncal_temp = ReadADC(2);
    uncal_humid= ReadADC(3);
    calib_temp = ((298/2.982)*((uncal_temp*5)/1023))-273;//calibrate temperature
    calib_humid= ((33*((uncal_humid*5)/1024))-32);//calibrate humidity

    if( calib_temp>TEMP_THRESH && calib_humid<HUMID_THRESH)// the value of
thresholds  depends on application
    {
      printf("warning!! temperature is  %d \n",calib_temp );
      printf("humidity is  %d \n",calib_humid );
      printf("at node %d \n",NODE_ID );
   }
    _delay_ms(100);// this delay is application dependant

 }
}
```

```
#ifndef _ECUMOTE_H_
#define _ECUMOTE_H_

#include <inttypes.h>
#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>

#define FALSE 0
#define TRUE 1
#define FOSC 1000000
#define BAUD 2400
#define MYUBRR FOSC/16/BAUD-1
#define SLA_W 0xA0
#define SLA_R 0xA1
#define NODE_ID 2// assign different node ID for evry node
#define TEMP_THRESH 34
#define HUMID_THRESH 50

uint8_t writeByte(uint8_t data, uint8_t address);
uint8_t readByte(uint8_t address);
void USARTInit(unsigned int ubrr_value);
void InitADC();
uint16_t ReadADC(uint8_t ch);

#endif
```

```
#include "ECUMote.h"
#include <avr/io.h>

void InitADC()
{
  ADMUX=(1<<REFS0);
  ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

}

uint16_t ReadADC(uint8_t ch)
{

  ch=ch&0b00000111;
  ADMUX|=ch;

  ADCSRA|=(1<<ADSC);

  while(!(ADCSRA&(1<<ADIF)));

  ADCSRA|=(1<<ADIF);

  return(ADC);

}
```

```
#include "ECUMote.h"


void USARTInit(unsigned int ubrr_value)
{
  UBRR0H=(unsigned char)(ubrr_value>>8);
  UBRR0L=(unsigned char)ubrr_value;


UCSR0A=(0<<RXC0)|(0<<TXC0)|(1<<UDRE0)|(0<<FE0)|(0<<DOR0)|(0<<UPE0)|(0<<U2X0)|(
0<<MPCM0);


UCSR0B=(1<<RXEN0)|(1<<TXEN0)|(0<<RXCIE0)|(0<<TXCIE0)|(0<<UDRIE0)|(0<<UCSZ02)|(0
<<RXB80)|(0<<TXB80);


UCSR0C=(0<<UMSEL01)|(0<<UMSEL00)|(0<<UPM01)|(0<<UPM00)|(0<<USBS0)|(1<<UCSZ
00)|(1<<UCSZ01)|(0<<UCPOL0) ;

}
```

```
#include "ECUMote.h"
#include <avr/io.h>
#include <util/twi.h>


uint8_t writeByte(uint8_t data, uint8_t address)
{


  TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN); // put start condition on the bus

  while (!(TWCR & (1<<TWINT))); //wait for TWINT flag set

  if((TWSR & 0xF8) == TW_START ) //check to see if the start condition has been put on the
bus
  {

  TWDR = SLA_W; //send SLA_W

  TWCR = (1<<TWINT) |(1<<TWEN); // clear TWINT flag set
  }

  while (!(TWCR & (1<<TWINT)));//wait for TWINT Flag set

  if ((TWSR & 0xF8) == TW_MT_SLA_ACK )//check to see ifthe acknowledgement has been
received
  {


    TWDR = address; // Send the address of EEPROM register we intend to write to
    TWCR = (1<<TWINT) |(1<<TWEN); // Clear TWINT flag set
  }

  while (!(TWCR & (1<<TWINT))); //Wait for TWINT flag set

  if ((TWSR & 0xF8) == TW_MT_DATA_ACK)//check to see ifthe acknowledgmnet has been
received
  {

    TWDR = data;
    TWCR = (1<<TWINT) |(1<<TWEN); // Clear TWINT flag set

  }
  while (!(TWCR & (1<<TWINT)));

  if ((TWSR & 0xF8) == TW_MT_DATA_ACK)//check to see ifthe acknowledgmnet has been
received
  {

    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);// put stop condition on the bus
```

```
 }

 return TRUE;

}

//This function reads one byte of data from EEPROM. Note that the adress refers to the internal
//register address of EEPROM, not the EEPROM address itself
uint8_t readByte(uint8_t address)
{

 TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN); // put start condition on the bus

 while (!(TWCR & (1<<TWINT))); //wait for TWINT flag set

 if((TWSR & 0xF8) == TW_START ) //check to see if the start condition has been put on the bus
 {

 TWDR = SLA_W; //load EEPROM address and Write bit

 TWCR = (1<<TWINT) |(1<<TWEN); // clear TWINT flag set
 }

 while (!(TWCR & (1<<TWINT)));//wait for TWINT Flag set

 if ((TWSR & 0xF8) == TW_MT_SLA_ACK )//check to see ifthe acknowledgmnet has been received
 {


   TWDR = address; // load the data
   TWCR = (1<<TWINT) |(1<<TWEN); // clear TWINT flag set
 }

 while (!(TWCR & (1<<TWINT))); //Wait for TWINT flag set

 if ((TWSR & 0xF8) == TW_MT_DATA_ACK)//check to see ifthe acknowledgmnet has been received
 {


   TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN); // put repeated startcondition on the bus

   while (!(TWCR & (1<<TWINT))); //wait for TWINT flag set

 if((TWSR & 0xF8) == TW_REP_START ) //check to see if the start condition has been put on the bus
 { // put repeated start on the bus
   TWDR = SLA_R;
```

```
    TWCR = (1<<TWINT) |(1<<TWEN);
  }
 }

  while (!(TWCR & (1<<TWINT)));
  if ((TWSR & 0xF8) == TW_MT_SLA_ACK )
  {

    return TWDR;
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);

  }
  return TRUE;

}
```