University of Southern Queensland

Faculty of Health, Engineering & Sciences

An Innovative Intelligent System for Child Care & Safety Applications using ZigBee Wireless Sensing Networks

A dissertation submitted by

Brad Goold

in fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Computer Systems Engineering (HON)

Submitted: October, 2014

This page has been intentionally left blank

Abstract

The focus of this paper is to outline the research undertaken of Wireless Sensor Networks and how they can be used to assist in home health and safety applications. The particular area of interest is a computer controlled system that tracks a wireless sensor attached to a child indoors for the purpose of alarming when the child enters a predetermined unsafe area. The system must also define boundaries and have the ability to alert a parent/guardian in the case of the boundaries being breached.

A thorough review of the available literature was undertaken which gave the author background information as to techniques for localisation and their suitability in indoor applications.

An empirical approach to localisation using the Profiling/Fingerprinting method was used. A centralised system design was implemented and an application called Adaptable Infant Monitoring System was written and tested to perform the localisation and boundary alarming and to provide the ability for remote location retrieval via storing relevant data in a database.

Algorithms were proposed to reduce the localisation error. These algorithms have been rigorously tested in two scenarios. A baseline test was applied to create a benchmark, then testing was applied to a real residential situation. The results are analysed and discussed.

The results concluded that localisation using the specified hardware and the fingerprinting technique was inadequate for the application domain. Factors attributing to the localisation error have been discussed including irregular radio signal propogation which has been earmarked for further work.

The localisation accuracy does not render the research work invalid. Providing an omni-directional radio signal and additional research and testing will see the full potential of the system design.

The system proposed in this research allows for user-defined boundaries to be set for the alarming mechanism. This feature alone provides, versatility and expandability which is one of a kind and has potiential to enter the market-place covering unlimited application scenarios.

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Brad Goold

0050085400

Signature

Date

Acknowledgments

Firstly I would like to thank my supervisor, Hong Zhou for her guidance and support throughout this project.

I must also mention those who helped keep me on track and gave assistance from their own disciplines: Martin Stager, Warrick Velt, and Sebastian Binnewies.

Most importantly, I would like to thank my wife, Carina, for her incredible support and patience with me throughout the whole degree. And a big thanks to my parents who believed in me.

Brad Goold

University of Southern Queensland

October 2014

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

AoA	Angle of Arrival
ADC	Analogue to Digital Converter
API	Application Programming Interface
APS	Ad-hoc Positioning System
BAN	Body Area Network
CAGR	Compounded Annual Growth Rate
CTP	Collection Tree Protocol
CSV	Comma Separated Values
DOF	Degrees of Freedom
DV	Distance Vector
EMI	Electro-Magnetic Interference
ETX	Expected Transmissions
GPS	Global Positioning System
HW	Hardware
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronic Engineers
IPTS	Indoor Personnel Tracking System
LEMON	Location Estimation by Mining Oversampled Neighbourhoods
LoS	Line of Sight
LQI	Link Quality Indication
LTS	Long Term Support
MAC	Media Access Control
MEMS	Micro-Electro-Mechanical-Systems
MIT	Massachusetts Institute of Technology

ML	Maximum Likelihood
MMSE	Minimum Mean Square Error
NAT	Network Address Translation
NLoS	Non-Line of Sight
OEM	Original Equipment Manufacturer
OS	Operating System
PoA	Phase of Arrival
PC	Personal Computer
PDM	Probability Density Map
PDR	Pedestrian Dead Reckoning
PRR	Packet Reception Rate
RF	Radio Frequency
RFID	Radio Frequency Identification
RSSI	Received Signal Strength Indicator
SW	Software
ToA	Time of Arrival
TMP	Time Management Plan
TDoA	Time Difference of Arrival
TX	Transmit
WAF	Wall Attenuation Factor
WSN	Wireless Sensing Network

Chapter 1

Introduction

Wireless Sensing Networks are fast-evolving into an era where the Internet Of Things is the "Hot Topic" for research and information gathering systems. Extensive research has already been undertaken into localisation and object tracking in indoor applications, but there is little in the way of complete solutions for home health and safety. This project aims to research current solutions and localisation approaches and aims to produce an out of the box solution for in-home localisation for home health and safety.

The motivation for the project emerged with the startling statistics of the amount of children below 5 dying in backyard swimming pools. The Royal Life Saving National Drowning Report stated that 19 children under 5 drowned in swimming pools alone in the 2012/13 financial year. In addition, 4 more children died from drowning in spas in the same period of time (Lifesaving Society Australia 2013). This raises the question as to whether the emerging WSN technology can be used to assist in reducing these incidents.

For completeness, a Market Research was undertaken in the preliminary stages of the project to discover the strengths and weaknesses of the WSN market . The compounded annual growth rate of WSN's in 2011 was at 42% (Shalini 2012), and market confidence shows interest in this evolving technology. Market projections for wearable devices which include WSN nodes also strengthen this confidence as shown in Figure 1.2. Torres-Solis, Falk & Chau (2010) state that there is "immense" opportunity in implementing an indoor localisation scheme, particularly in the area of providing navigational aids to Topographical Disorientation affected individuals.

It was found during an analysis of the literature that the area of using WSN's for child monitoring and health and safety has not been widely researched or applied in practice. This opens an area whereby a possible new application for pervasive computing can be applied.

1.1 Project Aims

This project aims to research, design and implement an out-of-the-box solution for personnel localisation in homes or residential buildings. It is proposed that we deliver a solution whereby the user requires minimal set-up time and technical knowledge in order to successfully interact with the WSN in the home.

The basis for the selection of this topic is to propose an *open source* and *cost effective* solution to localisation for indoor/residential applications in the interest of increasing the standards of living in and around the home, particularly in the application area of *home health and safety.*

In addition to the swimming pool deaths previously stated, one child is injured to some extent in the driveway every week by a motor vehicle (Eccleston 2014)[1]. With that comes the opportunity for using pervasive computing to help solve health and safety issues or enhance the quality of life for people in and around their homes.

Examples of uses for such an application are as follows:

- Child Monitoring—Anyone who has children will know that a toddler loves to explore, and this may mean that they can go out of sight in the blink of an eye. If predetermined "limits" were set out, a parent could be reassured that the child is within those limits if no alarm was to be raised [2]
- Patient Monitoring—Six out of ten patients who suffer from dementia or related illnesses can wander even in familiar places (www.alz.org 2014). This could put them in a situation whereby they could be injured or injure other people.

[1]The extent of the injuries are reported to be from minor to fatal

[2]The author in no means portrays that it is acceptable for the parent leave the child unsupervised. There are many variables that must be considered before the proper implementation of such an application, namely the confirmation of the child wearing the device.

Preliminary technology-based research was undertaken into the different wireless technologies — their strengths, weaknesses and opportunities. It was declared that the sensors that comply to IEEE802.15.4 would be the most suited to the application because of the low power and low cost of the devices. Amongst the considerations were — WIFI, Bluetooth and RFID. The main factors against the common infrastructure item, WIFI, was the bandwidth usage and power usage. Bluetooth only has a sensing range of 10m and is very high in power usage and cost (Chu, Wang, Liang, Ouyang, Cai & Chen 2011). RFID has a sensing range of only 1-2m and the cost of an RFID reader is relatively high (Liu, Darabi, Banerjee & Liu 2007). Furthermore, the university supplied Crossbow Telosb motes are IEEE802.15.4 compliant, low power, low cost and can be easily adapted/programmed using the TinyOS operating system which is designed for such devices[3]. Tafa (n.d.), who wrote "Sensor Networks in Pervasive Health-care Computing", states that "this operating system is very appropriate for most of the WSN applications including those used in health-care".

1.2 Objectives, Parameters and Constraints

The major objective of this project is to design and implement an effective indoor localisation system whereby a wireless sensor is able to be effectively tracked within a WSN. The term "localisation" in relation to WSN's can be defined as finding the location of a sensor's physical position in a deployment area efficiently (Abdelsalam & Olariu 2009*b*).

The localisation data is routed to a local computer/router where it is used to calculate an estimated position of the target and perform higher level tasks based upon a set of pre-defined rules.

An example of such rules could be that a certain area is not to be breached or an audible alarm will sound in conjunction with a call to the guardian's mobile phone. Figure 1.1 shows an illustration of the basic proposed system.

A practical viewpoint was taken when defining the parameters of this project which includes the fact that indoor localisation must tackle the expectation of obstacles in the localisation area. Krishna & Doja (2011) state that building walls, reflecting surfaces,

[3]see section 2.5.2 for more details on IEE802.15.4

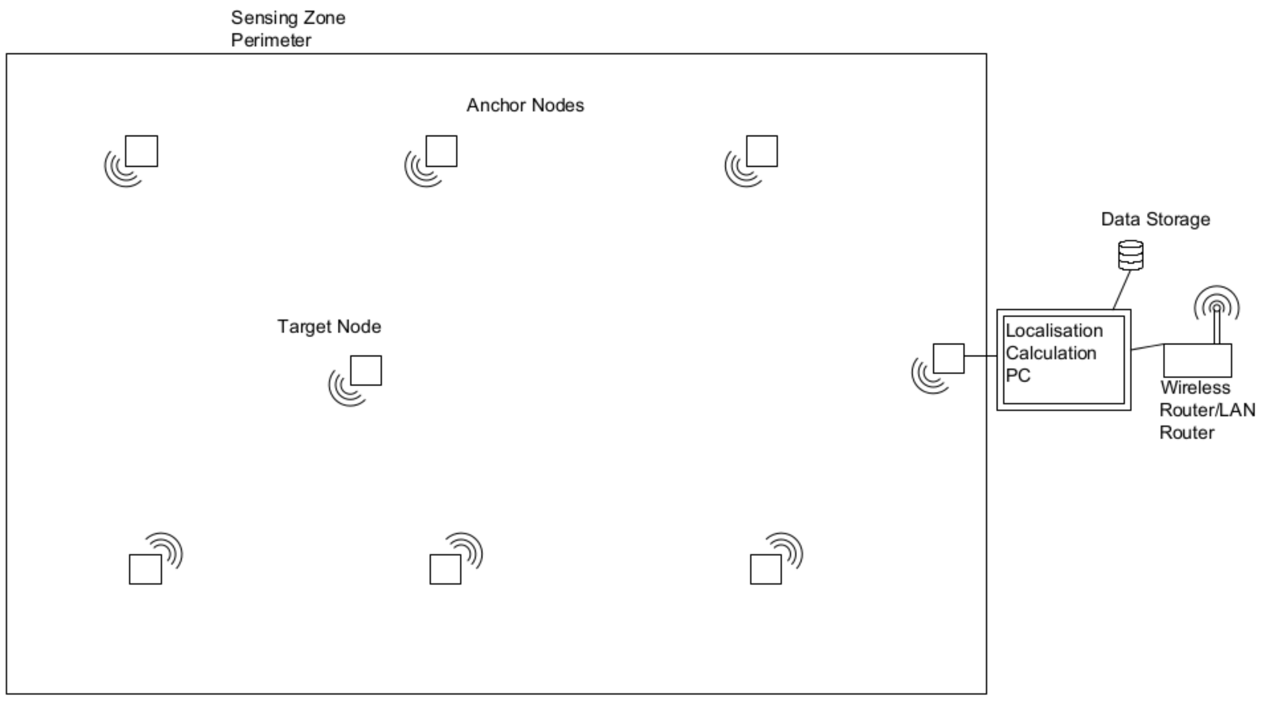


Figure 1.1: Sketch of proposed system outline

objects (transparent and opaque), and room shape are factors that must be considered when localising indoors. Another factor that must be considered is the latency of localisation. The system must be able to localise a moving object in real time. These variables are evaluated during testing and are considered to be the most important metrics as a real-life scenario is the aim of the project.

Certain assumptions and constraints were derived so that the project was able to be completed and tested with the supplied resources and within the supplied time frame. These constraints may be relaxed or removed as an opportunity for future work. These parameters indicate the particular situational, environmental and technological basis for which the design is to consider as the high-level inputs to the system:

2D localisation only — 3D localisation systems require more time spent on application writing as well as the fact that the added dimension opens up more opportunity for localisation error.

Use of Telosb Crossbow motes — The author has limited prior knowledge into WSN motes, so a research of a “standard” mote was chosen as the preferred platform. Furthermore, as research progressed, more evidence concreted the use of crossbow motes, namely their adherence to IEEE802.15.4 defining low power, low cost devices.

Programmed in nesC with TinyOS — Preliminary research and tutorials into TinyOS and nesC has formed the basis for using the Crossbow motes and TinyOS.

Open Source — For contribution to the research community, an open source software system is to be adhered to.

Low Cost — Cost is to be kept at a minimum without compromising the functionality of the system. The estimated budget for a complete system should be less than $300.

Time Constraint —Research, design, test and implement in 37 weeks. The deadline for this project to be complete and tested is 24th October 2014. This enables the author time to proof-read and print the final documentation of the project which has a due date of 30th October 2014.

1.3 Market Research

This section aims to describe the condition of the current Wireless Sensing Network market. It's not an exhaustive Market Research document, but rather an indication to the reader as to the motivations behind this project[4]. The WSN market is obscure in the fact that there are two tiers in the way we see Wireless Sensors. These being the "WSN solutions" and the "devices" themselves. This difference is outlined in the sections 1.3.1 and 1.3.3 but in general, we can consider the tiers as markets for end-users or developers/designers.

As this project is based on "indoor" WSN's, the research in respect to end users is focused on "smart homes" and WSN solutions for residential areas[5].

The IEEE802.15.4 and ZigBee standard has a large market share in smart metering,

[4]A full market research would have taken too much time away from the design and implementation of the project, although an idea of the shape of the market is important

[5]At times the line between industrial solutions and in-home solutions is grey in their application, and this overlap is also considered

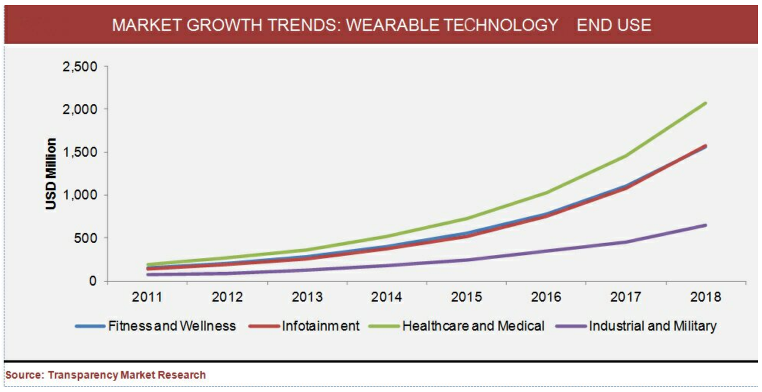


Figure 1.2: Market Projection of Wearable Devices (*Wearable Technology Market - Global Scenario, Trends, Industry Analysis, Size, Share And Forecast 2012 - 2018* 2013)

energy management and resource management (Hatler, Gurganious & Chi 2014). There is also strength in the ZigBee market in regards to consumer confidence and preferences with most people regarding ZigBee as “familiar” or “most familiar”. One negative aspect of using a sensor in the 2.4GHz frequency band, (such as ZigBee) is that the frequency band is “crowded” and suffers “limited range”. Renesas are contributing to the development of an international sub-GHz standard, but it has not yet become available (Mametani n.d.). We believe it would be too pre-emptive to adopt an un-released standard for the proposal of a new solution.

Because of the interoperability of ZigBee sensors, it would prove favourable to continue with market trends and momentum. The worldwide trend in revenue of WSN’s is rising and expected to rise as outlined in this chapter. Also, the amount of companies offering out-of-the-box solutions for WSN localisation is very small. We consider this an opportunity for this project to enter the market place in a competitive and comfortable state.

1.3.1 Market Definitions

Indoor localisation techniques using WSN’s can be purchased from specialised WSN firms that create a total solution, although this market is not well described or advertised. This was evident when researching specific companies offering solutions to home owners. Total solutions for homes are directed more towards smart energy, security and home-automation.

As for a developer who would potentially offer a solution such as the one proposed by this project, the technical aspect of the WSN market is very large and well defined. The market leaders on Wireless sensor technology are Renesas, Microsemi, Silicon Laboratories, Linear Technology / Dust Networks, Texas Instruments / National Semi, Lord, Semtech, Analog Devices, Millennial Net. Although companies such as Renesas provide the lower level IC's to intermediate firms who design and assemble the sensors.

1.3.2 Market Size

Globally, the market for wireless sensor devices was \$532 million in 2010 and \$790 million in 2011 with a CAGR (Compounded Annual Growth Rate) of 46%. (Shalini 2012). At the current standpoint in early 2014, 9 billion devices are connected to the Internet (www.researchmoz.us 2014). With the growth rate increasing each year, we can predict that the market is accepting the IoT as a part of everyday life and we can consider it as sense of confidence for pursuing a project in this area.

1.3.3 Market Segmentation

As previously stated, the market is split into two main tiers. On one side, there is the portion of the market who purchase the end product or the solution. For example, the home owner will purchase a WSN energy saving package from a company who supplies, installs and commissions the sensors. The other side of the market is based on the developers and designers of WSN's and the portion of the market who create OEM systems for specific purposes.

For the purposes of this project, we need to consider both of these tiers as we will be striving to develop a device that uses the "technical" aspects (such as sensors, controllers, base-stations etc). We must also consider the current market as it stands. For example if the market is saturated with solutions similar to the one proposed, then we may find the project infeasible.

1.3.4 Market Trends

According to www.researchmoz.us (2014) the amount of devices connected to the internet is set to rise to 100 billion by 2020. Focusing in on our area of research, WSN revenues from smaller buildings will increase 60% faster than larger buildings over the next five years. In 2017, global WSN revenues from buildings under 50,000 square feet will reach $1.3 billion up from $127 million in 2012" (Hatler, Gurganious & Chi 2013). Although the definition of a small building is relatively large, one can get an indication of the market direction and momentum of WSN's in smaller buildings.

"there will be 50 million smart homes and buildings enabled by wireless sensor network (WSN) technology", according to global technology research firm ON World.

The do-it-yourself (DIY) model is preferred or most preferred by half of the respondents compared with 41% that would prefer to purchase from a cable/broadband service provider. ZigBees brand recognition continues to grow as 45% more consumers ranked it as Familiar or Most Familiar in 2013 compared with ON Worlds previous survey in 2012.

Between 2012 and 2018, ZigBee and IEEE802.15.4 based systems will make up 43% of the wireless chip-sets used for smart metering, home and building energy management, distributed energy resources as well as lighting controls.

1.3.5 Market Forecasts

As previously stated, the future of the IEEE802.15.4 market remains strong and continues to grow. As the trend has not started to plateau, one can expect strong continuous growth. Hatler et al. (2014) describes the future in WSN devices with the following statement:

> "In fact, revenues from ZigBee/15.4 devices and associated equipment will reach $50 billion (up from previous years) within the next few years" (Hatler et al. 2014).

1.3.6 Market Research Summary

The state of the WSN market can be summarised as a market that achieves strong growth and a successful short-medium term future. In addition to the state of the market, it can be concluded that from the information gained from a product search and comparison (available on request), the devices that are currently available in the market for retail sale are costly and somewhat sparsely defined in terms of power, localisibility and adaptability. In regards to development products such as the Ember by Silicon Labs, the price for a SW license and development kit is expensive and ongoing for the developer which proves unsuitable for our purposes as a Research Project.

1.4 Overview of the Dissertation

The rest of this dissertation is organised as follows:

Chapter 2 is an overview of the background information that encompasses all facets of this project. Included are the current systems in place, an insight into the technology and an introduction to to the communications protocol and TinyOS/NesC programming structure.

Chapter 3 is a comprehensive review of the literature surrounding indoor localisation techniques and the application to health and safety applications. This review was used as the foundation for further research into the child health and safety monitoring system.

Chapter 4 defines the the theoretical concepts developed in this project in order to adhere to the project objectives. Algorithms are defined in this section in addition to further information into the Collection Tree Routing Protocol.

Chapter 5 introduces the AIMS system. This acronym stands for Adaptable Infant Monitoring System. The section details the design steps including hardware, software and system design and gives finer details of the functionality of the system.

Chapter 6 defines the evaluation and testing procedures and displays the results of the tests including further testing of re-design stages.

Chapter 7 concludes the project with an evaluation of the objectives and how they were met. Further work and research opportunities are also discussed.

1.5 Chapter Summary

The main focus points delivered in Chapter 1 are centred around using a Wireless Sensor Network to assist in reducing the number of childrens lives lost from drowning in swimming pools. The aims of the project are to research and deliver a versatile and user friendly WSN system which can detect the position of a child in a sensor network and alarm when the child breaches set boundaries.

A market research was undertaken which showed that the state of the WSN market is strong and consumer confidence in the technology is high. Actual WSN devices were researched for price and usability, but it was concluded that the university supplied TelosB wireless sensor be utilised because of the availability and adherence to the low-cost, low power IEEE802.15.4 network protocol.

Additional constraints were defined including a maximum cost of $300 for the total system package deliverable to the home. The system is to be programmed and made open source using TinyOS and NesC. Localisation will be in 2 dimensions, and the time frame for the project completion is 37 weeks.

Chapter 2

Technology and Tools

2.1 Chapter Overview

In order to have an objective view on the scope of the proposed system, the existing technology in child safety monitoring was researched. This research included papers, products and patents and a critical analysis of the solutions was made. Furthermore, the application areas and scenarios for the proposal are defined in conjuction with introductions to the tools required to meet the objectives. These tools give the reader the essential background understanding of the building blocks of the proposed system.

2.2 Current Technology in Child Safety Monitoring

Pool monitoring devices have been designed and patented since the 1970's all over the world. Amongst those are wave monitoring devices (Codina 1972) & (Millen 1983); wearable submersion devices that wirelessly transmit when the device is submerged (Quinones 1996) & (Flood 1998); Swimmer motion and behaviour tracking devices such as the system designed by Miller, Halwachs & Farstad (1999) where they use electromagnetic pulses to track and transmit the swimmers behaviour and position; video monitoring devices that track swimmer movements and locations (Lu & Tan 2004) & (Xiaoyang, Wenkai & Fei 2007).

Wu, Cai, Huo, Wu & Zhou (2013) designed a wearable drowning monitor whereby the micro-controller system is worn by the child and when the child is detected to be at a certain depth and movement is in a window of set parameters , then an alarm is raised. This design is aimed for use at crowded swimming pools in china where supervision is ‘difficult’. It is acknowledged that in this circumstance, the design has merit and is a viable solution for children who can already swim but may have sustained an injury causing unconsciousness in the water.

Taking into account the merits of the aforementioned designs, it is clear that these systems are only valid for discrete locations or swimming pools alone. The research direction and proposed solution outlined in this document is highly versatile whereby the user can define the areas that they deem as unsafe and it doesn’t necessarily need to be a swimming pool. The unsafe area may be a bathroom, garage, or driveway etc. This versatility along with the low cost will be more saleable in the growing WSN marketplace.

Furthermore, the use of a system where the boundaries are defined by the user means that the alarm can be raised BEFORE the child enters the water. This saves precious time in an emergency situation.

2.3 Application Areas

This project is aimed primarily at the domestic market whereby the system can be used in a normal home. A person who is deemed to be at risk in some way would have the target sensor attached to them. The areas in which the person may move around in are recorded into the systems computer prior to attachment of the target sensor. Unsafe areas are defined by the user to the software in order to set out boundaries for alarming purposes. Once the person breached the boundaries, the alarm would be raised to the user. The image in Figure 2.1 shows a typical application scenario whereby the person who is being tracked is allowed inside the house, but not outside. There are 2 physical exits whereby the wearer of the sensor would be able to enter the forbidden zone marked in yellow.

The application domain does not restrict itself to a particular scene such as a swimming pool or even just on child safety. The domain extends to situations such as Alzheimer’s patients or dementia patients who may get lost or enter an unsafe area where they may injure themselves or others. In addition, the system may be used to create an unsafe area around water tanks which is also a known risk for drowning amongst children (anon 2014).

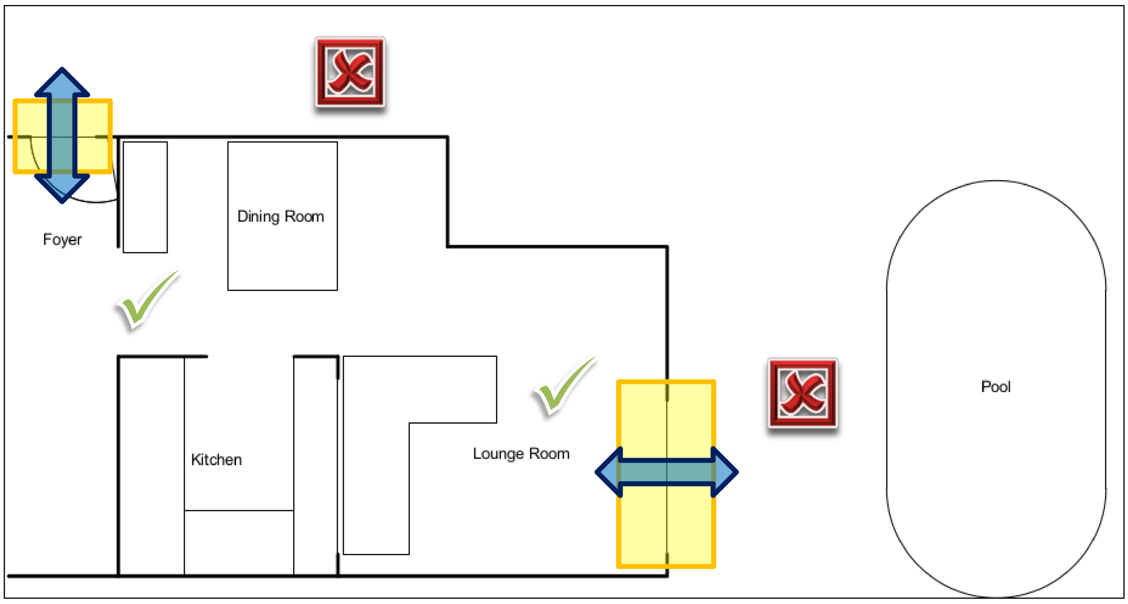


Figure 2.1: Layout depicting a typical application for use by the proposed system

2.4 Introduction to Wireless Sensor Networks

WSN's are defined as a network of small embedded systems with wireless connectivity via radio link that can perform tasks, gather information or route information to other sensors. Network types can vary in their formation, topology, communication protocol, routing protocol and communication range. Applications of wireless sensing networks range from monitoring environmental variables, activating remote devices, monitoring health of hospital patients (Purvis 2014), tracking assets and the location of objects/people, and even for security surveillance. The applications for wireless sensing networks is starting to evolve more as the price of purchasing and implementing a WSN falls to affordable levels.

Pervasive computing has fast become an accepted part of life over the last decade. More and more computers and computer controlled devices are used to perform tasks that we take for granted. Examples are automated toll gates on toll-roads and automated checkouts at the supermarket. Wireless sensing networks have been extended to the consumer market with Smart Metering devices and smart home automation controllers.

2.5 Crossbow TelosB Wireless Sensors

Telosb wireless sensors from Crossbow Inc have been specifically designed to enable cutting edge experimentation for the research community. The incorporation of the low power MSP430 MCU and CC2420 radio chip enables researchers to use the latest technology which adheres to the IEEE802.15.4 communications standard. In addition to this, the mote has on board temperature, humidity and light sensors as well as a USB connection for ease of programming and interface (Cro 2003). The **datasheet** for the TPR2420CA can be found in Appendix B.5.

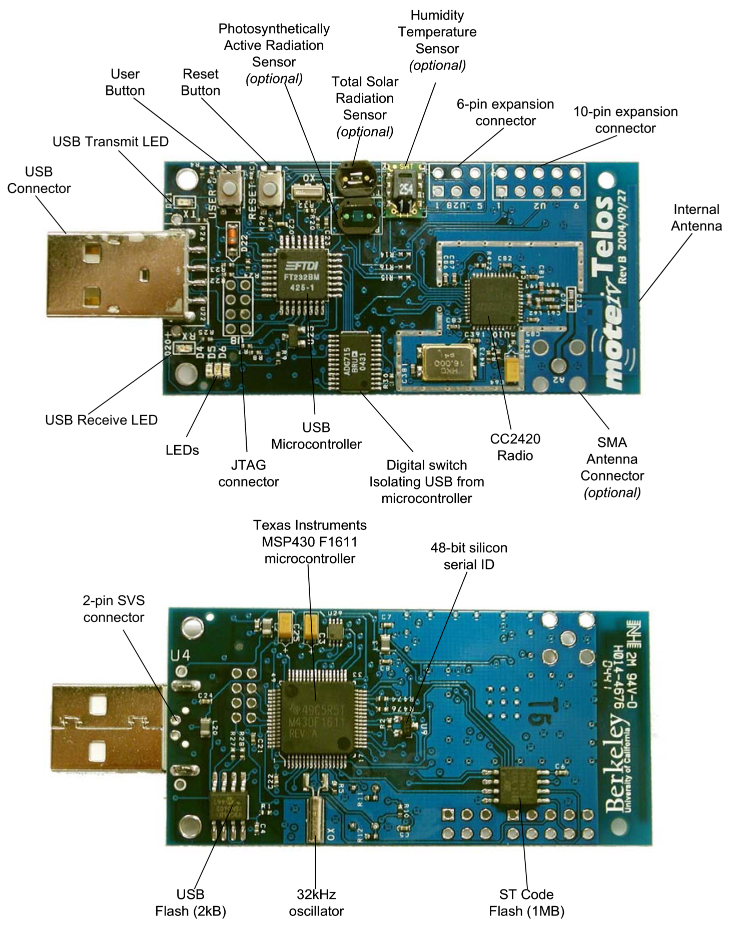

Photosynthetically Active Radiation Sensor *(optional)*
Humidity Temperature Sensor *(optional)*
User Button
Reset Button
Total Solar Radiation Sensor *(optional)*
6-pin expansion connector
10-pin expansion connector
USB Transmit LED
USB Connector
Internal Antenna
USB Receive LED
LEDs
JTAG connector
USB Microcontroller
Digital switch Isolating USB from microcontroller
CC2420 Radio
SMA Antenna Connector *(optional)*
Texas Instruments MSP430 F1611 microcontroller
48-bit silicon serial ID
2-pin SVS connector
USB Flash (2kB)
32kHz oscillator
ST Code Flash (1MB)

Figure 2.2: Crossbow TelosB Wireless Sensor (MoteIV 2004)

According to Gao, Zhou & Su (2011), the three major goals of the TelosB for experimentation are to consume minimal power, allow ease of use and increase the software/hardware robustness. The CC2420 radio chip allows for various transmit power settings (-25dBm -> 0dBm) as shown in Figure 2.3, which allows the developer to tune the transmission range for lower battery consumption (See B.4 for output current consumption). In addition, the utilisation of the MSP430's low power operation, sleep and active modes allows further reduction in power usage by the mote.

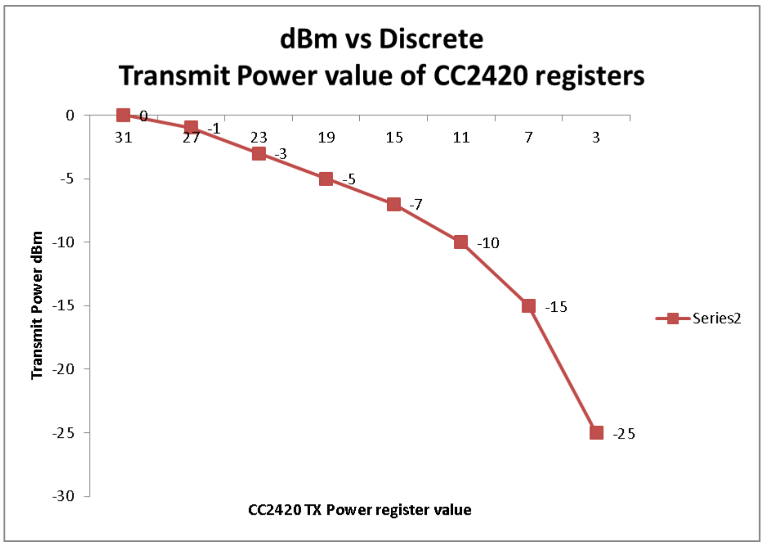


Figure 2.3: Transmission power reflected by the register values in the CC2420

2.5.1 Hardware components

The Crossbow telosb mote has been designed with additional functionality built into the hardware of the device. Included in this functionality is a temperature sensor, humidity sensor, total-solar radiation sensor and a photo synthetically-active radiation sensor (MoteIV 2004). The USB interface allows ease of connection and programming of the device in conjunction with the ability of using the USB for a power source. More importantly, the USB can act as the primary channel from the WSN to the outside world.

2.5.2 IEEE802.15.4 Communications

The TelosB mote, coupled with the CC2420 radio chip adheres to the 802.15.4 standard for LR-WPAN devices. The standard covers 3 main frequency ranges: 868MHz, 915MHz and 2.45GHz[1]. This standard defines the PHY and MAC layers and its main objectives are: ease of installation, reliable data transfer, low cost and extended battery life — whilst maintaining a simple, yet flexible protocol (IEEE 2011). These objectives are coupled with the intent for incorporating little or no further infrastructure.

The Physical Layer PHY

The physical layer provides the data transmission from the device. It is the layer that interacts directly with the RF transceiver providing services such as channel selection, energy management and signal management functions. 2.45GHz using the IEEE802.15.4 standard communicates through using Digital Sequence Spread Spectrum (DSSS). The CC2420 radio module incorporated in the TelosB handles the PHY later functionality in addition to many of the MAC functions described in 2.5.2.

The Media Access Control Layer MAC

TinyOS in conjunction with the CC2420 driver and CC2420 chip handle the MAC functionality (TinyOS 2013). The MAC layer enables the transmission of MAC frames and provides addressing and channel access mechanisms. Error protection is achieved by generating MAC frame checking sequences. The MAC layer also incorporates a control mechanism so that multiple communication mechanisms cannot access the radio chip simultaneously. This multiplexing allows the seamless operation of different network protocols operating over the same physical communication medium.

Access to the radio chip uses the CSMA/CA (Carrier Sense, Multiple Access with Collision Avoidance) control algorithm. CSMA/CA basically means that the device will listen on the network until there is no transmission heard. It keeps waiting until the channel is free and then sends a frame. It waits for an acknowledge from the receiving device before deeming success or failure. If the frame collides, the MAC functionality goes into a transmission delay called Binary Exponential Back-off whereby it waits a predetermined amount of time before starting the CSMA sequence again (IEEE 2011, pp21-23).

[1]The TelosB wireless sensors used for this project use the 2.45GHz band, and therefore 2.45GHz is the only frequency band discussed here

TinyOS provides this MAC functionality as an abstraction through the Active Message interface which is discussed in Section 2.5.3. Active Messages are directly accessible through the application layer.

2.5.3 Wireless Sensor Software Development (TinyOS and nesC)

Introduction

TinyOS is an operating system designed for low power embedded systems such as wireless sensors. It consists of a scheduler and drivers for a suite of different micro-controllers and hardware components. TinyOS is written using a programming language called 'nesC' which stands for 'Network Embedded Systems C' which is a component based and event driven (TinyOS 2013). The scheduler in TinyOS is very lightweight and robust. It schedules tasks in a round-robin technique whereby there is no concurrency and therefore it is favourable that short un-nested tasks are used.

The program structure of nesC varies significantly to a normal C program but the normal syntax of the code remains very similar. The nesC structure uses components and interfaces which are 'wired' together to form an application. According to Levis (2006), wiring is the most challenging concept about nesC for people new to the language. The following sections give the reader background information to the programming construct and calling mechanism of NesC. These sections aim to give a basis to understanding this wiring technique and how TinyOS can be used as a flexible and robust operating system for wireless devices.

Components and Interfaces

In nesC, Components are similar to objects in the fact that they encapsulate the functionality of a certain program element. A component in nesC has a "specification" and an "implementation". The specification defines the functions that the component provides and uses. These functions are usually grouped together in "interfaces" defined below. The implementation is the actual code that the component provides. For example if a component provides a function called open(), then the implementation for open must be coded in the implementation section for that component.

There are 2 types of Components: A Component Module provides the implementation of the interfaces that a component defines in it's specification. A Component Configuration is a way of linking different components together through their interfaces. This is discussed later.

Listing 2.1 shows an example of how component module is structured. The specification is defined in the "module" section where the interfaces that it uses and provides are defined. These used interfaces must be provided by a higher level component in order for the functionality of this component to be valid. The implementation section shows the actual callback for the Boot interface event called booted(). So when the booted event is raised, this code is called. In turn, the booted() function in this example then uses the SplitControl interface to start another lower level component. Note that the name of the interface can be changed by using the 'as' directive.

Listing 2.1: necC 'Component Module'

```
// ------------------------------------------------------------------------
module ComponentC(){
    uses {
        interface Boot;
        interface SplitControl as RadioControl;
    }
}
implementation {
    // Must define implementation of the booted event provided from the
    // boot interface
    event void Boot.booted() {

        // Call the start function provided by the SplitControl
        // interface.
        if (call RadioControl.start() != SUCCESS)
            fatal_problem();
    }

    // Must define implementation of startDone event
    event void RadioControl.startDone(error_t error) {
        if (error != SUCCESS)
            fatal_problem();
        // application code can go here..
        // ...
    }
}
// ------------------------------------------------------------------------
```

Interfaces - Collections of related functions that can be used by or provided to components. An example of a commonly used interface is the standard control interface:
The standard control interface defines 2 functions called start() and stop(). A component that "provides" this interface must therefore define the implementations for the functions. This means that there may be many implementations of start() and stop()

which are related to different components in the same application (Levis 2006, pp. 22-24). A set of “used” interfaces outlines to the user the interfaces it needs to perform its job. The functions defined in the interface have two directives called a “command” or an “event”. A command is to be defined by the component that provides the interface and the “event” is to be defined by the component that uses that interface. An example of the SplitControl interface is shown in Listing 2.2.

Listing 2.2: nesC ‘interface’

```
//
   ↪ ↪ ------------------------------------------------------------------------
interface SplitControl {
    command error_t start(); //implementation defined by the
        ↪ component
                             //that this interface is wired to.
    event start_done(error_t); //implementation defined by calling
        ↪ component
    command error_t stop();
}
//
   ↪ ↪ ------------------------------------------------------------------------
```

The block diagram in Figure 2.4 gives a visual representation of how components and interfaces interact. Although it is noted by Levis (2006, pp. 34) that it is “dangerous to signal events from commands”, the intention of the drawing is to give the reader an intuition of the calling structure of the code without introducing complexities.

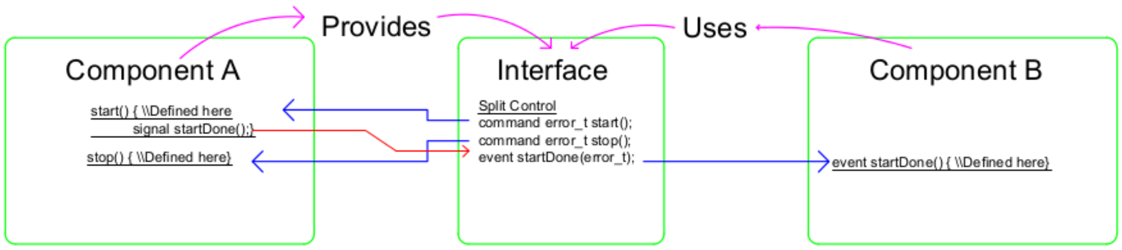


Figure 2.4: Component A provides the interface for use by component B

The second type of Component is called a ‘configuration’. A configuration is used to “wire” components together by connecting their interfaces. Every nesC application has at least one configuration that wires the components together (Gao et al. 2011). Configurations not only connect interfaces together, but they also can “export” interfaces (Levis 2006, pp. 54). Exporting means that an interface my not be required by a certain component, but can be used by a higher level component. Consider the case where the lower level application code receives the Received Signal Strength from the CC2420 radio chip. The higher level code may not require this information, but and interface that provides the functionality can be exported to the application level code for use by the application developer.

Component Configurations also have a specification and implementation section. The implementation section is used to define the actual wiring and components used. The specification is used to define interfaces that are provided or used for calling and caller components. The specification in the top level application component may be empty. An example application including a configuration is shown in Listing 2.3 which gives the reader a better intuition as to the wiring and structure. ComponentC from Listing 2.1 is also wired into this application.

Listing 2.3: nesC ‘Configuration’ and one of the Module files

```
 1 //--------------------------------------------------------------------
 2 //MyApp.nc
 3 //--------------------------------------------------------------------
 4 //Configuration
 5 configuration MyAppC { }
 6 implementation {
 7     components MainC, ComponentC, RadioC, SubC;
 8 
 9     ComponentC.Boot -> MainC; // MainC provides the Boot interface
10     ComponentC.SplitControl -> RadioC; // RadioC provides
           ↪ splitcontrol
11 
12     RadioC.subControl -> SubC; //SubC also provides a splitcontrol
13 }
14 //--------------------------------------------------------------------
15 //RadioC.nc
16 //--------------------------------------------------------------------
17 module RadioC {
18     provides interface SplitControl;
19     //...
20     uses interface SplitControl as subControl;
21 }
22 
23 implementation {
24 
25     command error_t SplitControl.start(){
26         // start device here
27         return SUCCESS;
28     }
29     //...
30     command error_t SplitControl.stop(){
31         // start device here
32         return SUCCESS;
33     }
34     //...
35 
36     task void foo() {
37         // ...
38         call subControl.start(); //calls the splitcontrol interface
               ↪ of
39                                  //the SubC component
40 
41         signal SplitControl.startDone(); //signals event to be run
               ↪ by
42                                          //the code in ComponentC
43         //...
44     }
45 
46     task void bar() {
47         //...
48         signal SplitControl.stopDone();
49     }
50 
51     event subControl.StartDone() {
52         // subcontrol has started
53         return SUCCESS;
54     }
55 }
```

Note that the specification for this configuration is empty (Line 5). This is the top level file and therefore the component need not provide or use any external interfaces. As mentioned, the implementation defines the components that are used before wiring them. The wiring occurs from line 9 with the use of ‘->’. The arrow points from the user to the provider. In the case of exporting an interface as previously mentioned, the '=' sign is used to signal that the 'user' doesn't actually use the interface but provides it to the next layer of abstraction. The wiring arrows can be reversed as long as the arrow points from the user to the provider.

An example calling structure would follow a simple routine such as:

1. ComponentC gets signalled by Boot.Booted() when the sensor boots up.
2. (Listing 2.1 line 15) ComponentC calls SplitControl.start() which, through the wiring, is wired to RadioC. Therefore RadioC.SplitControl.Start() (2.3 line 25) is invoked.
3. Hypothetically, RadioC does some stuff, and waits for a return event before posting the task foo() to the scheduler. When the scheduler runs foo() it calls subControl.Start() (2.3 line 38). As we can see by the configuration wiring (2.3 line 12), 'SubC' provides that interface, so we know that SubC will signal an event called subControl.StartDone() (2.3 line 51) when it has started.
4. After the return from subControl.Start() in the task foo() (2.3 line 38), the event SplitControl.startDone() is signalled (2.3 line 41)and therefore invokes the function which is coded in ComponentC (2.1 line 20) called

   event ComponentC.RadioControl.startDone().

As can be seen from following the simple calling structure above, the wiring and calling sequence can be quite hard to follow. This requires the application developer to gain experience in writing and learning the coding techniques in tinyOS and nesC before embarking on an application.

The TinyOS wiki from Stanford university [2] has a number of tutorials and code examples to help an application developer find their feet with nesC programming. It is recommended that a minimum of graduate level programming or experienced enthusiast is required to grasp the concepts of nesC development and programming (Levis 2006).

[2]http://tinyos.stanford.edu/tinyos-wiki/

Radio Communication

This subsection will give a brief outline on how messages are sent using the simple TinyOS message type called an Active Message. This is the lowest networking layer and is typically implemented over the sensor's radio. Active messages are single hop without a routing layer above performing some sort of routing protocol. The messages are given a type which is an 8 bit integer denoting the type of message in the packet (Gao et al. 2011).

The first thing that needs to be done to create a TinyOS application that sends a message via radio is to find which components are required to perform the task. The component called 'ActiveMessageC' provides the following interfaces:

```
configuration ActiveMessageC {
  provides {
    interface SplitControl;
    interface AMSend[am_id_t id];
    interface Receive[am_id_t id];
    interface Receive as Snoop[am_id_t id];
    interface Packet;
    interface AMPacket;
    interface PacketAcknowledgements;
    interface PacketTimeStamp<T32khz, uint32_t> as
       ↪ PacketTimeStamp32khz;
    interface PacketTimeStamp<TMilli, uint32_t> as
       ↪ PacketTimeStampMilli;
    interface LowPowerListening;
  }
}
```

Therefore one would guess that we we can write an application that uses these interfaces to send a packet via the radio. Unfortunately, this is **not recommended**. Following the tutorials on the TinyOS-wiki and spending some time trying to grasp the concept, one will find that we need to use the virtualised interfaces called AMSender and AMReceiver in conjunction with ActiveMessageC to send and receive packets. The reason for this is because AMSender provides a virtualized abstraction disallowing 2 components sharing the radio to interfere with each other. ActiveMessageC is only used for the SplitControl interface, i.e. turning on and off the radio.

```
generic configuration AMSenderC(am_id_t AMId) {
  provides {
    interface AMSend;
    interface Packet;
    interface AMPacket;
    interface PacketAcknowledgements as Acks;
  }
}
```

The AMSenderC component requires a parameter to be passed in. This is the Active Message Type. This type of 'generic' component actually creates a copy of the

AMSenderC code and replaces all uses of the parameter with the value passed in. In other words, when we create a 'new AMSenderC(10)' we create a new version of the AMSender module with active message type set to 10. All messages sent through this component will be sent with its type field set to 10.

From the above listing, we can see that the AMSenderC configuration provides us with Packet, AMPacket and AMSend. Now we can use these interfaces to send a message. The following code produces a simple calling sequence starting from the booted() event and ending up using a periodic timer to send messages over the radio with active message type set to 0x06.

Listing 2.4: Configuration for sending an AM via radio

```
//
    ↪ ------------------------------------------------------------------------
//MyApp.nc
//
    ↪ ------------------------------------------------------------------------
configuration MyApp{ }
implementation {
    components MainC;
    components ActiveMessageC;
    components new AMSenderC(0x06);
    components new TimerMilliC() as Timer;

    ComponentC.Packet -> AMSenderC;
    ComponentC.AMPacket -> AMSenderC;
    ComponentC.AMSend -> AMSenderC;
    ComponentC.SplitControl -> ActiveMessageC;
    ComponentC.Timer -> Timer;
}
//
    ↪ ------------------------------------------------------------------------
```

Listing 2.5 requires a few explanations. Firstly note that the payload data has to be requested from the 'Packet' module. This tells that component to return a pointer to the payload area of the specified size. The payload is wrapped in layers as traditional messages are when being transmitted. Also note that the data structure in message.h uses 'nx_struct'. This is called a 'platform independent' type and is required because of the differences in 'endianness' by different hardwares. For example when compiling for a MSP430, if this struct werent defined, the CC2420 would deem the data incompatible (Levis 2006, pp. 137-9).

Listing 2.5: Module for sending an AM via radio

```
//
    ↪ ------------------------------------------------------------------------
//ComponentC.nc
//
    ↪ ------------------------------------------------------------------------
#include "message.h"
module ComponentC() {
    uses interface SplitControl;
    uses interface Boot;
    uses interface AMPacket;
    uses interface Packet;
    uses interface AMSender;
    uses interface Timer<TMilli>;
}
implementation{

    bool busy = FALSE;

    event void Boot.booted() {
        call SplitControl.start(); //start the radio
        //go to sleep andd wait for the radio to start.
    }

    event void SplitControl.startDone(error_t error) {
        if (!error)
            call BTimer0.startPeriodic(TIMER_PERIOD_MILLI);
        else
            fatalproblem();
        //go to sleep and wait for timer to fire
    }


    event void Timer.fired() {

    if (!busy) {
        //get pointer to the payload
        Msg* msg = (Msg*)(call Packet.getPayload(&pkt, sizeof(Msg)))
           ↪ ;
        //set the payload
        msg->data1 = 0x5A5A;
        msg->data2 = 0xA5A5;

        //Send the packet
        if (call Send.send(AM_BROADCAST_ADDR,
                                    &pkt, sizeof(Msg)) == SUCCESS) {
            busy = TRUE;
        }
        //go to sleep and wait for sendDone()
    }

    event void Send.sendDone(message_t* msg, error_t error) {
        if (&pkt == msg) {
            busy = FALSE;
            //message has been sent!
        }
    }

//----------------------------------------------------------------------
//message.h
//----------------------------------------------------------------------
typedef nx_struct Msg {
  nx_uint16_t data1;
  nx_uint16_t data2;
} Msg;
```

Take note that when sendDone() is raised, the parameter passed is a pointer to the message sent. This is to ensure that the message type that was sent is the actual message that triggered this event. In the case of more components using the radio chip, it is necessary to check the message that was the required message for the triggered event.

Conclusion

TinyOS and nesC provides structured and robust platform in which to develop applications for the WSN domain. It has been shown that in order to create an application using TinyOS and nesC there is a fair level of understanding that is required. Once the developer has learn the concepts of the language and calling structure, the platform becomes very powerful and flexible. This section only provided a small introduction to a very large topic. It was intended that the reader understand how event based is implemented and how that can reduce the need for program loops which waste processor time and in turn battery power.

2.6 Interfacing a C application using the MySQL API

Using the 'MySQL C API' requires some background knowledge in the way connections are made to the databases and how data is returned. Creating queries is very similar to using the printf() function with the SQL query as the parameter. The following paragraphs will give a simple background on how a C program can use the MySQL API for simple tasks.

Firstly the MySQL development library must be downloaded, in addition with having the database software installed. The dev library is called "libmysqlclient-dev". After installation of the libraries, the API is accessed through the header file "mysql.h". Using the GCC compiler, the application files can be compiled and linked using "$(mysql_config –libs)". This will include the library and it's location.

To make a connection to a MySQL through a C function, the code requires the root username and password. This gives the user the privileges to create and drop databases, tables etc. There is an exhaustive amount of information about security and access rights, but they will be discussed no further here. To connect to the database, the following lines must be used:

```
MYSQL *conn = mysql_init(NULL);
if (!mysql_real_connect(conn, server, user, password, database, 0,
   ↪ NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
```

The parameters to the the mysql_real_connect() function, server, user, password and database are strings holding the relevant information. If the function returns NULL we have a problem connecting. The error from MySQL will be printed to stderr for debugging purposes. Once the connection has been made, the user can continue manipulating the database. The connection pointer is then used in the database access functions throughout that function call.

Queries are very simple, although the return from a query must be handled correctly to reach the required information. A simple query to select all information from a table is shown below [3].

Listing 2.6: MySQL API access to retrieve table contents

```
MYSQL_RES *result;
MYSQL_ROW row;
int numfields = 0;
int ii = 0;

MYSQL *conn = mysql_init(NULL);

if (!mysql_real_connect(conn, server, user, password, database, 0,
    ↪ NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
if (mysql_query(conn, "SELECT * from Main;"))
{
   fprintf(stderr, "%s\r\n", mysql_error(conn));
   mysql_close(conn);
   return 1; // or exit(1) if you are not handling error elsewhere.
}
result = mysql_use_result(conn);
num_fields = mysql_num_fields(result);

row = mysql_fetch_row(result);

while (row != NULL}
{
   // loop through each field in that row
   for (ii = 1; ii < num_fields; ii++)
   {
     // need to use only rows that have non-null values
     if (row[ii] != NULL)
     {
      printf("%2.4f, ", atof(row[ii]));
     }
     else
     {
       printf("NULL\r\n");

     }
     printf("\n\r");
   }
   row = mysql_fetch_row(result);
```

[3]The table is assumed to have all data-types as floats

```
43 }
44
45 mysql_free_result(res2);
```

First a connection is made to the database 2.6 Line 8. Then the query is sent (line 14). The result is then requested whereby it is further broken down into rows using the result returned from the API. The number of fields in each row is sought after to confirm the number of elements in the row array that is returned. Each iteration of the while loop takes another row from the result data. As shown, null values must be dealt with accordingly (line 37). It is also important to free the allocated memory for the result after the result has been used. The API allocates memory as large as the result, therefore passes the responsibility of the calling function to free the allocated memory when it so desires. This occurs at line 45 in this example.

The query string can be any standard MySQL query. This includes creating and dropping databases and tables etc. Information can also be retrieved from the table schema and other administrative information tables that reside inside MySQL. Inserting and deleting data is also accessed through the simple query function.

The API contains many functions that can help manipulate databases further. The information given here uses the relevant sections of Oracle (2014)'s MySQL C-API documentation as a guide and gives the reader a background as to how this project interfaces with the MySQL C API and how the data is manipulated.

2.7 Chapter Summary

Chapter 2 elaborated on the current state of the child safety devices that have been developed. It was found that although there are a number of patented devices, they all lack the ability to detect when the child has entered an unsafe area. They detect the fact that the child is in the water, or the state of the child's movements. It is noted that time is precious in life-saving situations and the earlier the alarm, the more chance of saving the life of the child if it enters an unsafe area.

The application domain for the proposed system is shown to have many applications, including monitoring of Alzheimer's patients or even monitoring children in regards to bath tubs or spas. As the systems boundaries are to be user-defined, the application scenarios are virtually limitless.

Background information into the WSN technology, and more specifically, the Crossbow TelosB mote was introduced. This included the hardware and communications protocol as well as TinyOS and NesC. The programming language and Operating System calling structure have been outlined for a thorough grounding on how the wireless sensors will fit into the system design.

Chapter 3

Review of Literature

3.1 Chapter Overview

A portion of this work has been aimed at completing a thorough survey of the literature outlining the *current* and *historical* state of wireless sensing localisation technology and it's applications in homes, buildings or residential areas. To narrow the research to the particular area of study, certain databases were targeted with selected keywords such as "indoor localisation" OR "Position tracking" AND "WSN"[1]. The results were then filtered down to capture the articles and studies that were published during the last 4 years. As this dissertation deals with the practicality of human-tracking, the results were further filtered into publications that applied (or could be applied) to tracking a human in the designated environment.

The resulting publications were then analysed for their specific metrics such as:

- Study type: Theoretical, empirical, comparative or survey
- The direction or 'school of thought' that the paper follows.
- The approach or methodologies taken to localise the person
- The validity of the data
- The references backing up their claims.

[1]Those databases included ACM, IEEE Xplore, Compendex and others

- Conclusions made and future works proposed.

Following on from each analysis references were traced back to the original source. This uncovered more information giving a thorough understanding of the roots of localisation in WSN's. This information was then used to create an understanding as to the application areas of this project and to possibly uncover areas of research which were vague or not yet embarked upon.

3.2 Wireless Sensing Network Indoor Localisation Techniques

Localising an object or person indoors has had a vast amount of attention over the last 10 - 20 years. Since GPS is ineffective indoors, the task of tracking an object indoors involves different techniques which can be broken into 2 major styles: **Range Free** and **Range Based**.

The range based technique uses range information which is provided when packets of data are sent and received, or utilise the angle of signal arrival (AoA) or the time of arrival (ToA). Another range technique calculates the position based on the signal phase of arrival (POA)(Liu et al. 2007). Range free uses techniques where the range is not required such as fingerprinting and DV-hop. Hybrid systems have also been implemented which exploit the benefits and suppress the disadvantages of both of these methods. This method is discussed in detail in Section 3.2.3.

Calculating the actual position can be performed using one of two broadly defined approaches. These are the **Distributive approach** and **Centralised approach**. The Distributive approach is when the node (or nodes) calculates the position based upon the information received from all other nodes. The Centralised approach is when all information is routed to a Base Station or a PC etc. which in turn calculates the position of the node (Nazir, Arshad, Shahid & Raza 2012).

3.2.1 Range Based Localisation

Range based localisation is where the target node's position is found by using range measurements such as received signal strength, angle, time, or phase to position the target node. The methods exploit these principles explicitly to calculate the location of the node. The following outlines the literature on the different techniques and the basic principles taken to achieve positioning.

Using RSSI as the location parameter is one of the most widely researched topics in regards to (indoor) localisation due to its availability in most modern radio chips (Schmid, Volker, Gadeke, Weber, Stork & Muller-Glaser 2010). Received Signal Strength Indication is the voltage received at a receivers signal strength indicator (RSSI) circuit (Wang, Chen, Lin, Chuang, Lai & Jiang 2012). It is the measured power of a received radio signal (Benkic, Malajner, Planinsic & Cucej 2008). The log normal shadowing model is the most commonly used propagation model which indicates that the average received signal strength decreases exponentially with distance:

$$P_r(d) = P_t(d_0) - 10\log\left(\frac{d}{d_0}\right) + x_\sigma \tag{3.1}$$

Where $P_r(d)$ is the RSSI power as a function of distance d. n is a path loss exponent which depends on the environment such as walls, obstacles etc. x_σ is a random variable which accounts for the random variation in path loss (Sahoo, Hwang & Lin 2008).

Most of the literature pointed out that the monetary cost of localisation was lowest when using RSSI, although Zhao & Smith (2013), who proposed an acoustic RFID based indoor system argues that the existing systems are "costly". Zhao & Smith (2013)'s approach with acoustics failed to indicate how to solve the problem of obstacles or NLoS localisation. Zhang, Xia, Yang, Yao & Zhao (2010) point out that "its (RSSI's) low cost does make it attractive for possible hybrid mobile terminal locating schemes" which are discussed further in Section 3.2.3.

RADAR (Bahl & Padmanabhan 2000) was one of the first RSSI based localisation systems proposed. They used 2 techniques, namely trilateration, which they define as 'triangulation' (shown in figure 3.1) – and an empirical approach using fingerprinting (defined in Section 3.2.2). The author's conclude an accuracy that RADAR produced

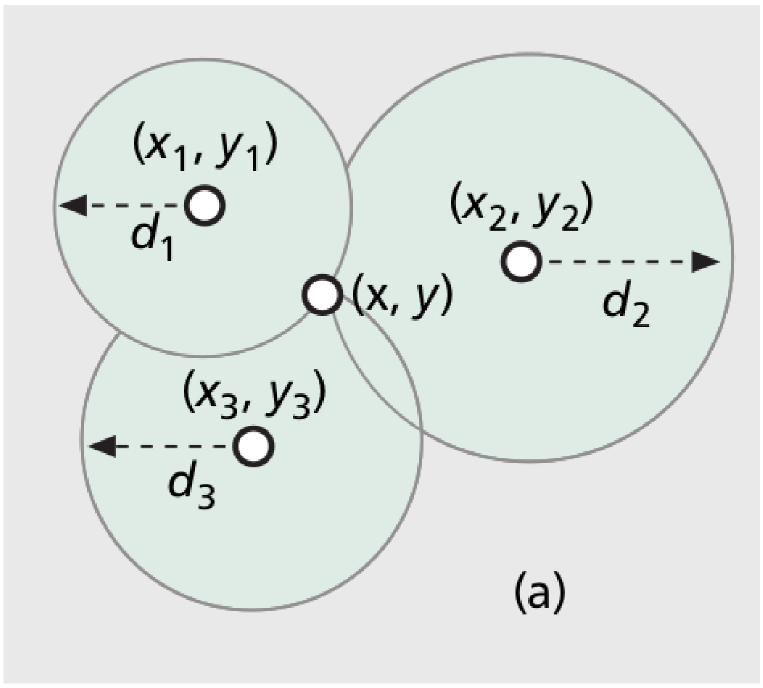


Figure 3.1: Trilateration technique showing signal propagation 'circles' (Boukerche et al. 2007).

was within 2-3m and with the high power of WIFI, they only required 3 beacon nodes per 1000m^2. Another pure RSSI based localisation scheme was proposed by Hightower, Vakili, Borriello & Want (2001). They called their scheme "SpotON" and the objective was to localise objects in office areas by calibrating each sensor to known distances and then letting the sensor search the calibrated list to find the mapped distance. Hightower et al. (2001) found that the results were somewhat inaccurate and concluded that there was opportunity for further work on localisation techniques. A dedicated empirical study on RSSI also confirmed these findings which was completed by Benkic et al. (2008).

Bahl & Padmanabhan (2000) was aware of the multipath, shadowing and attenuation effects of using RF indoors and adapted an algorithm which they called "Wall Attenuation Factor"(WAF) into their localisation calculations.

Most, if not all of the literature researched mention the difficulties of using an only-RSSI localisation solution. Zhao & Jia (2010) stated that RSSI may produce an error of as much as 50% and Haque, Nikolaidis & Gburzynski (2009) went further to mention that using only RSSI would be "futile". Technical electronic discrepancies also affect the RSSI read by a node even if the received power is the same, such as the bias on the reference voltage for the ADC(Zhong & He 2009). Benkic et al. (2008) completed an empirical study on RSSI measurements radio chips from 3 different vendors. They state that “There is a general belief that in Wireless sensor networks RSSI is a bad estimator of link quality. This belief was confirmed during our research”. On the contrary, Krishna & Doja (2011) state that RSSI is cost effective and suitable for short range applications, although they lack a definition of ‘short range’. Yedavalli, Krishnamachari & Venkatraman (2007) made an important point in saying that although averaging RSSI can improve the accuracy, the amount of beacon/anchor nodes (above 3) doesn’t necessarily have any effect. They also noted the obvious point that if we need to average the RSSI values, this takes up valuable time and can hinder the real-time accuracy when tracking a mobile target.

Giorgetti, Farley, Chikkappa, Ellis & Kaleas (2011), who proposed CORTINA, have an interesting approach where the beacon nodes continually monitor the RSSI of the neighbouring beacon nodes so they can keep track of the change in RSSI and channel quality. They also take multiple readings and calculate the mean RSSI. They use a system whereby the beacon nodes transmit the dummy RSSI signal to the target node (and also to each other).

Cortina also uses a 1 time setup where they use a RFID tag that’s located at the power outlet of the beacon nodes. The beacon then sends the RFID information back to the coordinator which then can tell the position of that beacon based upon the setup data. In addition, Cortina also uses barometric pressure to get a discrete 3D localisation estimation in which they can tell which floor the node is on. RSSI information is actually collected by the target node over a certain amount of time (4 seconds). Once this information has been collected, the information is packed in a couple of packets and sent back to the coordinator for dispatch to the main computer for localisation calculation. This saves on network throughput if each of the beacons send their info back over the network to the coordinator.

A major point to note about the use of CORTINA is, as stated above, the fact of mobility. If the information is collected over 4 seconds, the localisation system will be increasingly inaccurate and rough on faster moving targets.

There were conflicting views about signal propagation through a human body and the effects it has on localisation. Schmid et al. (2010) argue that their RSS means (averages) were relatively unchanged with reference to the position that the node was located on the body. Bahl & Padmanabhan (2000) have a more convincing argument, stating that if water effects the attenuation levels, and the human body is made of water, then it must also effect the attenuation to some degree. They found this was true in their results.

Angle of arrival (AoA) is a method where the angle of the received signal is measured by using rotating narrow-beam beacon nodes whereby the AoA is able to be calculated by the angle of the beacon when the signal was received at the target. This uses a technique called triangulation to calculate the distance based on the angles of 2 or more beacon nodes. Abdelsalam & Olariu (2009*b*), who used this technique claim an accuracy of 0.15 times the transmission range, therefore in regards to a CC2420 radio chip, this equates to approximately 4.5m. Unfortunately for AoA localisation, Line of Sight is required which is impractical in real world indoor localisation applications. This technique was not further researched.

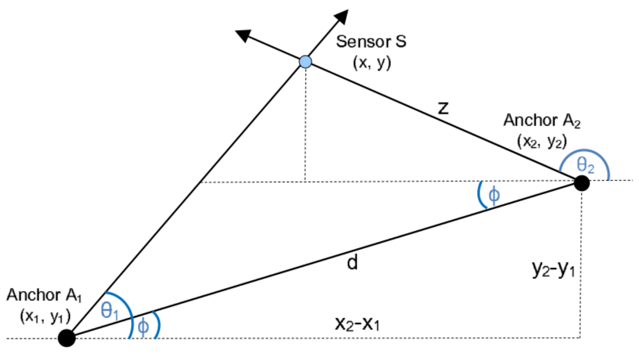


Figure 3.2: Triangulation technique showing angles from 2 beacons to the target node. (Abdelsalam & Olariu 2009*b*).

Time of Arrival (ToA) and Time delay of arrival (TDoA) exploit the radio frequency propagation speed in order to sense the distance to the target by measuring the delay or using precisely synchronised clocks (Zhang et al. 2010). ToA is the main principle of GPS. This method includes the use of specific hardware which adds additional cost to the project and cannot overcome the problem of NLoS range measurements. Mention must go to Zhao & Smith (2013) for their incredibly accurate acoustic RFID system which produced accuracies of less than 2cm with a latency of 700ms.

3.2.2 Range Free Localisation

Fingerprinting, otherwise known as Static Scene Analysis (Gwon, Jain & Kawahara 2004) is a 2 stage process whereby in the initial stage, the system must be 'profiled' or 'analysed' by the localisation software in order to record specific information about each location. This information can come from the RSSI, ToA, AoA etc and be used not for their range values, but in an attempt to 'fingerprint' the locations in a map of the the sensing area (Torres-Solis et al. 2010). The second stage is the localisation stage where the target sensor transmits the fingerprint information to the main computer (in a centralised approach). This information is matched in the database of stored values or a number of algorithms are performed to localised the node using MMSE or KNN (and many others) using this information.

RADAR (in addition to only RSSI) used the signal range as fingerprints to store in a database. RADAR used WIFI signals (802.1) and a centralised application written on a PC for storage and position calculation (Bahl & Padmanabhan 2000). Their accuracy was about 2-3m for a moving target using their technique, which somewhat confirms Glatz, Steger & Weiss (2010) who state that RSSI-fingerprinting turns out to result in quite noisy readings. We also must consider that RADAR uses WIFI signals which are higher power than IEEE802.15.4 low power signals.

A very similar approach to RADAR is LEMON, proposed by Haque et al. (2009). It stands for Location Estimation by Mining Over-sampled Neighbourhoods. Their method also involves using the RSSI for fingerprinting the position of the target node at a certain position. The profiling stage saves these fingerprints in a database. When the node is to be localised, the database is data-mined and the closest results are further interpolated to estimate the target's location. With 4 nodes in a 7mx7m area LEMON was able to accurately position to <2m. Another scheme using the exact same idea is the one proposed by Soleimanifar, Lu, Nikolaidis & Lee (2011) and they also report accuracies from 1 to 2m. Cherntanomwong & Suroso (2011) also use this approach and boast a mean error of less than 0.5m. These similar methods to the same approach report a range of error values which appears to be contradictary.

In 2008, Zhou, Jin, Zeng & Zhou (2008) proposed a localisation algorithm called Concentric Beacons Localisation which improves on the accuracy above. The basic approach is that 4 anchor nodes are located at the corners of a square 2D area. The anchor nodes transmit broadcasts at different power levels. The broadcast packet contains the TX power that the packet was sent. To estimate the position of the target, that node calculates the lowest signal strength from each different beacon and then computes the interpolation of the vertical and horizontal components. The intersection of the perpendiculars to each axis define the estimated position of the target node. The authors conclude an accuracy of 0.4 x the distance between transmission steps. If this estimation is accurate, the transmit power using the Crossbow Telosb mote will be able to provide an accuracy of less than 1/2 a meter.

Another system developed that also manipulated the transmit power was that provided by Angelopoulos, Filios, Karagiannis, Nikoletseas & Rolim (2012). In a different approach, they used statistical analysis of the received packets when transmitting with low transmit power. The anchors receive a transmitted packet and respond with a reply packet. The amount of successful packets received per "burst" of localisation packets is used to localise the node in comparison with the anchor nodes. The localisation is based upon the stochastic effect of RF propagation. Meaning that although the target may be in the transmission range, there is a possibility that the packet wont be received (due to external influences such as EMI, other RF, etc.). Also, if the target is outside the transmission range, there is also the possibility that the packet will get through. This enables the aspect of statistically analysing the "amount" of packets that are received in a certain burst to localise the target node. The MAC layer in the WSN protocol has the re-transmission number reduced to 0 so that the possibility of reception is not interfered with favourably (Angelopoulos et al. 2012). Although the author supplies good results, they fail to indicate to the reader how the results were measured and the method of data collection.

Bras, Pinho & Borges Carvaloh (2013) propose a scheme where sectorized pentagonal antenna is used as the signal transmitter. Packets sent to the node are numbered with the transmitted sector and the RSSI is sent back to the Localisation System computer for calculation based upon fingerprinting. It was concluded the localisation system they propose is marginally better than a 4 node RSSI fingerprinting method, although with the cost of WSN nodes reducing, the expectation that their technology will prevail seems unconfirmed.

A contextual approach was also developed whereby the location of the anchor nodes were specifically set out so that certain areas were monitored more heavily than others.

The localisation system called IPTS (Indoor Personnel Tracking System), proposed by Chu et al. (2011), would detect that the target had passed a certain way-point or door entry, then only take readings that were valid for that particular area. This empirical approach required refining their parameters during the testing phase. They used a HW/SW design around the ZigBee standard and boast an accuracy of around 0.4m but do not define the number of nodes or node density.

3.2.3 Hybrid Localisation

To overcome some of the shortcomings of the singular approaches, researchers and designers have been combining the different techniques. Ad-hoc Positioning System APS is a hybrid technique that combines the use of 3 GPS nodes for real-world localisation and DV-hop which uses the hop count back to a receiver node to calculate the distance. This type of localisation requires a dense internal network (Niculescu & Nath 2001). The objective of APS was to provide an absolute location for indoor localisation based upon the GPS locations derived from their respective sensors. This application area leans more towards an assisted GPS.

Colombo, Fontanelli, Macii & Palopoli (2011) used Pedestrian Dead Reckoning (PDR) from 9 DOF accelerometers and gyro's in combination with cameras and signal processing equipment to localise their target. Although their system provided an accuracy of less than 0.1m it would prove impractical because of the fact that most indoor environments have NLoS areas. They note that the accumulated error of PDR without reference is a problem for indoor applications.

Alternatively, Combining RSSI with other methods has improved the accuracy of localisation without necessarily increasing the cost. The method outlined by Coluccia & Ricciato (2010) which combines Maximum Likelihood (ML) estimation with RSSI improves the accuracy down to less than 2m. A combination that achieved higher accuracy was developed by Shen, Yu & Tan (2012) who used the 'Markov Random Field Model'. Simply put, they estimate what is a 'good' reading and apply their algorithm to it to get a more accurate position. Both of these approaches only provided theoretical results without including obstacles. On the other hand, Noguchi, Fukada, Mori, Sanada & Sato (2013) provided details of their empirical design, including obstacles and their results show high accuracy using RSSI and the kNN (k nearest neighbour) algorithm. The use of a Probability Density Map to guess the next position of the target node was proposed by Boom, Ros, De-Hosson & D'Souza (2012) although their accuracy is only 2.1m the complexity is quite low compared to other systems.

Another practical method which produced excellent results was proposed by Sugano, Kawazoe, Ohta & Murata (2006) who used Minimum Mean Squared Error (MMSE) and Maximim Likelihood Error (MLE) in conjunction with RSSI. Their empirical approach lead them to use these statistical methods in conjunction with raw RSSI values. They achieved an accuracy of 1.5-2m with a node density of 0.27 nodes/m^2. The appealing part of this distributed system is the simplicity and practicality of their approach.

Combining more aspects in a complex system, Schmid et al. (2010) used RSSI, GPS, PDR and Kalman Filters to localise an object in a building. The difference in their approach is that they used WIFI signals from infrastructure wireless routers. With this, they boast an accuracy of 3m without extra infrastructure cost. They fail to mention the added overhead of data sent over WIFI that restricts the user bandwidth.

A Mathematical Modelling approach was taken by Rudafshani & Datta (2007) using the Monte Carlo algorithm and a modified version of it. They claim to have the best results that they have discovered with localisation accuracy as low as 0.2m with a node density of approximately 0.002 (six nodes within the sensing range of the target node. The complexity of this system is somewhat daunting and the amount of theory required to understand and program this system is far beyond the time constraints of the project.

Cricket, proposed by MIT use the combination of Radio and ultrasonic signals to capture ToA and RSSI to localise the target node. They have a high accuracy (in the range of cm), but fail due to the inherent problem of NLoS human tracking (Abdelsalam & Olariu 2009*b*, pp 38).

DV-distance uses RSSI and the hop count back to the sink node to calculate a cumulative distance all distances are corrected with correction factor formula. An unknown node calculates it's position using trilateration when in the signal range of at least three beacon odes. (Niculescu & Nath 2003). In addition to this method, Zhao & Jia (2010) used the weighted centroid along with DV-distance in order bring down the localisation error associated with DV-distance only. They improved the results by more than 20%.

3.3 Localisation for Health and Safety Applications

There have been a number of designers and researchers that have produced solutions for health and safety applications (mainly in hospitals and aged care units). This section will outline some of the literature and derive it's relevance to this project.

OFlaherty & ODonoghue (2010) introduces a system called Vsense^{TM}, developed by Vital Technology Solutions whereby the patient wears a body area network (BAN) which monitors the vital signs of the patient and reports information back to a main database for evaluation. This system doesn't provide localisation information.

CodeBlue is a total solution for emergency response, localisation and vital-signs tracking of patients in hospitals (Lorincz, Malan, Fulford-Jones, Nawoj, Clavel, Shnayder, Mainland, Welsh & Moulton 2004). The system offers an integrated approach to Vsense's BAN incorporating a parcel of different solutions in one package as an attempt to provide an ubiquitous approach.

The Patient Centric Network (PCN) project is developing a common pervasive computing architecture for sensors in hospitals similar to the one outlined in CodeBlue. They argue that their focus is oriented towards the a integrated system without the need for proprietary devices and 'black-boxes' (nms.lcs.mit.edu n.d.). The website for PCN has a very sparse amount of information and it seems the program may have been abandoned.

An extensive and well written article on pervasive computing in health-care written by Tafa (n.d.) takes a more detailed approach at the topic of integrating sensor networks into a fully ubiquitous computing outfit. The author also studies the appropriate devices and connectivity of those devices. TinyOS is regarded as the choice Operating System because of its small size and event based operation.

Lu & Tan (2004) provided an interesting concept to help prevent drowning in swimming pools. Their vision based system monitors the activity of the swimmer and calculates their conditions based upon states derived in a state machine. This example of a vision based approach to monitoring arose many doubts in regards to video surveillance and its merits. The authors conclude that the system would provide many false alarms which were hard to rectify. Wu et al. (2013) had a similar approach although a wearable device was developed utilising accelerometers to track the behaviour of the person being monitored. They too reported many false alarms and suggested future work to the study of rectifying these alarms.

The aforementioned articles provide some insight into the systems that have been proposed. Yet they haven't touched on the application area in regards to location monitoring and alarming in specific circumstances. This opens an area whereby a possible new application for pervasive computing can be applied.

3.4 Information Logistics - Communication and Routing

Throughout the literature, a number of routing and upper-layer message protocols were mentioned, such as ZigBee, WirelessHART & 6LoWPAN, but there was no arguments in relation to the most effective for localisation (Giorgetti et al. 2011). Alternatively, there was mention in regards to the way that data was collected and processed which brought about the following definitions: **Distributive approach** and **Centralised approach**. Distributed is when the node (or anchor nodes) calculates the localisation result such as defined by Sugano et al. (2006). Centralised is when a Base Station or PC etc. calculates the result (Nazir et al. 2012).

Giorgetti et al. (2011) use centralised approach with collection tree protocol called JesNet. All of their information routes back over the JesNet protocol to a central computer for processing. There has been mention that this approach hinders traffic in the network and that a distributed approach is better (Sugano et al. 2006). On the other hand, the resources available on small WSN devices may also impact on the amount of data that can be stored for localisation if the WSN us used for multiple tasks.

Most of the literature didn't disclose the actual platform the software was written on, or the devices used. As the ZigBee protocol stack was inferred to be the choice protocol, further research was undertaken into the API, its complexity and availablility on Telosb motes. Many hours of searching and downloading prospective ZigBee stacks proved to be unsuccessful. Les Bowtel from USQ confirms the difficulty in getting an operational ZigBee stack for the TelosB mote as he embarked on the same search as the author. The following quote sums up this research:

> Many vendors offer us a ZigBee stack with their ZigBee solutions. No matter the vendor, working with a ZigBee stack still requires high programming skills and deep ZigBee theoretical knowledge. (Benkic et al. 2008)

Furthermore, the practicality of this project is considered more important than the theoretical understanding of an API for a routing protocol. The author and supervisor have concluded that the routing protocol will be chosen on its simplicity and effectiveness.

3.5 Chapter Summary

This review gave an insight into the indoor localisation techniques and methodologies that have been proposed or developed. The accuracies of some of the proposals was somewhat hard to gauge as the node densities, type, etc. were often not provided. Some of the localisation schemes required the use of large scale networks with a high number of beacon/anchor nodes which impedes on the "cost effectiveness" parameter of our project. Although some more accurate systems such as AoA and ToA have been defined, their limitations far outweigh their gains in accuracy in regards to NLoS solutions and additional cost of hardware. Hybrid systems utilising RSSI, Fingerprinting and statistical analysis prove to be the lowest cost in respect to the total system, although development time may inherently be an issue. The method of node clustering in order to improve accuracy around particular areas of interest will be explored as suggested by Werb & Lanzl (1998) who states that this approach "can be useful for tracking babies and Alzheimer patients". Taking the immense amount of information that is available on WSN localisation, the author believes a study into a system using RSSI fingerprinting and a combination of statistical approaches may prove the most effective and efficient given the time and constraints that the project is bound to.

Chapter 4

Localisation Protocol and Algorithm Design

4.1 Chapter Overview

This chapter provides more details on the theory behind localisation using profiling and the algorithms that were developed. The literature on profiling which was reviewed in the previous chapter contained contextual ideals and hence little information as to the finer details of the data storage and retrieval. More intense searching for details outlining the actual fingerprinting algorithms were unsuccessful also. The fingerprinting algorithm (below) was developed by the author based upon the concepts and information from the literature. Additional algorithms were produced to further reduce the error that was encountered during the initial testing phases.

4.2 Profiling/Fingerprinting Localisation

Profiling or Fingerprinting is a technique where signals such as RSSI are utilised to create a signature at locations throughout the working area. Static sensors are located at various positions throughout the area, but their individual positions aren't required for this technique. The target sensor periodically sends beacon signals to the static nodes. When the static nodes have received the RSSI information, it is routed towards the gateway for processing at the base-station PC. Each sensor's value creates a digital fingerprint of the actual location that the sensor is situated. Mathematically, this can be represented as follows:

$$P(x_n, y_n) = \{RSSI_{s1}, RSSI_{s2}, \ldots, RSSI_{sk}\} \qquad n = x_{max} \times y_{max} \tag{4.1}$$

where k is the number of static sensors and x and y are the physical locations defining position P.

```
Data: xSize ySize numSensors numSamples Packet PacketBuffer SensorID's
initialisation;
ii=0, jj=0, kk=0;
likePackets;
begin
    for ii := 1 to xSize do
        for jj := 1 to ySize do
            print "put the sensor at position ii,jj";
            for kk := 1 to numSamples do
                Packet = receivePacket();
                push packet onto packetBuffer;
                search packetBuffer for a suite of likePackets;
                save indexes of likePackets;
                if likePackets = TRUE then
                    store RSSI from like packets in DB table ii,jj,sensorID ;
                end
            end
        end
    end
    for ii := 1 to xSize do
        for jj := 1 to ySize do
            for kk := 1 to numSensors do
                calc Average of DB table ii,jj,SensorID[kk];
                store Average in Main Table (iijj,SensorID[kk]);
            end
        end
    end
end
```

Algorithm 1: Profiling Algorithm

The algorithm for fingerprinting is shown in Algorithm 1 which defines the major procedural functions of fingerprinting.

4.3 Centroid Algorithm

This project contributed to the research by deriving 2 more algorithms to further increase the accuracy of the MMSE algorithm. The first is the Centroid algorithm where the k closest matches from the database are used to form a polygon.

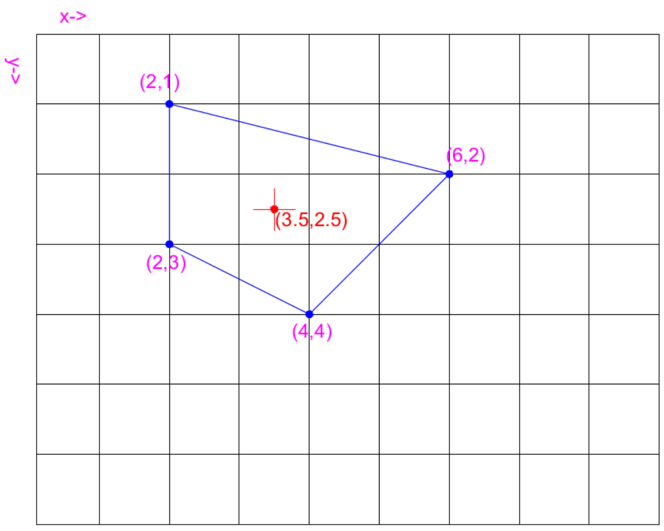


Figure 4.1: The Centroid of the 4 best database matches for localisation

The centroid of this polygon provides a more accurate localisation given that empirical results show that the best match is not necessarily the closest point to the actual position of the sensor. The centroid can be defined mathematically as follows:

$$C = \sum_{n=1}^{m} \frac{P_n}{m}$$

which minimises the squared distances between the points on the polygon. m is the number of points in the polygon, or more specifically, the number of best matches chosen from the database. P is the position at point n of m. A visual representation showing the 4 closest database matches and their centroid is shown in Figure 4.1.

4.4 Weighted Centroid Algorithm

The second algorithmic contribution is the Weighted Centroid. It follows from the Centroid Algorithm that the best matched point on the polygon is (calculated to be) the closest location to the position of the target sensor. Each successive point is a lesser match. If this is taken true, we can hypothesise that the position would be more weighted towards the best matched points on the polygon.

Starting from the best-matched point on the polygon, vectors are drawn to the lesser matched points. The endpoints of these vectors are then reduced towards the better matched point by a 'weighting factor'. The next step is to choose the second best match and do the same for each lesser matched point. The weighting factor is calculated by the differences between their error values. Once all points are moved, the centroid of the new points is found. Figure 4.2 shows a visual representation of the algorithmic steps. Note that each positional movement is denoted by the 'prime' notation such as A-> A' and B-> B'.

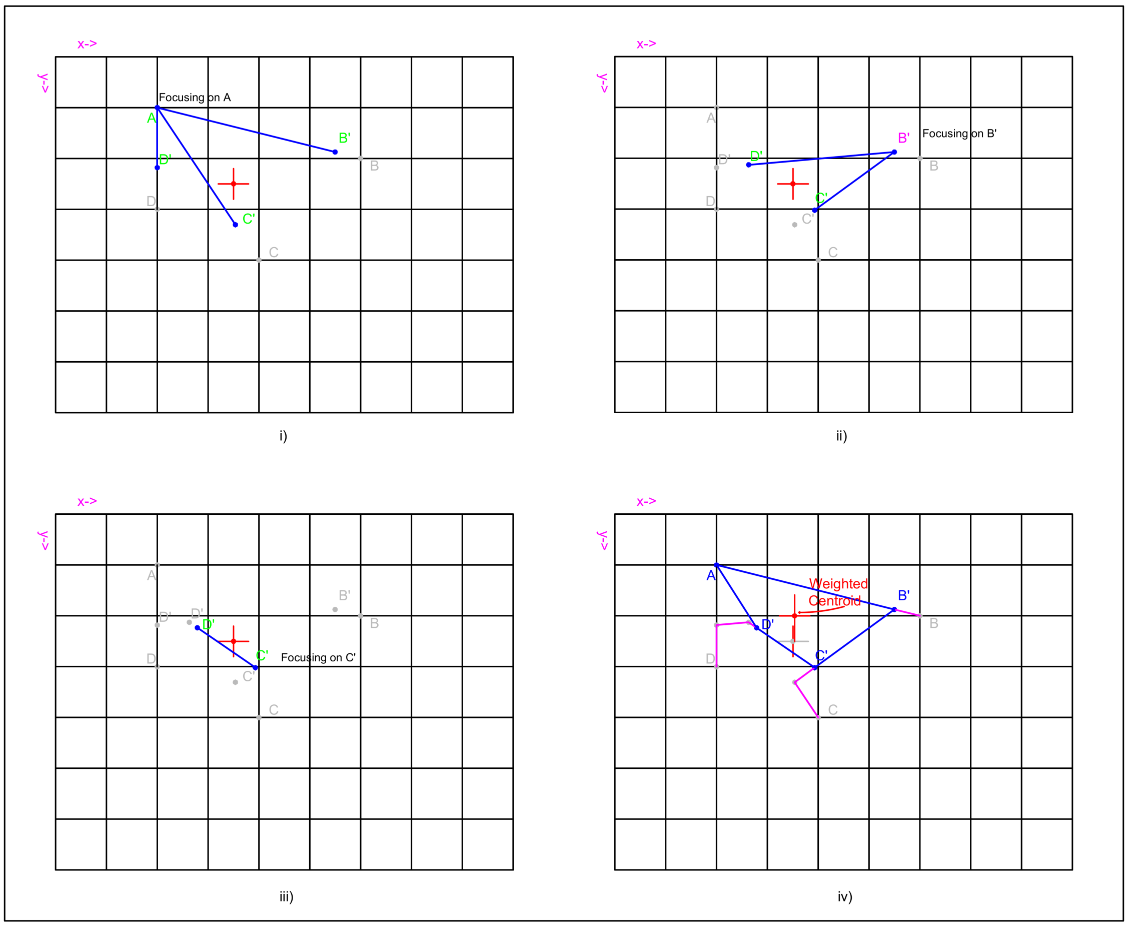


Figure 4.2: The Weighted Centroid of a 4 vertex polygon

The general form of the algorithm is as follows.

```
Data: Position(1...n), ii, jj, kk
Result: WeightedCentroid
initialisation;
ii=0, jj=0, kk=0;
n = number of positions in polygon;
begin
    for ii := 1 to (n-1) do
        for jj := ii to (n-1) do
            calculate length from Position(ii) to position(jj+1);
            calculate weighting factor;
            move position(jj+1) towards Position(ii) by weighting factor;
        end
    end
    for kk := 1 to n do
        sumx += Position.x;
        sumy += Position.y;
        WeightedCentroid.x = sumx/n;
        WeightedCentroid.y = sumy/n;
    end
end
```

Algorithm 2: Weighted Centroid Algorithm

4.5 Collection Tree Routing Protocol (CTP)

As previously stated in Section 3.4, the routing protocol adopted in this project is due to it's simplicity and effectiveness. The CTP routing protocol offers a simple routing protocol that delivers messages to the root node in a best effort style that proves to be very effective.

CTP is address free in the sense that nodes do not directly address each other to deliver a message. The process of delivering a message to the root node is by choosing the next hop by using a routing gradient. The routing gradient is created by using Link Quality Estimates which calculate the amount of uni-cast messages are acknowledged by nearby motes. The LQI is then used to calculate the Expected Transmission or ETX. The ETX is the metric used to form the basis of CTP (Fonseca, Gnawali, Jamieson, Kim, Levis

& Woo 2006).

Each node calculates the ETX to each of it's parents. The ETX is simply the sum of the expected transmissions to the parent plus the ETX that the parent has calculated. The key point is that the parent only keeps the ETX that is the lowest in the direction of a root node therefore creating a gradient of ETX's for which the CTP frameworks runs from.

Figure 4.3 shows and example network using integers for simplicity. The ETX for node 'H' would be set to 7 as this is the best ETX to the root node. Note that node 'F' the calculated ETX as 9 through node 'H' indicating that H's ETX is chosen to be 7.

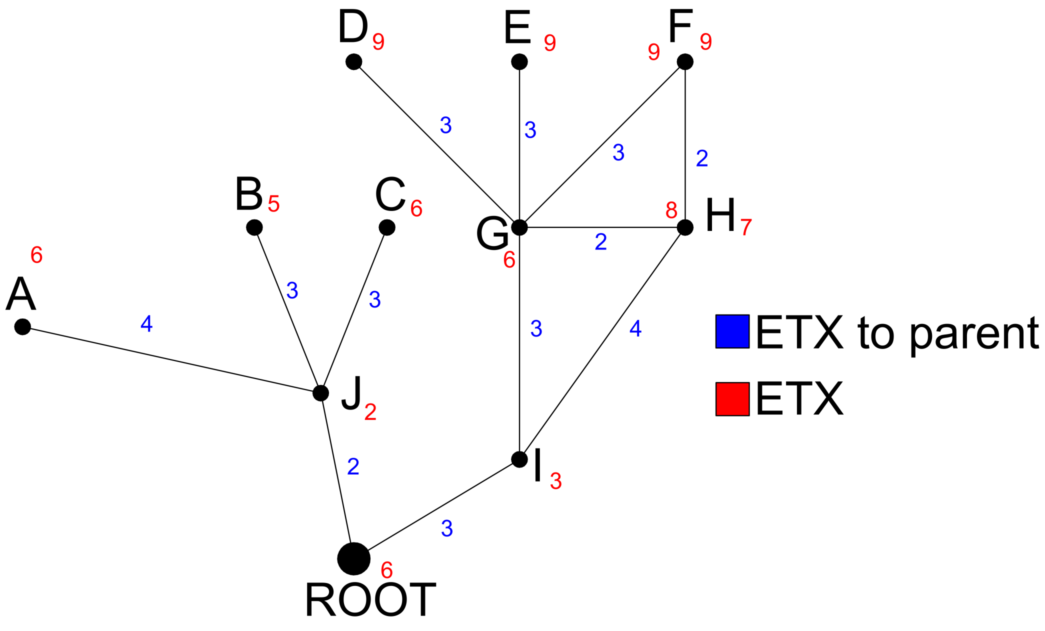


Figure 4.3: Example network showing the ETX gradient of nodes

4.6 Chapter Summary

This chapter aimed to introduce the techniques for communication routing and localisation. Utilising the information gathered during the Literature Review, the chosen localisation concepts were built upon and a formal protocol for the fingerprinting method was introduced.

Furthermore, 2 algorithms were derived to enhance the localisation accuracy of the system. The Centroid Algorithm and an enhancement called the Weighted Centroid Algorithm use the k best matches in the fingerprint database to narrow down on the centroid of the formed polygon. These algorithms are tested against the default Best Match algorithm during the evaluation and testing stage in Chapter 6.

Chapter 5

System Design and Implementation

5.1 Chapter Overview

This chapter introduces the software platform that has been developed to track the location of an infant in a WSN using the theoretical ideals set out in Chapter 4. The software is regarded as Adaptable Infant Monitoring System (AIMS) in this thesis. AIMS incorporates the software developed for the TelosB wireless sensors in conjunction with the software that resides on the central computer for localisation, storage and remote retrieval.

Section 5.2 is a description of the system architecture design of the development platform. 5.3 describes the hardware and how it is interconnected to the central computer. The central computer localisation software is described in Section 5.4. A breakdown of the sensor node software is given in Section 5.5.

5.2 System Architecture

In order to have a robust platform PC to program the TelosB motes using TinyOS and to provide a scenario for programming the localisation application, a Virtual Machine was chosen as the best option. This method was chosen for the fact that a vanilla install of a Linux operating system could be installed and that version is a Long Term Support (LTS). Ubuntu 12.04LTS was chosen as the choice system as it was recommended in the TinyOS tutorials for a base platform to build the tool-chains. To adhere with the project objectives, the code shall be open source. This enhances the use of a Linux virtual machine. Figure 5.1 gives an overview of the major system components.

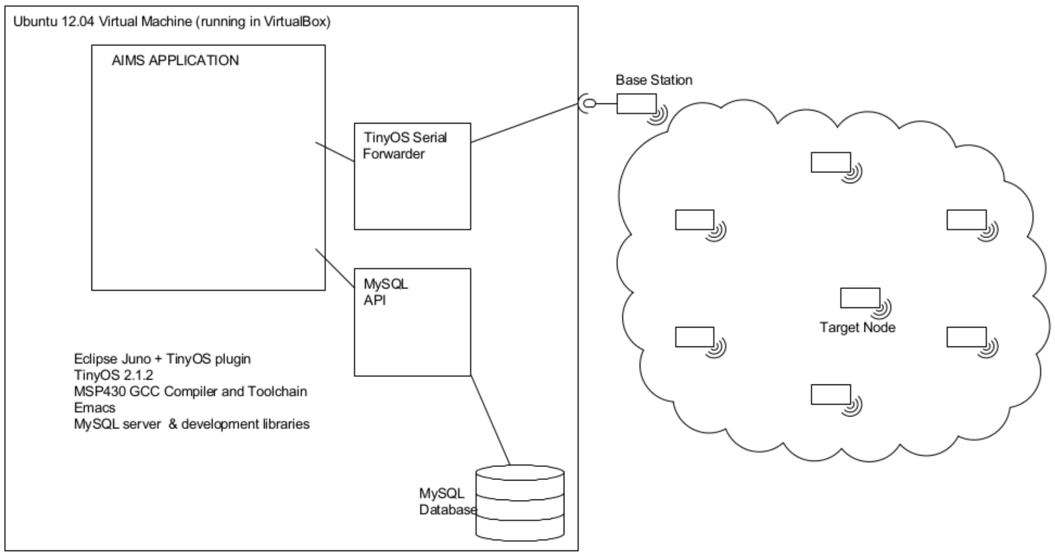

Ubuntu 12.04 Virtual Machine (running in VirtualBox)
AIMS APPLICATION
TinyOS Serial Forwarder
MySQL API
Eclipse Juno + TinyOS plugin
TinyOS 2.1.2
MSP430 GCC Compiler and Toolchain
Emacs
MySQL server & development libraries
MySQL Database
Base Station
Target Node

Figure 5.1: Overview of the system design from the VM to the WSN and Database

For the code to interact with the WSN, the TinyOS SDK (Software Development Kit) was used. This code is open source and was manipulated to adapt to the AIMS software.

MySQL server was installed in addition to the database system. The development files and libraries have also been installed to give access to the MySQL API.

Oracle's Virtual-box was used as the virtual machine driver and the Ubuntu VM was created on that machine. The network was set up using NAT so that the relevant drivers and updates could be installed as required. Emacs was used as the text editor for the C code and Eclipse was downloaded for developing the NesC code for the TinyOS platforms on the motes.

Each of the Mote's id's were filtered in the Virtual-Box hardware filter so that they would be passed directly to the virtual machine when they were plugged in. Furthermore, 'udev rules' were created to enable easier interaction with the sensors. Each of the serial numbers of the devices were translated in udev rules so that they corresponded to their mode id's which were labelled on the devices. This reduces confusion and the added hassle of knowing which device will be programmed by 'chance'.

NesC code syntax has a few dissimilarities from C so a plugin was downloaded to handle the syntax changes and additional keywords. The plugin also allows the user to follow through the calling structure to break apart components and their commands.

The tool-chain for compiling and downloading to the telosb motes was sourced and downloaded. Getting the correct tool-chain files proved more difficult than expected, so the author saved all files used to set up the machine in a folder and is accessible for further developers.

A full tutorial for setting up the virtual machine is located in Appendix D. Furthermore, the files required for executing the tutorial are available from the author.

5.3 System Hardware

The hardware design of this system has specific constraints as outlined in Chapter 1. The university-supplied TelosB motes have been selected be used for the design and implementation. TelosB motes have been designed specifically for the research community and for prototyping purposes. A saleable product will require the use of a device suited to the wearable market. The size of wireless sensors is reducing with the development of the technology. Sensium (2014) have produced a medical patch-like wireless sensor for a different application, but the physical size and flexibility of the sensor confirms the technology is available and possible to be adapted to the project.

To adhere to the project objectives, the system need only be scalable to 6 static sensor nodes plus an additional 1 target node and 1 base-station node. Taking this into account, the bandwidth of data also considers the network topology and the routing protocol used to convey the information to the central PC.

The distance between sensors is to be such that the area to be utilised by the system is sufficiently covered by no more than 6 static sensors to adhere with cost restraints. According to Cro (2003) the Crossbow TelosB mote has a transmission range of 20-30m indoors. Using the Collection Tree protocol, the AIMS Central PC can be located anywhere within 1 hop distance from any of the static sensors. The current system supports an area whereby the target sensor is to effectively transmit to ALL static nodes on each beacon signal. Therefore, if the localisation area is set out as a Square, the maximum distance between each of the corner nodes is $\approx 20m$.

The computer system running the AIMS Software must be Linux/UNIX based and be preinstalled with the MySQL Database. The system also must have a USB port for interfacing with the Gateway node and the WSN. In addition, the system must have the hardware to access the internet.

5.4 System Software

The central localisation software serves 3 main purposes. The first is to retrieve the packets of information from the WSN. The second is to perform the localisation function on the received data. And the final purpose is to store this information in a database so that a remote client can connect to the database and retrieve localisation results. Figure 5.2 is a basic outline of the AIMS localisation software. The user interface is text based but performs the required tasks efficiently and effectively.

The software comes as 1 executable called “aims” in addition to a number of source code files and a Makefile for compilation.

To run the software, call “aims” from the command line in the current directory or by using the system’s file manager.

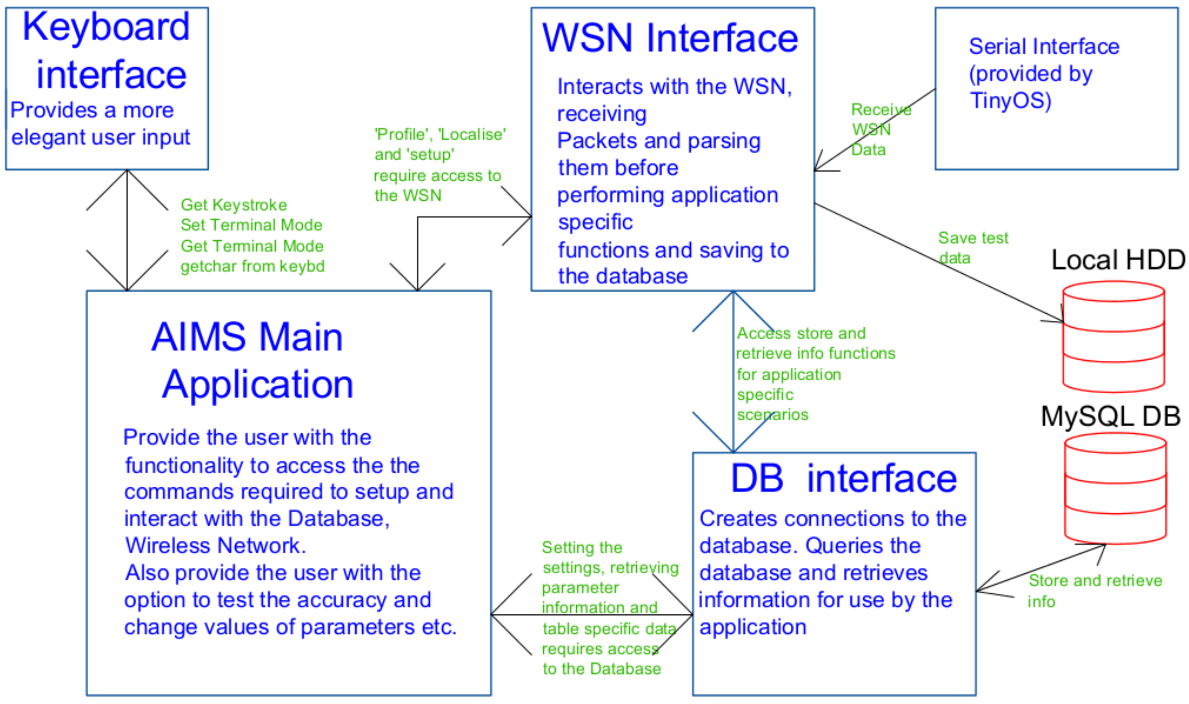

Keyboard interface
Provides a more elegant user input
WSN Interface
Interacts with the WSN, receiving Packets and parsing them before performing application specific functions and saving to the database
Serial Interface (provided by TinyOS)
Receive WSN Data
'Profile', 'Localise' and 'setup' require access to the WSN
Get Keystroke
Set Terminal Mode
Get Terminal Mode
getchar from keybd
Save test data
Local HDD
AIMS Main Application
Provide the user with the functionality to access the the commands required to setup and interact with the Database, Wireless Network.
Also provide the user with the option to test the accuracy and change values of parameters etc.
Access store and retrieve info functions for application specific scenarios
MySQL DB
DB interface
Creates connections to the database. Queries the database and retrieves information for use by the application
Setting the settings, retrieving parameter information and table specific data requires access to the Database
Store and retrieve info

Figure 5.2: Basic Software Design Model

5.4.1 User interface

The AIMS software that resides on the central PC runs a text-based user interface as a back-end. The functionality of the user interface is such that the user can interact with the software to perform monitoring, set-up, profiling, and even testing procedures. A basic menu offers the user a choice of options whereby they press a key to select that option. The interface has been enhanced by the use of raw terminal mode which enables the software to detect the keystrokes and act upon them without waiting for a newline such as what occurs by using stdio.h and scanf().

```
|==================================|
|Adaptable Infant Monitoring System|
|==================================|
     |=========|
     | A I M S |
     |=========|

Welcome! Please make your selection from the menu
|======|
| MENU |
|======|
1. Profiling
2. Monitor
3. Graph
4. Set up
5. Set Algorithm Parameters
```

```
6. Change number of samples for Monitoring
7. Test Accuracy
8. Set Boundaries
'q' to QUIT or go back
```

The following sections provide more detail as to the functionality of the menu options.

5.4.2 Profiling Function

The profiling function interacts with the WSN to store the individual fingerprint information into the database. The profiling function for AIMS is set out below. It is based on the concepts outlined by Haque et al. (2009), Cherntanomwong & Suroso (2011), and Soleimanifar et al. (2011) and follows Algorithm 1 in section 4.2.

Step 1 — The user selects the Profiling option from the AIMS menu system. AIMS then indicates to the user to place the target node at the first position in the grid. The Target Node is switched on and periodically sends out beacon signals in addition to a sequence number for packet synchronisation. The contents of the beacon frame are shown in Figure 5.3. The sequence number gets incremented by the target node on each transmit.

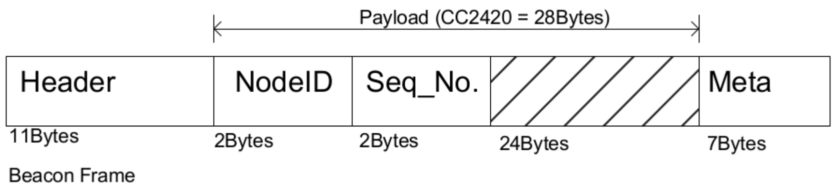


Figure 5.3: Format of the the Beacon Frame

Step 2 — The static sensors will receive this message and pack it in a different message type for transmission over the Collection Tree Protocol. The payload contents that are transmitted are shown in the Data Frame in Figure 5.4. The 'interval'

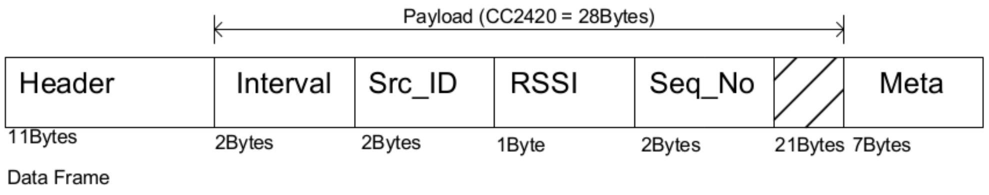


Figure 5.4: Format of the the Data Frame

is the time period of beacons that the target is sending. This is generally 100ms. The source id, is the id of the static sensor that actually received the beacon. The RSSI is calculated by the static sensor using the receive beacon packet. As mentioned, the sequence number is relayed for synchronisation of packets at the localisation PC.

Step 3 — Once a packet is received by the Root Node, it is queued for transfer via serial communications to the Localisation PC. The packets used for serial transfer change their packet structure once again. The packets received by the PC are of the following format.

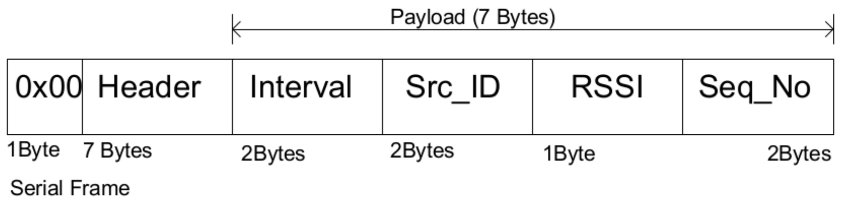


Figure 5.5: Serial Frame transferred from Base Station

Step 4 — The TinyOS Software Development Kit was used by AIMS to handle the reception of packets from the serial port. This SDK is distributed under the GNU General Public License. The library module used returns a pointer to the data that has been read and the length of the data are. The data contains the same format as Figure 5.5. The Localisation software dissects this packet and copies it's contents into a data structure holding the relevant information for use by the code.

Step 5 — The 'profiling()' function in AIMS clears the serial buffer and then reads in successive packets. These packets are checked for their content before storing the packets in a packet buffer. The buffer is shifted right on each successful read.

Step 6 — When a designated number of packets in the buffer have the same sequence number and independent source ID's the locations are then recorded. This 'suite' of packets denotes a single beacon that has reached the designated sensors.

Step 7 — The suite of packets' RSSI's are then stored in the MySQL database table corresponding to each sensor and its location.

Step 9 — Steps 5-8 are repeated for a set number of sample. For example, we may like to take 6 samples for each sensor at each location, so we take 6 full sequence sets, and save their RSSI's in the table corresponding to each position. Below is an example of a table for a specific sensor at a specific location.

```
+--------+------+
| sample | rssi |
+--------+------+
|    166 |  -31 |
|    167 |  -32 |
|    168 |  -31 |
|    169 |  -31 |
|    170 |  -32 |
+--------+------+
```

Step 10 — AIMS then indicates to the user to move the sensor to the next position. Steps 2-10 are repeated until the entire grid has been profiled.

Step 11 — Once the profiling data collection is complete. The software then takes the average of each sensor's location table, and stores this average RSSI in a table denoting the physical locations and each Sensor's average RSSI values:

```
+----------+----------+----------+----------+----------+----------+----------+
| position | S07_rssi | S04_rssi | S03_rssi | S05_rssi | S06_rssi | S02_rssi |
+----------+----------+----------+----------+----------+----------+----------+
|      101 |      -54 |      -45 |      -56 |      -68 |      -57 |      -55 |
|      102 |      -55 |      -40 |      -56 |      -50 |      -67 |      -61 |
|      103 |      -58 |      -40 |      -58 |      -52 |      -54 |      -62 |
|      104 |      -57 |      -40 |      -64 |      -60 |      -57 |      -55 |
|      105 |      -64 |      -55 |      -66 |      -72 |      -67 |      -50 |
|      106 |      -62 |      -54 |      -71 |      -56 |      -81 |      -49 |
|      107 |      -63 |      -60 |      -56 |      -53 |      -73 |      -45 |
|      108 |      -71 |      -50 |      -62 |      -56 |      -61 |      -41 |
|      109 |      -58 |      -63 |      -66 |      -56 |      -61 |      -48 |
|      201 |      -54 |      -44 |      -56 |      -54 |      -62 |      -56 |
...
+----------+----------+----------+----------+----------+----------+----------+
```

Position 101 relates to x=1, y=1 or (1,1). This table defines the 'Fingerprint' of the location as defined by each of the individual sensors' RSSI average (see Equation 4.1.

5.4.3 Localising Function

Once the database is filled with the values for each location, the user can select 2 different localising functions from the menu, 'single' and 'continuous'. The procedure for localising a single position is achieved by taking the RSSI's as stated above and matching them to the database. The initial chain of events for localising is very similar to steps 1 through 6 of the profiling stage except that the target sensor is in any location within the profiled area. The following steps will outline the events after 5.4.2 'Step 6' above, when the full suite of values has been received at the PC:

Step 1 Once a full suite of values has been transmitted to the PC, the localisation module then stores these values in a table called Monitor. This table has similar format to the Main fingerprint table above whereby the columns denote the RSSI values for each sensor. In this case each row contains a set of values for the number of samples:

```
+--------+----------+----------+----------+----------+----------+----------+
| sample | S07_rssi | S04_rssi | S03_rssi | S05_rssi | S06_rssi | S02_rssi |
+--------+----------+----------+----------+----------+----------+----------+
|      1 |      -78 |      -60 |      -60 |      -56 |      -62 |      -62 |
|      2 |      -78 |      -61 |      -60 |      -56 |      -62 |      -62 |
|      3 |      -78 |      -60 |      -60 |      -58 |      -62 |      -62 |
|      4 |      -77 |      -54 |      -60 |      -56 |      -62 |      -61 |
|      5 |      -77 |      -54 |      -60 |      -56 |      -62 |      -61 |
|      6 |      -77 |      -54 |      -60 |      -58 |      -62 |      -61 |
|      7 |      -76 |      -54 |      -60 |      -58 |      -62 |      -61 |
|      8 |      -77 |      -54 |      -57 |      -58 |      -62 |      -62 |
|      9 |      -77 |      -54 |      -57 |      -56 |      -62 |      -61 |
|     10 |      -77 |      -54 |      -60 |      -56 |      -62 |      -61 |
+--------+----------+----------+----------+----------+----------+----------+
```

Step 2 The above sequence completes until the table is populated with the required number of samples for localisation.

Step 3 To find the location of the sensor, the most simple approach is to find the MMSE of the Monitor table compared with the Main table. The Mean Squared Error is defined by:

$$E(n) = \sum_{k=1}^{m} \sqrt{(R_k - S_{k,n})^2} \tag{5.1}$$

where 'm' is the number of locations in the grid. 'k' is the sensor number, R_k is the current RSSI reading and $S_{k,n}$ is the database reading corresponding to the location 'n'. The minimum value of n from the list of E(n) is taken to be the position in the database of the closest position to the sensor.

Cherntanomwong & Suroso (2011) claim that using this technique, they achieved an average error of < 0.5m. See Chapter 6 for the results that this project produced using this technique.

5.4.4 Defining Safe/Unsafe areas function

The versatility of AIMS is further enhanced by the use of user-definable boundaries which can be changed at the user's discretion.

In order to create boundaries for the purpose of alarming the user when the sensor/child enters an unsafe area, a 'state table' was implemented into the database. The table holds the state information for each position.

There are 3 States, namely Safe, Unsafe and Transitional. These states are given to the sensor as it moves throughout the grid. This enables the software to make decisions based on its state only. A simple state diagram is shown for the AIMS monitoring system in Figure 5.6.

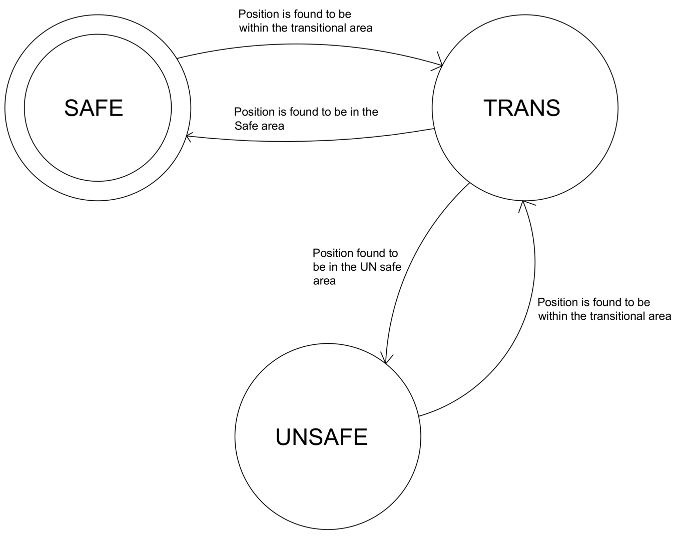

Position is found to be within the transitional area
SAFE
Position is found to be in the Safe area
TRANS
Position found to be in the UN safe area
Position is found to be within the transitional area
UNSAFE

Figure 5.6: State diagram for boundary control

The transitional state allows the transition between safe and unsafe states based upon real-world physical boundaries. For example, if AIMS were to detect a position change that physically cannot occur such as going through a wall, this would lead the system to alert the user on a false alarm. The transition state lets AIMS know that there is a thoroughfare between the safe and unsafe zones at that point and that a state change can occur through this area.

The sequence of events for setting the boundaries are as follows:

Step 1 — The user selects 'set safe/unsafe areas' from the AIMS main menu.

Step 2 — AIMS then prompts the for the position in which to change the state. All states are initialised to safe, so only the transitional and unsafe states need to be defined.

Step 3 — Once a user has input the position and the state, AIMS saves this state in the database. An example of the 'boundary' table is shown:

```
+---+------+------+------+------+------+------+------+
| y | _1   | _2   | _3   | _4   | _5   | _6   | _7   |
+---+------+------+------+------+------+------+------+
| 1 | s    | s    | s    | s    | s    | s    | u    |
| 2 | s    | s    | s    | s    | s    | s    | u    |
| 3 | s    | s    | s    | s    | s    | s    | u    |
| 4 | s    | s    | s    | s    | s    | s    | u    |
| 5 | s    | s    | t    | t    | s    | s    | u    |
| 6 | u    | u    | t    | t    | u    | u    | u    |
| 7 | u    | u    | u    | u    | u    | u    | u    |
| 8 | u    | u    | u    | u    | u    | u    | u    |
| 9 | u    | u    | u    | u    | u    | u    | u    |
+---+------+------+------+------+------+------+------+
```

Step 5 — After updating, the table, AIMS offers the user to exit or keep inputting values. Steps 1-5 are repeated until the user exits.

5.4.5 Graph Function

The Graph option has been used primarily for debugging purposes. It performs the same function as one iteration of 'monitor' except that it will create a text grid and place an X in the position that is the floor() of the x position and y position. This gives a visualisation as to the position of the sensor inside the grid.

```
newpos =  3.0000, 1.0000, e=12
current state = TRANS
last_state = SAFE
3.00, 1.00
| 0 0 X 0 0 0 |
| 0 0 0 0 0 0 |
| 0 0 0 0 0 0 |
| 0 0 0 0 0 0 |
| 0 0 0 0 0 0 |
| 0 0 0 0 0 0 |
| 0 0 0 0 0 0 |
```

The MMSE error value is also shown in the first line to assist with accuracy and algorithmic calculations.

5.4.6 Semi-Automated Accuracy Testing Function

A module was created to reduce the repeatability of testing and data entry into spreadsheets for statistical analysis. This module performs 2 tasks. The first task is to direct the user to the position where the sensor is to be placed. The second task is that it takes readings, converts them to a position using all localisation algorithm parameters for either the Weighted or Centroid algorithms[1]. The function saves the actual and expected results into a CSV file for processing by software such as Matlab and Excel. Each position is calculated a set number of times enabling the analyst the ability to see whether the results are spurious, or follow some uniformality.

The CSV file is saved with the current date and time as to give each file a unique timestamp as to when the test was started. A snippet of the file as it would be imported into spreadsheet is as follows. Note that all columns and rows are not included in this example.

Table 5.1: CSV file output from Test Function. Test_Data2014_10_19_13_29_16.csv

SAMPLE	ACT_X_POS	ACT_Y_POS	KNN	KC	EST_X_POS	EST_Y_POS	TIME_PER_SAMPLE	ERROR
1	1	1	3	1	5	7	0.11047	7.2111
1	1	1	3	1	5	7	0.12388	7.2111
1	1	1	3	1	5	7	0.16545	7.2111
1	1	1	3	1	7	5	0.17229	7.2111
1	1	1	3	1	6	8	0.14904	8.60233
2	1	2	3	1	1	3	0.14307	1
2	1	2	3	1	1	3	0.21927	1
2	1	2	3	1	3	6	0.16187	4.47214
2	1	2	3	1	1	3	0.19118	1
2	1	2	3	1	1	5	0.16718	3
3	1	3	3	1	6	7	0.15147	6.40312
3	1	3	3	1	4	5	0.15337	3.60555
3	1	3	3	1	3	5	0.16438	2.82843
3	1	3	3	1	3	5	0.1946	2.82843
3	1	3	3	1	3	5	0.13106	2.82843
4	1	4	3	1	2	4	0.16836	1
4	1	4	3	1	2	4	0.13192	1
4	1	4	3	1	4	5	0.13081	3.16228
4	1	4	3	1	4	5	0.13099	3.16228
4	1	4	3	1	4	5	0.15433	3.16228

5.4.7 Setup Function

The setup is very self-explanatory. The system asks a number of questions about the physical layout and sensors. One thing to note is that when the 'setup' function is called, the database will be deleted and profiling will be required again. The reason for this is that the tables in the database need to be set up with the correct sizes and columns. Furthermore, the setup phase checks the WSN for the sensors that it can hear and uses that information to create names and columns.

[1]The initial version of this file allowed the user to select the parameters, but proved slow and time consuming. It was updated during the residential testing procedures

The following shows the user interaction with the ‘setup’ function. All of these parameters are saved in the global settings data structure and then stored in the mirrored table in the database for future retrieval.

```
Current Settings:
Number of Sensors = 6
Number of Samples = 20
Size in x direction = 7
Size in y direction = 9
number of samples for monitoring = 3
k nearest nodes = 6
k centroid points = 3
-----------------------------
How many sensors are you using?   6
How many samples for profiling?   20
What is the x-size of the area?   7
what is the y-size of the area?   9
How many samples to take during monitoring? 3
Nearest nodes? 6
Centroid points? 5
```

As mentioned, AIMS listens to the network after the parameters are input. It makes a list of the ‘heard’ sensors and stores them until the number of sensors designated are found. The ID of the sensors is then used to create the table names for the positional data and the column names in the Main and Monitor tables.

5.5 Software Modules for Distributed Sensor Motes

The objectives of the software that resides on the mote(s) is to perform the functions outlined in Section 5.4.2 whereby there are 3 major tasks to accomplish. The first being for the target mote to periodically send beacon signals. Following this, the static nodes are required to capture the RSSI, and sequence number and route that back to the central computer using CTP. The third task is for the base-station or root node to forward these packets to the central computer as soon as possible upon arrival. All of these functions were written into the one program although the functionality of unused components is disabled by filtering out the Mote ID’s.

5.5.1 Target Node Function

The target node is the node that is worn by the infant. The task of the target node is to broadcast simple Active Messages to the static nodes carrying the sequence information for synchronisation. These messages are referred to in the code as ‘Beacons’. The structure of the code is much the same as the Radio Communication example set out in Section 2.5.3. The destination address is AM_BROADCAST_ADDR, therefore all motes receiving Active Messages will receive this message using their AMReceiverC.receive() event.

5.5.2 Static Nodes

The static nodes perform the task of receiving the Beacon messages from the target and packing it up in a frame for the Collection Tree Protocol to route back to the root node. The receive event is named BReceive.receive() (refer to Appendix C.1). The flow charts in Figure 5.7 describe the sequence of events that occur when a beacon message is received [2].

The other functionality performed by the static nodes is through the Collection Tree Protocol which is described in section 4.5. These static nodes also serve as forwarding mechanisms for the protocol. They perform underlying tasks such as determining the link quality to the nearest nodes and creating a routing gradient back to the root node.

5.5.3 Root Node (Base Station)

The base station has the task of receiving all CTP messages and forwarding them via serial communications to the Localisation PC. The flowcharts in Figure 5.8 describe the functionality and outline the calling structure. Refer to the listings in Appendix C.1 — in particular the Receive.receive() (as a starting point).

[2]Note the dashed lines in Figures 5.7 and 5.8 do not link the charts but give an indication as to the flow of events that the actual static node will achieve. Other events may occur during this time that are unrelated to the functionality shown here

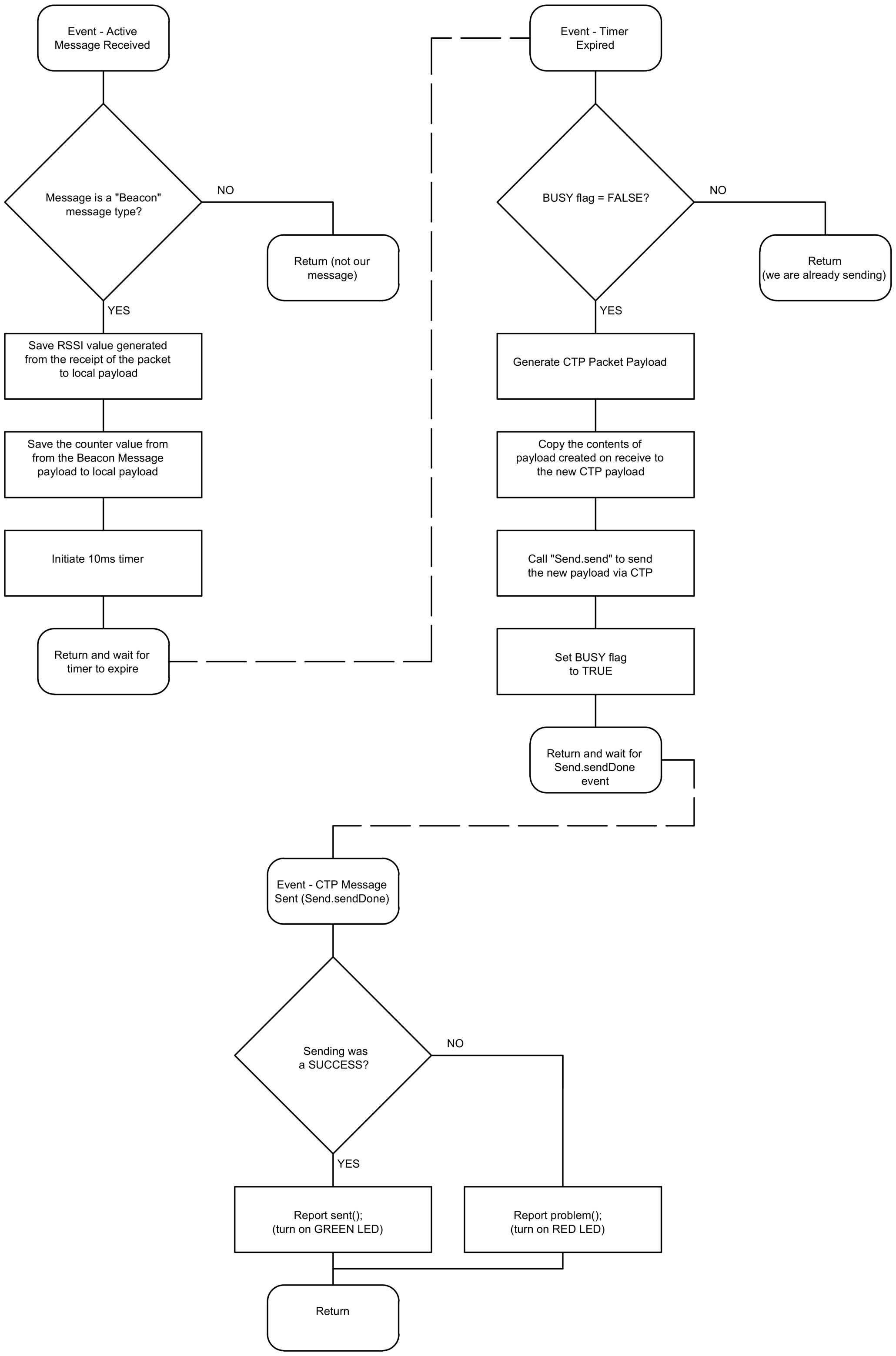

Event - Active Message Received
Message is a "Beacon" message type?
NO
Return (not our message)
YES
Save RSSI value generated from the receipt of the packet to local payload
Save the counter value from from the Beacon Message payload to local payload
Initiate 10ms timer
Return and wait for timer to expire
Event - Timer Expired
BUSY flag = FALSE?
NO
Return (we are already sending)
YES
Generate CTP Packet Payload
Copy the contents of payload created on receive to the new CTP payload
Call "Send.send" to send the new payload via CTP
Set BUSY flag to TRUE
Return and wait for Send.sendDone event
Event - CTP Message Sent (Send.sendDone)
Sending was a SUCCESS?
NO
YES
Report sent(); (turn on GREEN LED)
Report problem(); (turn on RED LED)
Return

Figure 5.7: Sequence of events for receiving a Beacon Message

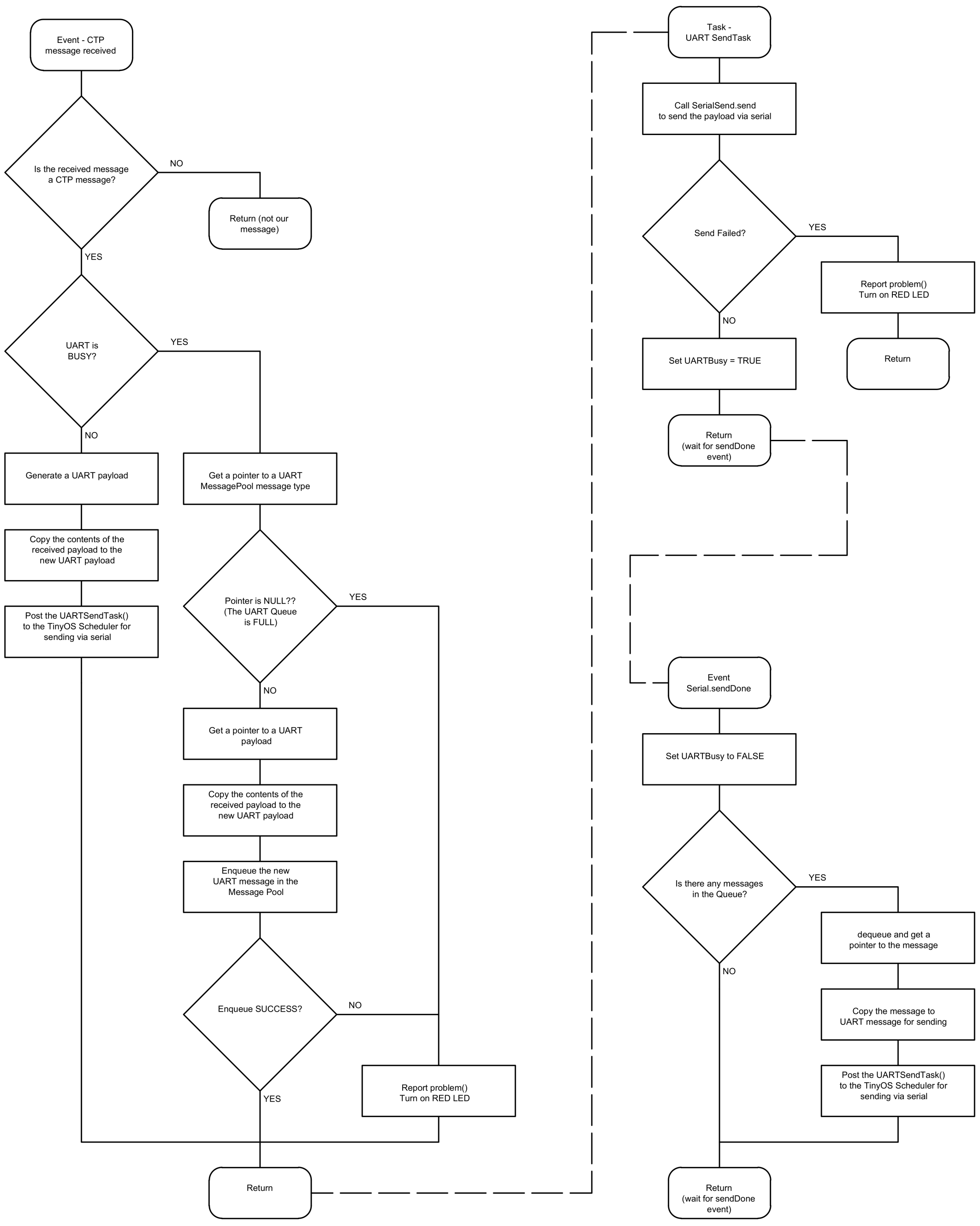

Event - CTP message received
Is the received message a CTP message?
NO
Return (not our message)
YES
UART is BUSY?
YES
NO
Generate a UART payload
Copy the contents of the received payload to the new UART payload
Post the UARTSendTask() to the TinyOS Scheduler for sending via serial
Get a pointer to a UART MessagePool message type
Pointer is NULL?? (The UART Queue is FULL)
YES
NO
Get a pointer to a UART payload
Copy the contents of the received payload to the new UART payload
Enqueue the new UART message in the Message Pool
Enqueue SUCCESS?
NO
YES
Report problem() Turn on RED LED
Return
Task - UART SendTask
Call SerialSend.send to send the payload via serial
Send Failed?
YES
Report problem() Turn on RED LED
Return
NO
Set UARTBusy = TRUE
Return (wait for sendDone event)
Event Serial.sendDone
Set UARTBusy to FALSE
Is there any messages in the Queue?
YES
dequeue and get a pointer to the message
Copy the message to UART message for sending
Post the UARTSendTask() to the TinyOS Scheduler for sending via serial
NO
Return (wait for sendDone event)

Figure 5.8: Sequence of events for the root node receiving CTP messages

5.6 Chapter Summary

AIMS was developed to encapsulate the whole system proposal including the WSN and centralised approach. The software developed on the wireless sensors was designed to route information efficiently to the root node where it is transferred to the AIMS localisation software for processing.

The localisation software that resides on the PC performs a number of tasks including setting up the wireless network parameters, profiling the working area, adjusting the algorithm parameters, and monitoring the location of a child in that network.

This project is multi-faceted and the software design is only one element that makes up the whole system. The software design in AIMS has moderately coupled modules and to take an objective approach, could have been better designed given more time. Albeit, the code performs the required tasks and does so with moderate ease of use.

The current AIMS system produces a text-only user interface which is very basic. The functionality to perform the tasks, however is enhanced by the fact that all relevant information is stored in a MySQL database. As mentioned a few times, this allows for further expansion in regards to remote client software accessing the information and providing a graphical and aesthetic user experience.

In regards to user-friendliness, the fingerprinting style of localisation holds little weight. Profiling takes time and a little background knowledge for the user. As this was the best performing method of localisation determined from the literature review, the method was chosen on that merit.

Future consideration must be taken into information security. With a system proposal such as AIMS, it is vital that the system is secure from hackers. This additional facet of the system is suggested for further research and implementation.

Chapter 6

Evaluation And Testing

6.1 Chapter Overview

To ensure the system can perform the desired outcomes, this chapter aims to define the evaluation parameters, their acceptable limits and to report on the results of testing these derived metrics. The average accuracy (in meters) and the response time are the two major metrics that are tested which determine if the AIMS software in conjunction with the algorithms and parameters are able to achieve the objectives of this project.

6.2 Evaluation Design

6.2.1 Evaluation Metrics

Localisation Accuracy — This is the most important metric for the project and should been tested thoroughly using different circumstances and algorithms. The basic method of calculation is the linear distance between the actual position and the calculated distance. An automated testing function has been in-built into the program for this purpose. The average error calculated by the algorithms in conjunction with the statistical analysis will be used to determine the accuracy of that particular test.

Latency - System Response Time —

> *"In a data collection tree, a node must handle the data of all of its descendents. If each child transmits a single sensor reading and a node has a total of 60 descendants, then it will be forced to transmit 60 times as much data. Additionally, it must be capable of receiving those 60 readings in a single sample period. This multiplicative increase in data communication has a significant effect on system requirements. Network bit rates combined with maximum network size end up impacting the effective per-node sample rate of the complete system".*(Hill 2003, pp. 24)

Following from the quotation above, we must provide the amount of data that can be transferred through the root node to the localisation PC as this node is the logistical bottleneck for the packet data. This will be measured by the amount of packets that get through per second. This will give the 'system response time', or latency, of the system. The total system response time will be the sum of the latency from the WSN and the time that it takes to localise by searching the database and performing the localisation algorithms. It is calculated by taking the system time before the localisation process begins, and then again after it has evaluated the position. These values will be plotted to show the distribution and statistical information.

Boundary Alarms — The alarming mechanism of the system will be evaluated by taking a sensor through a transitional zone to a unsafe zone and testing whether the alarm functions. This will give an indication as to the effectiveness of the system and will be used to determine if the system can perform the major objective of the project. It must be noted that this metric is highly dependent on the two aforementioned metrics. It is considered that the theory behind the alarm states will be accurate if, and only if the localisation accuracy AND the response time is such that an individual can be tracked inside each cell.

6.2.2 Design of Testing Procedures

The testing designs in this section have been included to evaluate the aforementioned metrics. The initial testing designs created individual spreadsheets for each of the algorithm parameters. This meant that the grid would have to be tested many times over. Additional functionality was added to the automated testing module after the second baseline testing stage. The new procedure calculates the estimated positions of the combinations of most algorithm parameters for each cell. This reduced the amount of iterations of the test and hence reduced the overall testing time. Furthermore, it meant that the algorithms were carried out on the same testing data. The second residential tests were completed using the redesigned testing module.

For clarity, the localisation tests were carried out as follows:

- Residential - Obstacles were included. Only the Best Match algorithm was used.
- Baseline - No obstacles. LoS to all static sensors from the target node. Best Match Algorithm. Created for a benchmark. Automated testing module created and used.
- Baseline - As above, but uses Centroid and Weighted Centroid algorithms. Device buffering error found and repaired.
- Residential - Obstacles included. Best Match, Nearest nodes, Weighted Centroid and Centroid algorithms tested. Automated testing procedure tweaked to perform analysis on the same data.

NOTE: The baseline tests were not revisited with the updated automated testing procedure as the original area was unavailable for access during the later stages of the testing period.

It must be noted that there are more tests that can be applied to this project, for instance varying the weighting factor of the Weighted Centroid Algorithm. Another variation would be to test whilst there are many people in the area interrupting the signal. The time consumed in testing these parameters would have been impractical to complete by the project deadline. These are included in the plan for further work.

Localisation Testing and response time

The practicality of the application defines that testing of localisation should be applied to real-life situations. The design of testing procedures are be based upon this ideal.

As previously mentioned, AIMS provides a testing function which returns the expected position as well as the actual position calculated by the algorithms. These values are saved into a CSV file for processing by mathematical or spreadsheet software. Furthermore, the system response time is also calculated from the real time clock. This is irrespective of the CPU time. The reason for this is to take into consideration the processes the operating system has scheduled in during normal system running. This will give a real life response time. The processing time, CSMA/CA time and database access time are all encapsulated by the timing functions and therefore the latency on localisation results will be

$$L(t) = \frac{\sum_{k=1}^{m} t_r}{m}. \tag{6.1}$$

Here t_r is the average response time and m is the number of samples taken. The following steps outline the testing procedure for localisation and system response:

Step 1 — Create a grid of the area to be tested using $1m^2$ cells.

Step 2 — Place sensors about 2m outside of the perimeter of the area that is desired to be tested.

Step 3 — Profile the area as defined in Section 5.4.2.

Step 4 — From the AIMS menu, select 'Test Accuracy'. Follow the prompts until the area has been tested. The CSV file will be saved in the format: Test_DataYYYY_MM_DD_HH_MM_SS.csv [1].

Step 5 — After the test is complete, open the file manager and proceed to the location where the AIMS software resides. Open the CSV file with EXCEL or OpenOffice Calc. The data will be imported in their corresponding rows and columns. Each sample is completed 5 times, therefore the sample column will contain 5 of each of the same number. The columns will contain the sample, actual x position, actual y position, value for nearest nodes, value for centroid vertices, time response, and calculated error.

Step 6 — The data in the error column has been calculated using Pythagoras' theorem based upon the estimated position from the algorithm as follows:

[1]It is advisable to note the time that the test was started to help locate the file.

$$e(n) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \ ,$$

where n is the number of cells in the grid multiplied by 5 (samples). x_2, y_2 is the estimated position and x_1, y_1 is the actual position.

The average of this column has been included at the bottom of the CSV file so the user is required to perform less calculations into comparing the metrics.

Furthermore, the system response time is also averaged at the bottom of the CSV file. The results of this testing is shown in Section 6.3.

Common House Localisation Testing

Using the procedure set out in Section 6.2.2, the system should be tested in a common home to evaluate if the system is accurate enough to perform in a real life situation.

Using the results from this test in comparison to the baseline testing allows us to establish the differences between an area with obstacles, walls etc to one with direct LoS to the target sensor.

Baseline Testing

Baseline testing is performed with a complete LoS from all static sensors to the target node. The reason for baseline testing is to gain a benchmark to compare the real life accuracy to. It will give an indication as to the effectiveness of the system as compared to the baseline which is expected to be the most accurate. It uses the procedure in 6.2.2 in conjunction with the following parameters.

Step 1 — Create a flat area with cells denoting the x and y positions in the grid.

Step 2 — Place the sensors approx 2m from each corner of the grid and so they are in direct LoS of every cell in the grid.

Step 3 — Perform a profile of the area.

Step 4 — Perform the localisation tests as described in 6.2.2 and import the results into a spreadsheet for comparison.

Algorithms

Table 6.1 outlines the tests that were designed in order to make a calculated assessment of the results. The Baseline Test was only completed using a TX power of 20 because of limited access to the test area in conjunction with the testing time. For comparison between the Baseline and Home test, only this transmit power will be considered.

Table 6.1: Algorithm Test Table

Test No.	TX Power	Algorithm
1	20	Best Match
2	20	Centroid
3	20	Weighted Centroid

For clarity, Test 1 uses the centroid algorithm, but has the vertices of the centroid set to 1, therefore this equates to a single point which is the best match in the database.

The results from these tests are to be further analysed by using statistical analysis. Upon finding the best performing parameters in each of the locations (baseline and home), they are compared against each other to understand the differences between a real life situation and a LoS situation.

6.2.3 Boundary Test Design

The boundary test is to be carried out using the same profile and localisation data as one of the most favourable algorithms and parameter sets taken from the previous tests. The principle of this test is to establish whether the alarms are raised when they should be, and whether the localisation accuracy effects the alarming mechanism.

The following steps define the Boundary and Alarming Test:

Step 1 After the localisation tests have been completed and analysed, choose one of the algorithms and parameter sets for this test. Enter those parameters into AIMS.

Step 2 Define the safe and unsafe areas so that there is a 2 cell wide transitional zone through a doorway or similar. For example, the following table shows the transitional zone centring around (3,5):

	1	2	3	4	5	6	7
1	s	s	s	s	s	s	s
2	s	s	s	s	s	s	s
3	s	s	s	s	s	s	s
4	s	s	s	s	s	s	s
5	s	t	t	t	s	s	s
6	u	t	t	t	u	u	u
7	u	u	u	u	u	u	u
8	u	u	u	u	u	u	u
9	u	u	u	u	u	u	u

Step 4 Start with the sensor at a position in the safe zone. Select 'Monitor' from the AIMS menu. The results should scroll the screen returning the position and a graph on each successful localisation. The state at that position should also be displayed such as “SAFE” or “UNSAFE” or “TRANS”.

Step 5 Move the sensor through the transitional zone to the unsafe zone and record the results. Upon entering the unsafe zone, AIMS will report that the boundary has been breached.

Step 6 Repeat these steps a number of times and report the findings.

It is important to note that actual functionality of the software is being tested here. The test is deemed successful IF and only IF the software reports a boundary breach when the sensor passes from a transitional zone to an unsafe zone. If the localisation result passed to the monitoring function is erratic, this problem does not affect the functionality of the state machine (refer to Figure 5.6).

6.3 Testing and Results

The following sections cover the actual tests performed and give an analysis of the data obtained.

6.3.1 Localisation Testing

Initial testing was performed in the residential situation. Surprisingly, the initial test results were quite erroneous. As a result, baseline tests were then designed as per

Chapter 6.2.2 to create a reference for possibly a best case scenario. It was through the initial stages of the baseline testing that a buffering issue with the device driver for the root node was found. This problem has been concluded to be the cause of the error in the initial residential tests. The problem was rectified in the code and the baseline testing was carried out as described.

Figure 6.1 shows the residential situation that was used to carry out the tests. The grid gives a visualisation as to the cells in the database and their real life size and applications.

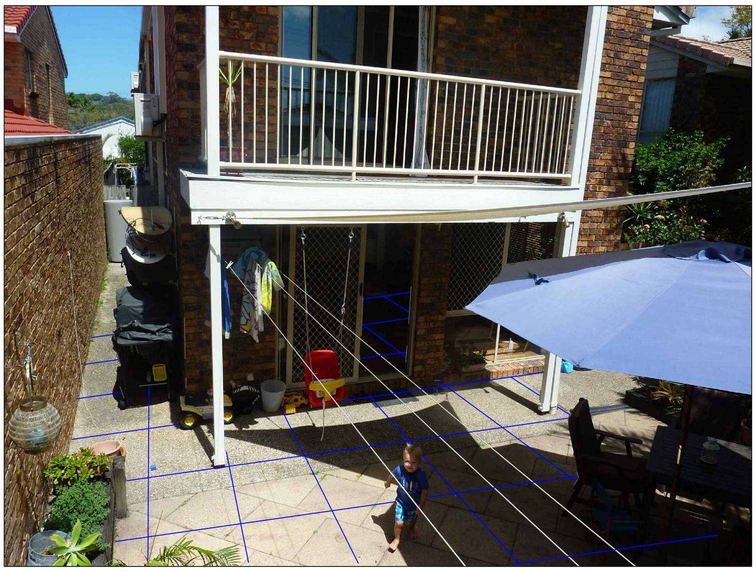


Figure 6.1: Common residential home showing overlaid grid for visualisation

All tests were carried out with the Telosb motes set to channel 20 to lock to one specific channel. The baseline testing was carried out with the TX power register value set to 20.

This section describes the tests and how they were performed on a practical level. The results are then analysed and discussed.

Residential Test 1

The first localisation test was performed in the preferred use case of a normal home. The test used a grid of 5m x 7m with $1m^2$ cells. The area was profiled immediately before the test was performed.

The results returned from a short analysis showed very scattered reading and a normal mean of >3.5m. These readings did not reflect the claimed results in the literature such as the claims from Cherntanomwong & Suroso (2011) of <0.5m average. No further tests were carried out until a solution was arrived upon. The reason for the very low accuracy was found after the baseline test was designed and carried out.

Baseline Testing

Figure 6.2 indicates the actual test area that was used for the baseline test. LoS was achieved for all cells in the grid and the area was away from stray RF signals. A thorough test was conveyed initially covering 690 data positions and taking approximately 20 hours to complete.

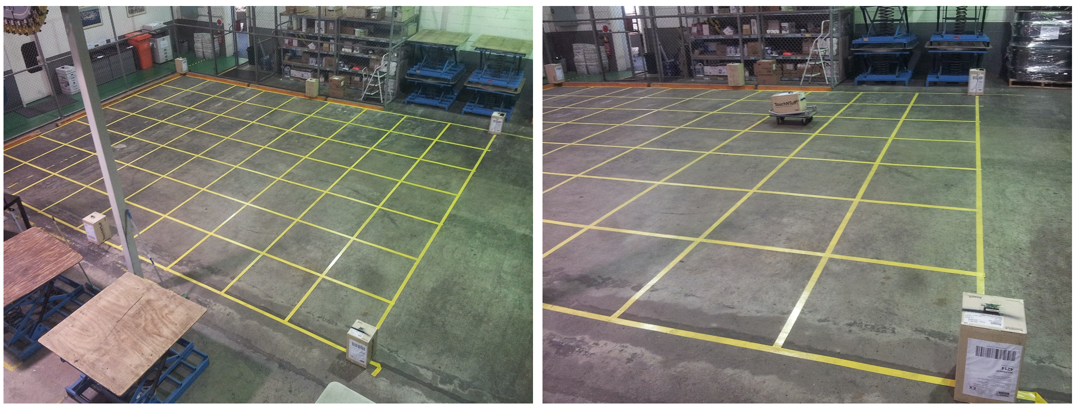


Figure 6.2: Testing the sterile area with LoS for all positions

During the final tests of this testing stage, it was found that the data read from the WSN would be taken only at select times and not directed by the software. Further investigation uncovered an issue with device buffering, hindering the ability to take in the real time readings from the WSN. Many hours were spent performing the testing designs, however, the results were not as expected. Figure 6.3 shows the scattered data from the initial baseline test. As can be seen by the plot, the results are non uniform and very inaccurate. As the buffering problem was discovered at this time, a statistical analysis was completed on these results and the error was found to be approx. 3.67m.

The problem was rectified by clearing the device buffer before each set of packets were read. The Baseline test was successfully completed over the next 16 hours [2].

[2]this discludes the prior tests that proved invalid which took a similar amount of time

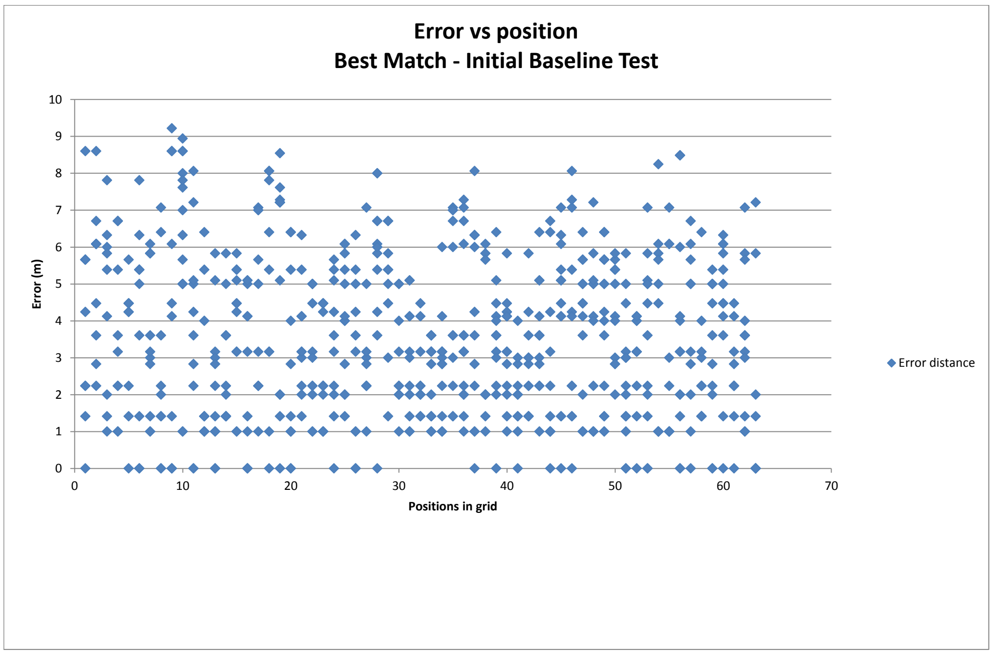


Figure 6.3: Scatter Plot for initial Baseline test. Indicates very random and spurious readings.

The results of the two tests for the Best Match algorithm were:

Table 6.2: Baseline Test - Best Match Algorithm

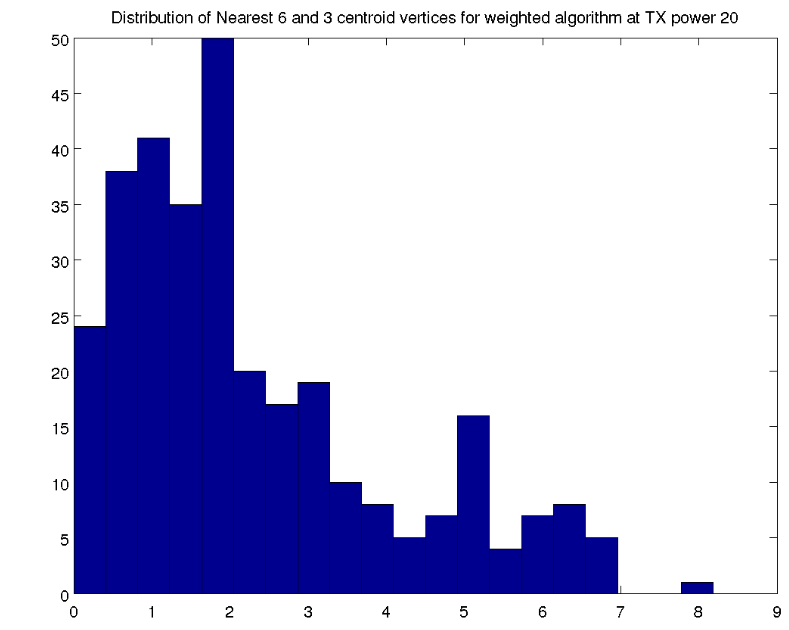
Test.	TX Power	Mean
Initial	20	3.67m
Secondary	20	3.40m

The slight improvement was still far from the stated values in the literature whereby Cherntanomwong & Suroso (2011) claim 0.5m average error. The tests in the residential scenario using baseline tests back up this data to strengthen the argument that the best match algorithm cannot achieve an accuracy of 0.5m using this hardware and in these scenarios.

Redesign of algorithm and Baseline Test 2 results

As accuracy is the key metric in this project, the design stage was revisited as set out in the project specification. At this point, the Centroid Algorithm and Weighted Centroid algorithm were developed. These algorithms are defined the Section 4.3 and 4.4.

After testing these algorithms, the data was imported into Matlab where basic statistical analysis was performed. Histograms were created from each of the data sets and their shape (positively skewed) was confirmed by plotting the data-points against the various distribution probability plots in the Matlab 'probplot' function. Figure 6.4 shows the positively skewed histogram and its fit to the Rayleigh Distribution. This probability plot shows that the data conforms to that distribution up until the 90th percentile.



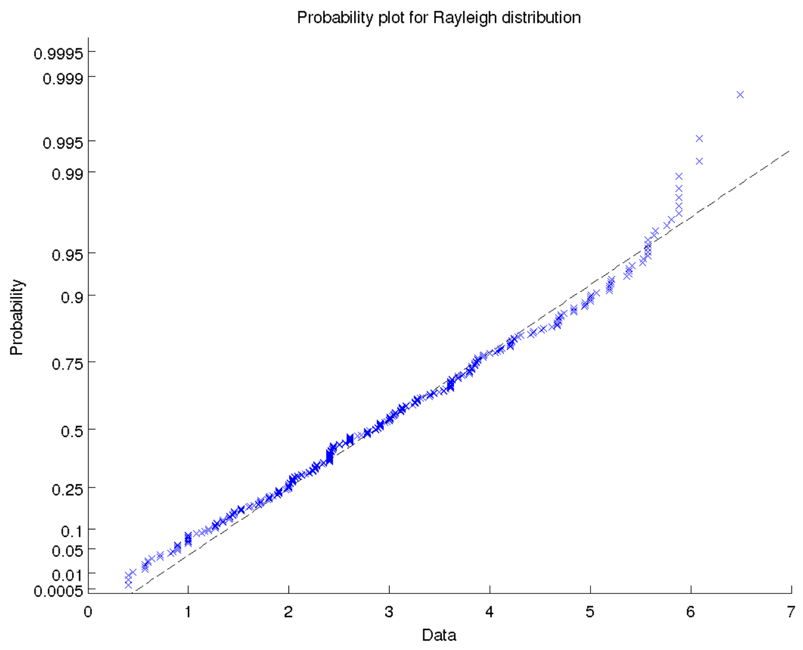


Figure 6.4: Baseline test distribution showing positive skewing and the probability plot against the Rayleigh Distribution.

To calculate the mean of this data, the Rayleigh mean was used as the data was not normally distributed. Using the Rayleigh distribution functions in Matlab, it was possible to get a clearer picture of the real mean of the data and its distribution. These metrics were then applied to the testing procedures for new comparisons.

The results for the Baseline tests showed a minimum mean error of 2.17m using the weighted centroid algorithm. Table 6.3 shows the other results attained. The Weighted centroid clearly outperforms all others in this scenario.

Table 6.3: Comparison of algorithms for baseline tests

Algorithm	Weighted	Centroid
Best Match - 3 nearest nodes	3.04	N/A
Best Match - 4 nearest nodes	2.85	N/A
Best Match - 5 nearest nodes	2.89	N/A
6 nearest, 2 vertices	2.38	2.91
6 nearest, 3 vertices	2.53	2.42
6 nearest, 4 vertices	2.26	2.29
6 nearest, 5 vertices	2.17	2.41

Residential Test 2

The second residential test batch was completed using the information gathered during the previous test and design iterations. A grid of 9x7m was used which incorporated both indoors and outdoor areas. 6 nodes were used as was the case for the baseline testing. Table 6.4 contains the same tests as Table 6.3 for comparison. As previously stated, the newer automated testing module created more test data to be analysed.

Table 6.4: Comparison of algorithms for residential rest

Algorithm	Weighted	Centroid
Best Match - 3 nearest nodes	2.77	N/A
Best Match - 4 nearest nodes	2.80	N/A
Best Match - 5 nearest nodes	2.76	N/A
Best Match - 6 nearest nodes	2.45	N/A
6 nearest, 2 vertices	2.18	2.24
6 nearest, 3 vertices	2.10	2.08
6 nearest, 4 vertices	2.06	2.03
6 nearest, 5 vertices	2.04	2.04

As a better comparison of the algorithms themselves, the full test data has been analysed. This test data was plotted on the same axes to give a comparison of the best performing algorithm over the parameter set. The results in Figure 6.5 show that the Centroid algorithm outperforms the Weighted Centroid in all but a few of the parameter sets. Ultimately, the Centroid algorithm achieves the best accuracy as stated above.

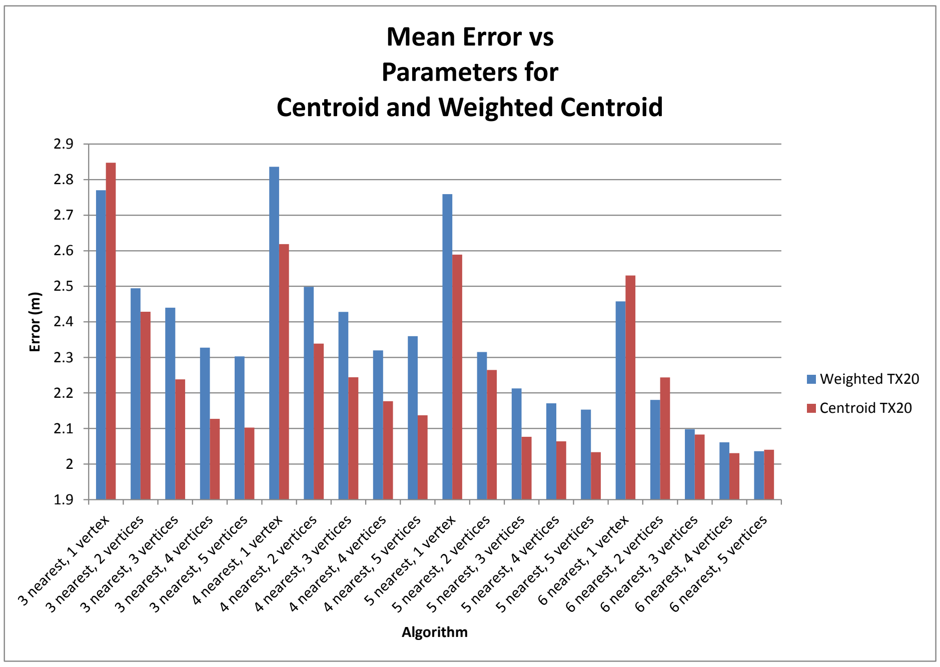


Figure 6.5: Mean Error vs Parameters for Weighted Centroid and Centroid Algorithms

6.3.2 Comparison of Localisation Tests

A comparison of the localisation test results revealed that the residential scenario was more accurate than the baseline tests. Figure 6.6 visualises these results.

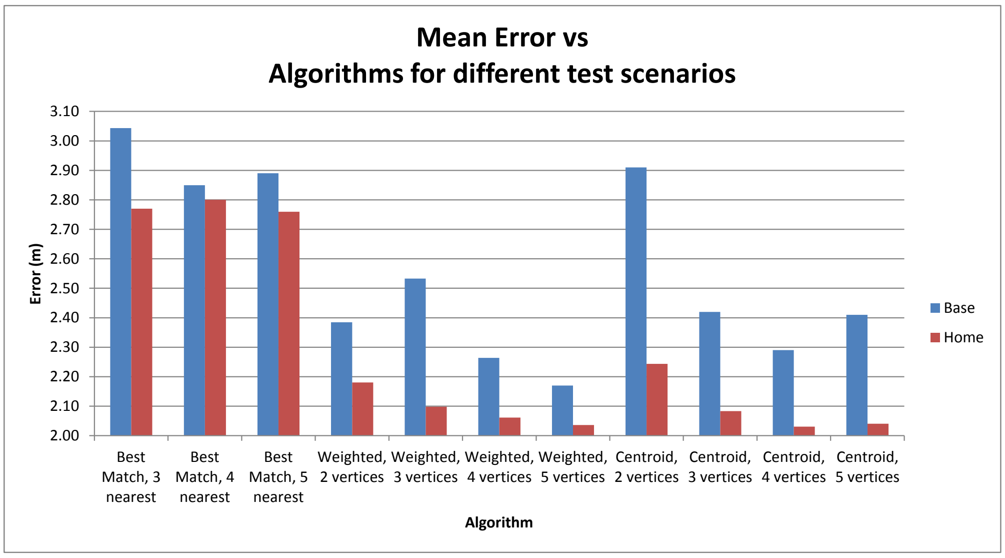


Figure 6.6: Mean Error vs Algorithms for Home and Baseline tests

There are some possibilities as to the reason the baseline tests did not prove to be more accurate. On reason is the fact that there is no attenuation factor such as walls and obstacles in the baseline test giving a certain distinctness to the readings. The theoretical exponentially decaying RSSI signal strength would apply less in the residential situation with walls and furniture that attenuates the signal further.

The second possibility is that the baseline algorithm tests were carried out with their parameters set individually. This means that for each received position data, only one set of parameters and algorithm acted on that data. As previously mentioned, access to the same grid wasn't permitted to perform the tests using the newly developed automated testing procedure.

In relation to the results attained, it can be concluded that the best performing algorithm based on the test data was the Centroid Algorithm at 4 vertices. This equates to the centroid of a polygon of the 4 best database matches. This resulted in a mean error of 2.03m.

The Weighted Centroid algorithm's internal parameters were not tweaked during this testing procedure. Tweaking the weighting factor adds another dimension to the testing procedure, and holds value for future tests. It can be speculated that the correct tuning on the weighting factor could in turn reduce the error below 2m.

6.3.3 Sensor Orientation

To further analyse the data for uniformality, the orientation of the sensor was taken into account. The test was performed using the Weighted centroid algorithm with 5 vertices which proved to be one of the best ranked schemes. The sensor was placed at an arbitrary position (3,3) and was rotated 90 degrees for each test. Each of the 4 orientation tests took approx 30 localisation results. The orientation of the sensor are:
Orientation 1 = $\uparrow$
Orientation 2 = $\rightarrow$
Orientation 3 = $\downarrow$
Orientation 4 = $\leftarrow$
where Orientation 1 faces the y=1 horizontal. The distribution plots in Figure 6.7 clearly show that the orientation of the sensor affects the localisation results. The red bars denote the frequency at the X positions whereby the blue denote frequency at the Y positions.

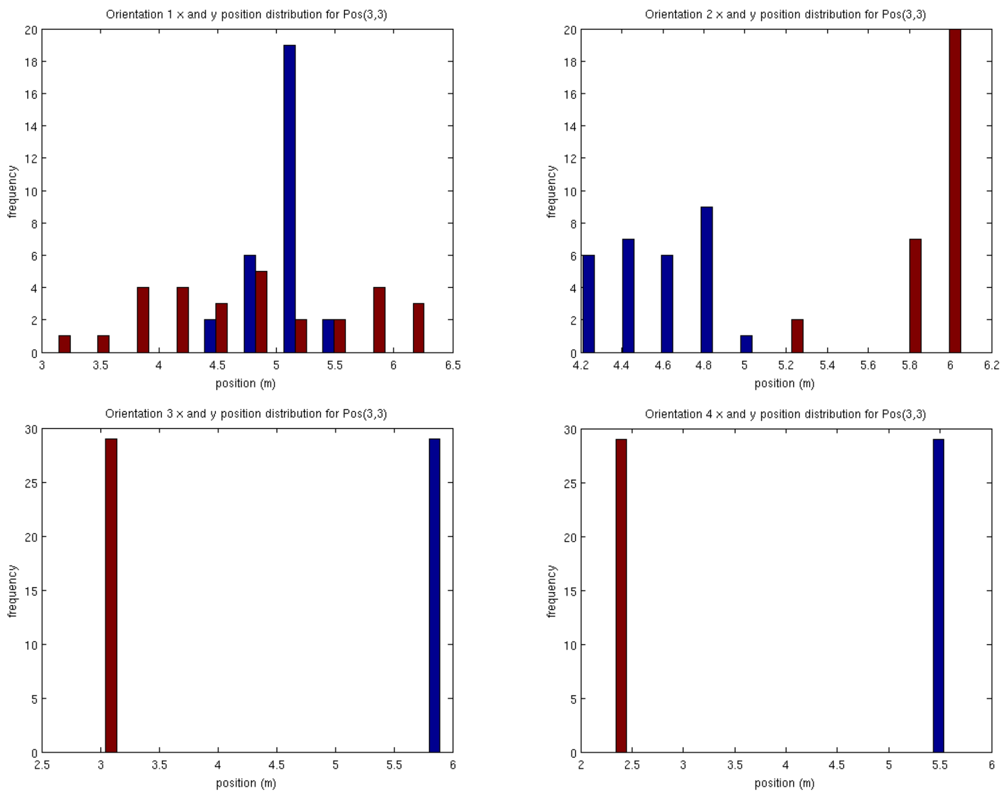


Figure 6.7: Location distribution for different sensor orientations

As a secondary test, a whip antenna was attached to the target node and the test was re-performed. Figure 6.8 is the result of this test.

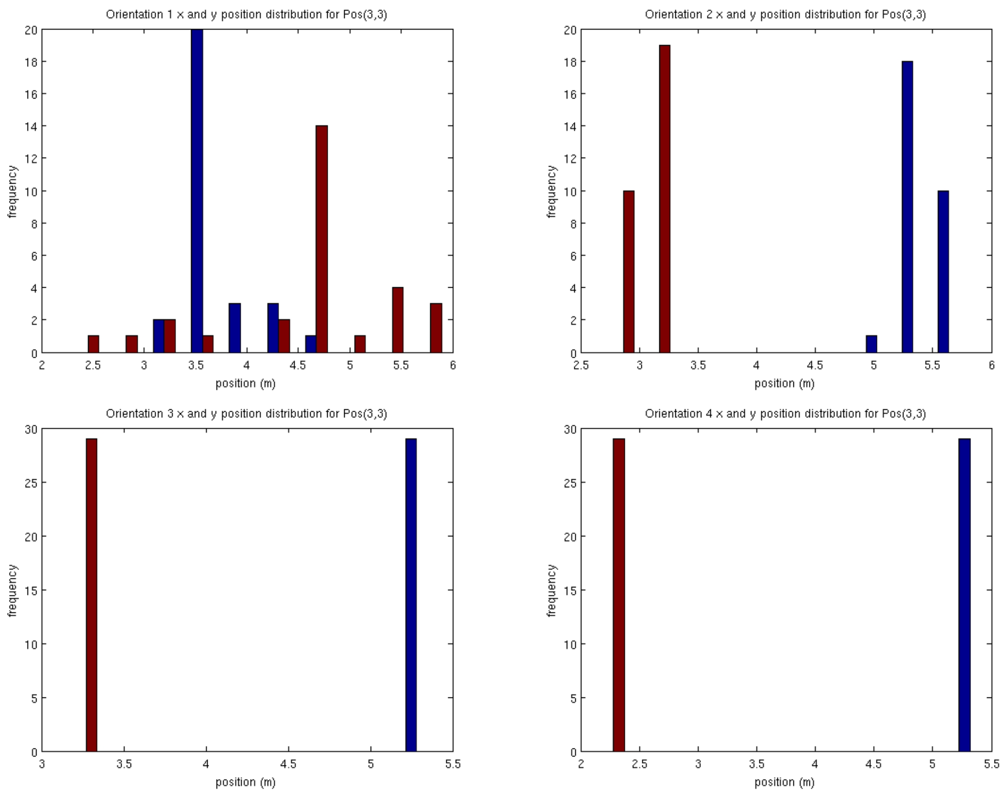


Figure 6.8: Location distribution for different sensor orientations

As both of these tests returned similar results in regards to the orientation of the device, further research was embarked upon late into the testing stage. The research was centred around the signal propagation of CC2420 devices. It was found that the propagation signal is not omni-directional from a CC2420 with the standard PCB antenna. This can be confirmed by the empirical results attained by Azevedo & Santos (2007). Their study shows the propagation of the CC2420 to have no omni-directionality on any plane. The sensor used to completed their study was the MicaZ sensor which has also been designed by Crossbow. The radio propagation model with their sensor at orientation 1 is shown in Figure 6.9. The propagation shows that there is a 9db difference depending on the location of the static sensors around the node.

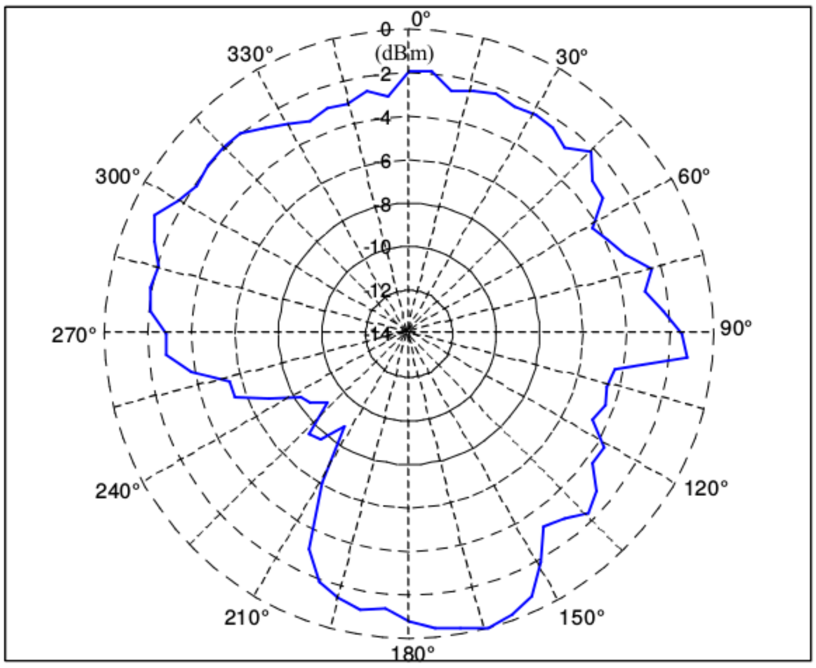


Figure 6.9: Propagation Model of MicaZ mote with cc2420 Radio Chip (Azevedo & Santos 2007)

A similar test was carried out using the RSSI from the Target Sensor to a static node sensor. This test was performed in a park away from obvious radio signal interference or reflective surfaces and obstacles. The target sensor was positioned inside a circle and the static node was positioned 3m away from the target in a circular shape. The TX power of the target sensor was set at 25 or -2dB. The results of the test are shown in Figure 6.10 where it is clear that the signal propagation from the TelosB mote using the on-board antenna and CC2420 radio chip is not omni-directional. The table of the testing results can be found in Table E.3.

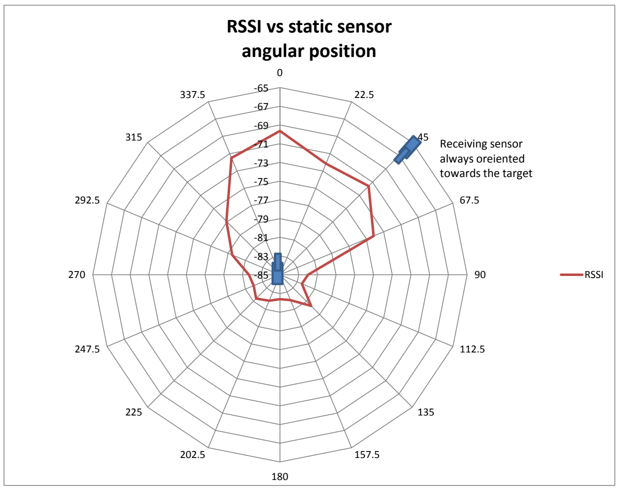


Figure 6.10: RSSI Vs Angle of reference for TelosB mote at 3m.

As this characteristic greatly reduces the reliability of the signal received, it can be hypothesised that if the transmitted signal is not omni-directional, the accuracy of the system is greatly hindered. The size and compactness of the sensor would need to suit the application as does the effectiveness of the device. Additional research uncovered an antenna designed in 2010 by Tze-Meng & Reza (2010). They created an omni-directional antenna which suits the 2.4Ghz radio band. It is 51mm x 16mm x 0.8mm which is slightly smaller than the PCB that comprises all of the TelosB components. A suggestion for further work would be to initially test the use of the antenna by gathering empirical results and further utilise this style of antenna in an industrial design incorporating the sensor hardware and antenna.

6.3.4 System Response Time

The system response time was recorded during the automated testing results collected for the localisation data. This data was collated and imported into Matlab for statistical analysis.

The histogram for the response time data shows that the distribution is of the Rayleigh Distribution as was the localisation data. There area a number of reasons that the probability curve tracks away from the Rayleigh distribution after the 95th percentile. One reason is that the packet may not have been delivered through the CTP protocol which causes the next packet to be taken and therefore a delay of longer than 220ms would elapse. Another reason is that the signal was not heard at one of the sensors. Again, this will cause a further delay for a full suite of packets to be read by the AIMS localisation software.

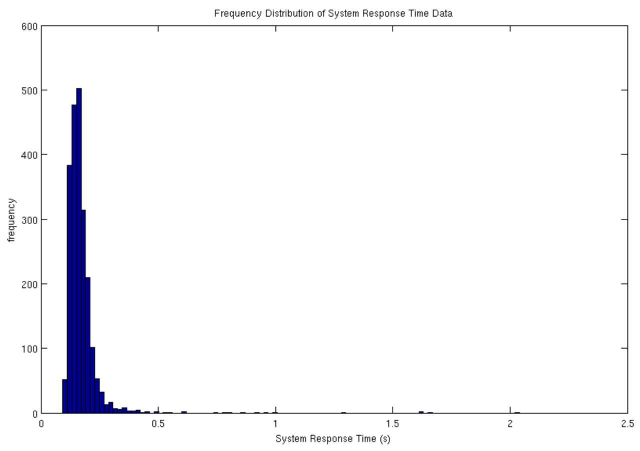


Figure 6.11: Distribution Plot of System Response Time

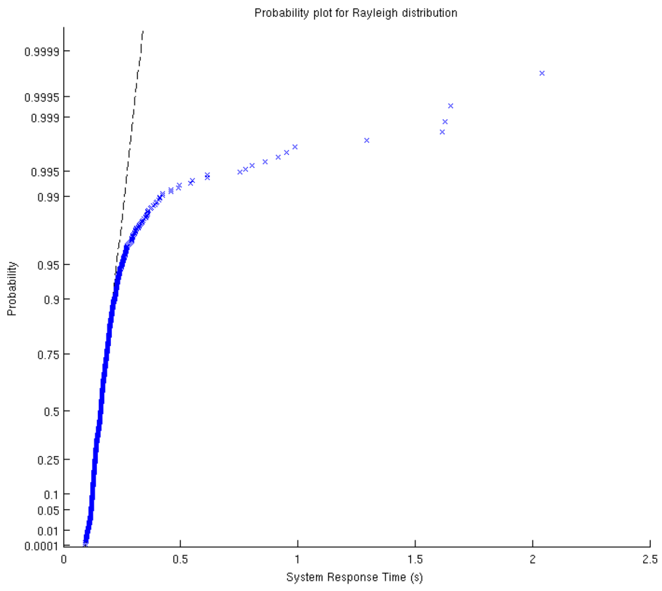


Figure 6.12: Probability Plot of System Response Time

The Rayleigh Mean system response time is calculated to be 0.175 seconds or 175ms. To take a practical viewpoint on this result, we must make an assumption. We must assume that the child is to be tracked in every cell no matter of his/her speed. This is so that the alarming module has the best possible chance to detect a boundary crossing.

Therefore the child can not be moving any faster than: $\frac{1}{0.175} = 5.7m/s$.
This equates to $\approx 20km/h$ which is definitely unattainable for a child under 5 years old considering that the four minute mile achieves requires a speed of 6.67m/s and is considered a challenge for a adult athlete.

Taking this response time into consideration and careful forward planning around localisation accuracy, the speed of the system can be deemed acceptable.

6.3.5 Boundary and Alert Test

The boundary and alert test defined in Section 6.2.3 was carried out a number of times. The following output shows the successful rejection of a state change without detecting a transitional intermediate state:

```
=========================================
OLD Position = 5.163596, 5.170851
newpos =  5.0611, 5.7424, e=33
current state = SAFE
last_state = SAFE
New Position is set to 5.061055, 5.742446
=========================================
OLD Position = 5.061055, 5.742446
newpos =  5.0166, 6.1037, e=34
current state = UNSAFE
last_state = SAFE
Moving from SAFE to UNSAFE without being in TRANS not allowed...
keeping position
New Position is set to 5.061055, 5.742446 (old position)
=========================================
```

The following output shows the successful state change from SAFE to TRANS.

```
=========================================
OLD Position = 3.212254, 4.252542
newpos =  4.4013, 5.9706, e=29
current state = TRANS
last_state = SAFE
New Position is set to 4.401327, 5.970576
=========================================
OLD Position = 4.401327, 5.970576
newpos =  4.3624, 5.9819, e=30
current state = TRANS
last_state = TRANS
New Position is set to 4.362372, 5.981872
=========================================
```

The following output shows the successful transition from a transitional zone to an unsafe zone. As mentioned, this state in the software can trigger a variety of different alarm mechanisms. For example a digital output could be used, or even an email or message could be sent.

```
==========================================
OLD Position = 4.401327, 5.970576
newpos =  4.3624, 5.9819, e=30
current state = TRANS
last_state = TRANS
New Position = 4.362372, 5.981872
==========================================
OLD Position = 4.362372, 5.981872
newpos =  4.7633, 7.2312, e=22
current state = TRANS
last_state = UNSAFE
SENSOR IN UNSAFE AREA
New Position is set to 4.763312, 7.231243
==========================================
```

Note that the only *functionality* of the AIMS software was tested during this test. Outside factors such as location accuracy or spurious readings were ignored so that the focus of this test could be achieved. It must be mentioned that false alarms are common due to the randomness of the localisation data getting fed into the state machine. This further enhances the need for finer grained localisation in order to compliment a functional system.

6.4 Chapter Summary

This chapter defined the metrics that were tested in order to deduce whether the system was able to achieve the project directives. These metrics included system response time, localisation accuracy, and boundary tests.

The localisation analysis and results described in this chapter return an average error of 2.03m as a best possible case using the Centroid Algorithm. This algorithm compared the best in most cases against the Weighted Centroid and the Best Match algorithms. The exhaustive testing during this time concludes that the AIMS system cannot achieve the 0.5m mean error stated by Cherntanomwong & Suroso (2011) using the same concept. In fact, using the simple Best Match algorithm, the error was 3.67m on average.

It was found that increasing the number of nodes (node density) improved the results also. Tests showed that using the nearest k nodes whilst fingerprinting was detrimental with a decreasing k.

The comparison between Baseline Testing and Residential Testing proved a worthy exercise. It shows that using the fingerprinting technique, accuracy improves significantly with obstacles such as walls and furniture etc. This can be put down to the attenuation of the signal as it passes through a different medium creating more defined signatures of some positions.

It was found that the orientation of the sensor was a major influence in the localisation result. This led to further research into the propagation of the signal generated from the CC2420 with an on-board antenna. Further tests unfolded that the propogation was not omni-directional and therefore restricts the orientation of the sensor in order to achieve more uniform results.

Fortunately, the non-uniform signal propagation does not render the AIMS system invalid. An omni-directional antenna was found which suits the size and practicality of this project. This antenna style coupled with the AIMS system has merit for further research and trails in regards to reducing the localisation error.

The system response time resulted in an average latency of 175ms. This response time is considered satisfactory if the accuracy of the system is reduced to below 1m.

Taking all metrics into consideration, it must be concluded that the accuracy of the system is the driving factor for success or failure. In the current state, the AIMS software requires further work in regards to localisation accuracy and signal propagation This will enable the other metrics described above to achieve synergy for the whole AIMS system.

Chapter 7

Conclusions and Further Work

This project aimed to research the viability of using a wireless sensor network based on the IEEE812.15.4 standard for use in localisation and boundary alarming in residential areas. The primary scenario is directed towards child safety and to reduce the number of children drowning under the age of 5.

The concept proposed was to use the radio signals from a TelosB Wireless sensor to effectively localise that sensor in a WSN. The location of the sensor is then to be used to derive whether the child is within user-defined boundaries and hence "safe". The versatility of the proposal is that the system has no set scenario and can be extended to other situations such as tracking of Alzheimer's patients or of children in the vicinity of water tanks or bath-tubs.

The following section will summarise the process and findings undertaken during this research and empirical study. The last section will sum up the key findings of the dissertation and the measure of achievement to the research community.

7.1 Research Objectives - Key Findings and Conclusions

Initially, a market research was undertaken to establish the viability of such a product in the marketplace. It was found that the WSN market is gaining momentum and evidence shows that much of the market prefer DIY WSN systems. It was also deemed that the system proposed should cost less than $300 to ensure it is sellable in the marketplace.

In addition to the market research element, research into the existing solutions covering child safety application was completed. This research revealed that there was a niche in the market for a system that is versatile and can provide the functionality set out above.

A thorough review of the literature surrounding localisation techniques and their application to indoor scenarios was undertaken. This area proved to be well researched. The localisation strategies were broken down into sub-categories and their characteristics were rigorously examined. The range of complexity of the localisation procedures defined in the literature ranged from relatively simple, to largely complex. It was found that the claims of localisation accuracy using different approaches was somewhat contradictive in some instances. The practicality of the systems was analysed for its usability in a residential area and it was concluded that the "Fingerprinting" method of localisation would prove to be the most accurate. Some authors claimed an average localisation error as low as 0.5m using this method.

Background research into the CrossBow TelosB wireless sensor revealed that it would be the device of choice because of its adherence to IEEE802.15.4, which defines low-cost, low-power devices. These sensors were supplied by the University of Southern Queensland as the prime candidate for the project and they were specifically designed for for research and innovation. They are designed to use the TinyOS operating system which is coded in NesC and the TinyOS API allows access to low level functionality of the radio interface.

Built on these foundations, the proposed system called AIMS (Adaptable Infant Monitoring System) was introduced. It comprises a total software/hardware package for WSN communication, localisation, and boundary alarming. The major hardware components are comprised of the wireless sensors (TelosB motes), a central computer complete with USB interface and connection to the internet for remote information retrieval. The system design elements for the software include a user interface, an interface to the wireless sensor network, and an interface to a storage element which allows remote retrieval of information. The MySQL database system was chosen as it is free, simple and there is a vast online community offering help and tutorials.

The functionality of AIMS was satisfactory and performs the required tasks to achieve its objectives. Time constraints on the project forced the software design to lack in elegance and a well defined structure. Furthermore, as this is a safety device, software integrity tests would have to be completed thoroughly before delivery to the market. It is also suggested that system security be applied to the system in the form of encryption and defence against attacks such as communications jamming or DoS attacks.

The nature of the design and testing of this project put to work the concepts and claims founded in the literature. The choice of localisation method was based on these claims, however, the localisation accuracy that was attained was not as expected. Additional algorithms that were produced did reduce the mean error by approximately 17%, but not to acceptable levels for a child safety device. The causes for the significant error were further researched and it was found that the signal propagation model from the sensor was not homogeneous. It is proposed for further work to research omni-directional antennae and their effectiveness of reducing the error of the system. Furthermore, tweaking the internal parameters of the weighting algorithm has the potential to increase the accuracy and is therefore a suggested research point to be embarked upon.

The actual response of the system was found to be acceptable and, given a higher degree of accuracy, is able to track the position of a child in each 1m x 1m cell.

The aesthetic nature of the sensors require additional research in regards to their physical size, wear-ability and mounting positions. Both the target sensor and the static sensors would need to be designed in such a way that it doesn't impede on the environment that they are utilised in.

The major aspect of this project that holds strong momentum for a niche in the market is the versatility of the boundary settings. No system has been found that offers the option of user-defined boundaries and hence enabling limitless application scenarios. Further research into the application scenarios that could be applied to this sensor would give great merit to the sale-ability of the system.

7.2 Closing Summary

The measure of success or failure is generally considered to be whether the objectives of the project have been achieved. In this case, the empirical nature of this project uncovered some facts surrounding localisation in WSN's that are very rarely mentioned or considered in the literature. This project achieved a best average error of 2.03m for localisation using derived algorithms and found that the uneven signal propagation model results in far noisier readings than expected. Therefore the contribution that this dissertation gives to the research community is the realisation that these factors must be considered in further research in the effective use of RSSI and the Fingerprinting method. The open source software AIMS achieves its major objectives and is available for use in future research projects as it is believed that with finer grained accuracy and additional time for refinement, the system proposed can assist in saving lives.

References

Abdelsalam, H. S. & Olariu, S. (2009*a*), A 3d-localization and terrain modeling technique for wireless sensor networks, *in* 'Proceedings of the 2Nd ACM International Workshop on Foundations of Wireless Ad Hoc and Sensor Networking and Computing', FOWANC '09, ACM, New York, NY, USA, pp. 37–46.
URL: *http://doi.acm.org/10.1145/1540343.1540351*

Abdelsalam, H. S. & Olariu, S. (2009*b*), Passive localization using rotating anchor pairs in wireless sensor networks, *in* 'Proceedings of the 2Nd ACM International Workshop on Foundations of Wireless Ad Hoc and Sensor Networking and Computing', FOWANC '09, ACM, New York, NY, USA, pp. 67–76.
URL: *http://doi.acm.org/10.1145/1540343.1540355*

Ahn, H. & Rhee, S.-B. (2010), Simulation of a rssi-based indoor localization system using wireless sensor network, *in* 'Ubiquitous Information Technologies and Applications (CUTE), 2010 Proceedings of the 5th International Conference on', pp. 1–4.

Angelopoulos, C. M., Filios, G., Karagiannis, M., Nikoletseas, S. & Rolim, J. (2012), Fine-grained in-door localization with wireless sensor networks, *in* 'Proceedings of the 10th ACM International Symposium on Mobility Management and Wireless Access', MobiWac '12, ACM, New York, NY, USA, pp. 159–162.
URL: *http://doi.acm.org/10.1145/2386995.2387025*

anon (2012), 'Wireless sensor networks (wsn) 2012-2022: Forecasts, technologies, players', `http://www.idtechex.com/research/reports/wireless-sensor-networks-wsn-2012-2022-forecasts-technologies-players-000314.asp?viewopt=desc`.

anon (2014), 'Water safety for children'.

URL: *http://raisingchildren.net.au/articles/safe_fun_with_water.html*

Azevedo, J. A. R. & Santos, F. E. (2007), 'Signal propagation measurements with wireless sensor nodes'.

Bahl, P. & Padmanabhan, V. (2000), Radar: an in-building rf-based user location and tracking system, *in* 'INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE', Vol. 2, pp. 775–784 vol.2.

Benkic, K., Malajner, M., Planinsic, P. & Cucej, Z. (2008), Using rssi value for distance estimation in wireless sensor networks based on zigbee, *in* 'Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on', pp. 303–306.

Boom, J., Ros, M., De-Hosson, G. & D'Souza, M. (2012), Indoor localisation using a context-aware dynamic position tracking model, *in* 'International Journal of Navigation and Observation', Vol. 2012, pp. 1–12.

Boukerche, A., Oliveira, H., Nakamura, E. & Loureiro, A. (2007), 'Localization systems for wireless sensor networks', *Wireless Communications, IEEE* **14**(6), 6–12.

Bras, L., Pinho, P. & Borges Carvaloh, N. (2013), 'Evaluation of a sectorised antenna in an indoor localisation system', *Microwaves, Antennas Propagation, IET* **7**(8), 679–685.

Bulusu, N., Heidemann, J. & Estrin, D. (2000), 'Gps-less low-cost outdoor localization for very small devices', *Personal Communications, IEEE* **7**(5), 28–34.

Chen, H., Liu, B., Huang, P., Liang, J. & Gu, Y. (2012), 'Mobility-assisted node localization based on toa measurements without time synchronization in wireless sensor networks', *Mob. Netw. Appl.* **17**(1), 90–99.
URL: *http://dx.doi.org/10.1007/s11036-010-0281-3*

Chen, M. J., Zhang, D. & LM, N. (2006), An empirical study of radio signal strength in sensor networks in using mica2 nodes, *in* 's', HKUST.

Chen, T., Yang, Z., Liu, Y., Guo, D. & Luo, X. (2014), 'Localization-oriented network adjustment in wireless ad hoc and sensor networks', *Parallel and Distributed Systems, IEEE Transactions on* **25**(1), 146–155.

Cherntanomwong, P. & Suroso, D. (2011), Indoor localization system using wireless sensor networks for stationary and moving target, *in* 'Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on', pp. 1–5.

Chi (2004), *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver.*
URL: *http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf*

Chu, C.-H., Wang, C.-H., Liang, C.-K., Ouyang, W., Cai, J.-H. & Chen, Y.-H. (2011), High-accuracy indoor personnel tracking system with a zigbee wireless sensor network, *in* 'Mobile Ad-hoc and Sensor Networks (MSN), 2011 Seventh International Conference on', pp. 398–402.

Chuenurajit, T., Phimmasean, S. & Cherntanomwong, P. (2013), Robustness of 3d indoor localization based on fingerprint technique in wireless sensor networks, *in* 'Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2013 10th International Conference on', pp. 1–6.

Codina, J. (1972), 'Alarm'. US Patent 3,636,544.
URL: *https://www.google.com.au/patents/US3636544*

Colombo, A., Fontanelli, D., Macii, D. & Palopoli, L. (2011), A wearable embedded inertial platform with wireless connectivity for indoor position tracking, *in* 'Instrumentation and Measurement Technology Conference (I2MTC), 2011 IEEE', pp. 1–6.

Coluccia, A. & Ricciato, F. (2010), On ml estimation for automatic rss-based indoor localization, *in* 'Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on', pp. 495–502.

Cro (2003), *Telosb Mote Platform.*
URL: *http://www.willow.co.uk/TelosB_Datasheet.pdf*

Denis, B., Raulefs, R., Fleury, B., Uguen, B., Amiot, N., de Celis, L., Dominguez, J., Koldsgaard, M., Laaraiedh, M., Noureddine, H., Staudinger, E. & Steinboeck, G. (2013), Cooperative and heterogeneous indoor localization experiments, *in* 'Communications Workshops (ICC), 2013 IEEE International Conference on', pp. 6–10.

Eccleston, D. (2014), 'Danger lurks in the driveway', *The Road Ahead* .

El Helou, M., Lahoud, S., Ibrahim, M. & Khawam, K. (2013), A hybrid approach for radio access technology selection in heterogeneous wireless networks, *in* 'Wireless Conference (EW), Proceedings of the 2013 19th European', pp. 1–6.

Engineers-Australia (2010), 'Code of ethics - engineers australia'.
URL: *http://www.engineersaustralia.org.au/ethics*

Fischer, C. & Gellersen, H. (2010), 'Location and navigation support for emergency responders: A survey', *Pervasive Computing, IEEE* **9**(1), 38–47.

Flood, J. (1998), 'Water-activated emergency radio beacon'. US Patent 5,710,989.
URL: *http://www.google.com.au/patents/US5710989*

Fonseca, R., Gnawali, O., Jamieson, K., Kim, S., Levis, P. & Woo, A. (2006), 'The collection tree protocol (ctp)'. TinyOS Enhancement Proposals, TEP123.
URL: *http://www.tinyos.net/tinyos-2.1.0/doc/html/tep123.html*

Gao, R., Zhou, H. & Su, G. (2011), Structure of wireless sensors network based on tinyos, *in* 'Control, Automation and Systems Engineering (CASE), 2011 International Conference on', pp. 1–4.

Giorgetti, G., Farley, R., Chikkappa, K., Ellis, J. & Kaleas, T. (2011), Cortina: Collaborative indoor positioning using low-power sensor networks, *in* 'Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on', pp. 1–10.

Glatz, P. M., Steger, C. & Weiss, R. (2010), Design, simulation and measurement of an accurate wireless sensor network localization system, *in* 'Proceedings of the 5th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks', PM2HW2N '10, ACM, New York, NY, USA, pp. 1–8.
URL: *http://doi.acm.org/10.1145/1868612.1868614*

Gwon, Y., Jain, R. & Kawahara, T. (2004), Robust indoor location estimation of stationary and mobile users, *in* 'INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies', Vol. 2, pp. 1032–1043 vol.2.

Haque, I., Nikolaidis, I. & Gburzynski, P. (2009), A scheme for indoor localization through rf profiling, *in* 'Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on', pp. 1–5.

Hatler, M., Gurganious, D. & Chi, C. (2013), 'Smart Building Wireless Sensor Networks', `http://onworld.com/smbldgs/index.html`.

Hatler, M., Gurganious, D. & Chi, C. (2014), '802.15.4 and Zigbee: Enabling the Internet of Things', `http://onworld.com/zigbee/`.

He, T., Huang, C., Blum, B. M., Stankovic, J. A. & Abdelzaher, T. (2003), Range-free localization schemes for large scale sensor networks, *in* 'Proceedings of the 9th Annual International Conference on Mobile Computing and Networking', MobiCom '03, ACM, New York, NY, USA, pp. 81–95.
URL: *http://doi.acm.org/10.1145/938985.938995*

Hightower, J., Vakili, C., Borriello, G. & Want, R. (2001), Design and calibration of the spoton ad-hoc location sensing system, Technical report, University of Washington.

Hill, J. L. (2003), System Architecture for Wireless Sensor Networks, PhD thesis, University of California, Berkely. AAI3105239.

Huang, C.-N. & Chan, C.-T. (2011), 'Zigbee-based indoor location system by k-nearest neighbor algorithm with weighted {RSSI}', *Procedia Computer Science* **5**(0), 58 – 65. The 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information Systems (MobiWIS 2011).
URL: *http://www.sciencedirect.com/science/article/pii/S1877050911003358*

Huang, Y.-F., Jheng, Y.-T. & Chen, H.-C. (2010), Performance of an mmse based indoor localization with wireless sensor networks, *in* 'Networked Computing and Advanced Information Management (NCM), 2010 Sixth International Conference on', pp. 671–675.

IEEE (2011), 'Ieee standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans)', *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)* pp. 1–314.

Kim, H., Park, C., Cho, Y., Shin, C., Park, J. & Park, D. (2012), A study on a ratiometric gps iteration algorithm for indoor localization, *in* 'Computing and Convergence Technology (ICCCT), 2012 7th International Conference on', pp. 32–35.

Kim, K. & Lee, W. (2007), Mbal: A mobile beacon-assisted localization scheme for wireless sensor networks, *in* 'Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on', pp. 57–62.

Krishna, M. B. & Doja, M. N. (2011), Computing methodologies for localization techniques in wireless sensor networks, *in* 'Proceedings of the International Conference &#38; Workshop on Emerging Trends in Technology', ICWET '11, ACM, New York, NY, USA, pp. 1024–1028.
URL: *http://doi.acm.org/10.1145/1980022.1980245*

Kung, H. T., Lin, C.-K., Lin, T.-H. & Vlah, D. (2009), Localization with snap-inducing shaped residuals (sisr): Coping with errors in measurement, *in* 'Proceedings of the 15th Annual International Conference on Mobile Computing and Networking', MobiCom '09, ACM, New York, NY, USA, pp. 333–344.
URL: *http://doi.acm.org/10.1145/1614320.1614357*

Levis, P. (2006), 'Tinyos programming'.

Li, Z., Trappe, W., Zhang, Y. & Nath, B. (2005), Robust statistical methods for securing wireless localization in sensor networks, *in* 'Proceedings of the 4th International Symposium on Information Processing in Sensor Networks', IPSN '05, IEEE Press, Piscataway, NJ, USA.
URL: *http://dl.acm.org/citation.cfm?id=1147685.1147703*

Lifesaving Society Australia, R. (2013), 'Royal Lifesaving Society Australia', *National Drowning Report 2013* .
URL: *http://www.royallifesaving.com.au/__data/assets/pdf_file/0003/9759/RLS_NationalDrowningReport_2013.pdf*

Liu, B. H., Otis, B., Challa, S., Axon, P., Chou, C. T. & Jha, S. (2006), On the fading and shadowing effects for wireless sensor networks, *in* 'Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference on', pp. 51–60.

Liu, H., Darabi, H., Banerjee, P. & Liu, J. (2007), 'Survey of wireless indoor positioning techniques and systems', *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* **37**(6), 1067–1080.

Lorincz, K., Malan, D., Fulford-Jones, T., Nawoj, A., Clavel, A., Shnayder, V., Mainland, G., Welsh, M. & Moulton, S. (2004), 'Sensor networks for emergency response: challenges and opportunities', *Pervasive Computing, IEEE* **3**(4), 16–23.

Lu, W. & Tan, Y.-P. (2004), 'A vision-based approach to early detection of drowning incidents in swimming pools', *Circuits and Systems for Video Technology, IEEE Transactions on* **14**(2), 159–178.

Mametani, T. (n.d.), 'Renesas Solutions for Wireless Sensor Networks - Part 1', `http://am.renesas.com/edge_ol/features/07/index.jsp`.

McGuire, M., Plataniotis, K. & Venetsanopoulos, A. (2003), 'Location of mobile terminals using time measurements and survey points', *Vehicular Technology, IEEE Transactions on* **52**(4), 999–1011.

Millen, T. (1983), 'Wave responsive swimming pool alarm'. US Patent 4,408,193.
URL: *http://www.google.com.au/patents/US4408193*

Miller, B., Halwachs, J. & Farstad, A. (1999), 'Swimmer location monitor'. US Patent 5,907,281.
URL: *https://www.google.com.au/patents/US5907281*

MoteIV (2004), 'Telos - ultra low power ieee 802.15.4 compliant wireless sensor module', *Telos Revision B - Datasheet* .

Nasipuri, A. & Li, K. (2002), A directionality based location discovery scheme for wireless sensor networks, *in* 'Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications', WSNA '02, ACM, New York, NY, USA, pp. 105–111.
URL: *http://doi.acm.org/10.1145/570738.570754*

Nazir, U., Arshad, M., Shahid, N. & Raza, S. (2012), Classification of localization algorithms for wireless sensor network: A survey, *in* 'Open Source Systems and Technologies (ICOSST), 2012 International Conference on', pp. 1–5.

Niculescu, D. & Nath, B. (2001), Ad hoc positioning system (aps), *in* 'Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE', Vol. 5, pp. 2926–2931 vol.5.

Niculescu, D. & Nath, B. (2003), 'Dv based positioning in ad hoc networks', *Telecommunication Systems* **22**(1-4), 267–280.

nms.lcs.mit.edu (n.d.), 'Patient-centric network'.
URL: *http://nms.lcs.mit.edu/projects/pcn/*

Noguchi, H., Fukada, H., Mori, T., Sanada, H. & Sato, T. (2013), 'Object and human localization with zigbee-based sensor devices in a living environment'.

Oracle (2014), 'Mysql c api - mysql 5.0 reference manual'.
URL: *http://dev.mysql.com/doc/refman/5.0/en/c-api.html*

Ouyang, R., Wong, A.-S. & Lea, C.-T. (2010), 'Received signal strength-based wireless localization via semidefinite programming: Noncooperative and cooperative schemes', *Vehicular Technology, IEEE Transactions on* **59**(3), 1307–1318.

OFlaherty, B. & ODonoghue, J. (2010), 'The lead-user method in practice', *The Development of Emerging Medical Devices* pp. 121–133.

Panwar, A. & Kumar, S. (2012), Localization schemes in wireless sensor networks, *in* 'Advanced Computing Communication Technologies (ACCT), 2012 Second International Conference on', pp. 443–449.

Patwari, N. & Hero, III, A. O. (2003), Using proximity and quantized rss for sensor localization in wireless networks, *in* 'Proceedings of the 2Nd ACM International Conference on Wireless Sensor Networks and Applications', WSNA '03, ACM, New York, NY, USA, pp. 20–29.
URL: *http://doi.acm.org/10.1145/941350.941354*

Priwgharm, R. & Chemtanomwong, P. (2011), A comparative study on indoor localization based on rssi measurement in wireless sensor network, *in* 'Computer Science and Software Engineering (JCSSE), 2011 Eighth International Joint Conference on', pp. 1–6.

Purvis, M. (2014), 'Zigbee and 802.15.4 to Drive 43 Percent of Wireless Smart Energy Solutions', http://www.onworld.com/news/ZigBee-802154-to-Drive-43-Percent-of-Smart-Energy.html.

Quinones, S. (1996), 'Baby pool guard alarm'. US Patent 5,486,814.
URL: *http://www.google.com.au/patents/US5486814*

Rudafshani, M. & Datta, S. (2007), Localization in wireless sensor networks, *in* 'Proceedings of the 6th International Conference on Information Processing in Sensor Networks', IPSN '07, ACM, New York, NY, USA, pp. 51–60.
URL: *http://doi.acm.org/10.1145/1236360.1236368*

Sahoo, P. K., Hwang, I.-S. & Lin, S.-Y. (2008), A distributed localization scheme for wireless sensor networks, *in* 'Proceedings of the International Conference on Mobile Technology, Applications, and Systems', Mobility '08, ACM, New York, NY, USA, pp. 77:1–77:7.
URL: *http://doi.acm.org/10.1145/1506270.1506366*

Savvides, A., Han, C.-C. & Strivastava, M. B. (2001), Dynamic fine-grained localization in ad-hoc networks of sensors, *in* 'Proceedings of the 7th annual international conference on Mobile computing and networking', ACM, pp. 166–179.

Schmid, J., Volker, M., Gadeke, T., Weber, P., Stork, W. & Muller-Glaser, K. (2010), An approach to infrastructure-independent person localization with an ieee 802.15.4 wsn, *in* 'Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on', pp. 1–9.

Sensium (2014), 'Overview - worlds leading wireless healthcare solutions'.
URL: *http://www.sensium-healthcare.com*

Shalini, R. (2012), 'Global Markets and Technologies for Wireless Sensors', http://www.bccresearch.com/market-research/instrumentation-and-sensors/wireless-sensors-technologies-markets-ias019a.html.

Shen, G., Yu, J. & Tan, L. (2012), Hierarchical rss-based indoor positioning using a markov random field model, *in* 'Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on', pp. 1–4.

Soleimanifar, M., Lu, M., Nikolaidis, I. & Lee, S. (2011), A robust positioning architecture for construction resources localization using wireless sensor networks, *in* 'Proceedings of the Winter Simulation Conference', WSC '11, Winter Simulation Conference, pp. 3562–3572.
URL: *http://dl.acm.org/citation.cfm?id=2431518.2431941*

Stoleru, R., He, T., Stankovic, J. A. & Luebke, D. (2005), A high-accuracy, low-cost localization system for wireless sensor networks, *in* 'Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems', SenSys '05, ACM, New York, NY, USA, pp. 13–26.
URL: *http://doi.acm.org/10.1145/1098918.1098921*

Sugano, M., Kawazoe, T., Ohta, Y. & Murata, M. (2006), 'Indoor localization system using rssi measurement of wireless sensor network based on zigbee standard', *Target* **538**, 050.

T V, S., Katti, A. K. & V S, A. (2006), Analysis of mobile beacon aided in-range localization scheme in ad hoc wireless sensor networks, *in* 'Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing', IWCMC '06, ACM, New York, NY, USA, pp. 1159–1164.
URL: *http://doi.acm.org/10.1145/1143549.1143782*

Tafa, Z. (n.d.), *Sensor Networks in Pervasive Healthcare Computing*, IGI Global.
URL: *http://www.igi-global.com/chapter/sensor-networks-pervasive-healthcare-computing/50654*

TinyOS (2013), 'Tinyos overview'.
URL: *http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyOS_Overview*

Torres-Solis, J., Falk, T. H. & Chau, T. (2010), 'A review of indoor localization technologies: towards navigational assistance for topographical disorientation, ambient intelligence'.
URL: *http://www.intechopen.com/books/ambient-intelligence/a-review-of-indoor-localization-technologies-towards-navigational-assistance-for-topographical-disor*

Tze-Meng, O. Goek, T. K. & Reza, A. W. (2010), A dual-band omni-directional microstrip antenna, *in* 'Progress In Electromagnetics Research, Vol. 106, 363376, 2010', Vol. 106, pp. 363–376.

Wang, J.-Y., Chen, C.-P., Lin, T.-S., Chuang, C.-L., Lai, T.-Y. & Jiang, J.-A. (2012), High-precision rssi-based indoor localization using a transmission power adjustment strategy for wireless sensor networks, *in* 'High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on', pp. 1634–1638.

Wearable Technology Market - Global Scenario, Trends, Industry Analysis, Size, Share And Forecast 2012 - 2018 (2013), "Transparency - Market Research".
URL: *http://www.wearabletechworld.com/topics/from-the-experts/*

articles/323855-wearable-technology-next-mobility-market-booming.htm

Werb, J. & Lanzl, C. (1998), 'Designing a positioning system for finding things and people indoors', *Spectrum, IEEE* **35**(9), 71–78.

Wu, J., Cai, J., Huo, M., Wu, M. & Zhou, J. (2013), 'A wearable early monitoring and alarming device for swimming pool drowning incidents', *Journal of Computational Information Systems* **9**(21), 8619–8627.

www.alz.org (2014), 'Alzheimers - wandering and getting lost'.
URL: *http://www.alz.org/care/alzheimers-dementia-wandering.asp*

www.researchmoz.us (2014), 'Semiconductor Wireless Sensor Internet of Things (iot) Market Analysis 2014 - industry size, shares, growth, trends and forecast research report 2020', http://www.researchmoz.us/semiconductor-wireless-sensor-internet-of-things-iot-market-shares-strategies-and-forecasts-worldwide-2014-2020-report.html.

Xiaoyang, C., Wenkai, C. & Fei, L. (2007), Application of image restoration based on robust estimation in drowning warning system, *in* 'Digital Media and its Application in Museum Heritages, Second Workshop on', pp. 33–35.

Xie, Z., Hong, M., Liu, H., Li, J., Zhu, K. & Stankovic, J. (2011), Quantitative uncertainty-based incremental localization and anchor selection in wireless sensor networks, *in* 'Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems', MSWiM '11, ACM, New York, NY, USA, pp. 417–426.
URL: *http://doi.acm.org/10.1145/2068897.2068968*

Yedavalli, K., Krishnamachari, B. & Venkatraman, L. (2007), 'Fast/fair mobile localization in infrastructure wireless sensor networks', *SIGMOBILE Mob. Comput. Commun. Rev.* **11**(1), 29–40.
URL: *http://doi.acm.org/10.1145/1234822.1234828*

Zhang, D., Xia, F., Yang, Z., Yao, L. & Zhao, W. (2010), Localization technologies for indoor human tracking, *in* 'Future Information Technology (FutureTech), 2010 5th International Conference on', pp. 1–6.

Zhang, S. & Xing, T. (2013), Open wsn indoor localization platform design, *in* 'Instrumentation and Measurement, Sensor Network and Automation (IMSNA), 2013 2nd International Symposium on', pp. 845–848.

Zhang, X., Banavar, M., Willerton, M., Manikas, A., Tepedelenlioglu, C., Spanias, A., Thornton, T., Yeatman, E. & Constantinides, A. (2012), Performance comparison of localization techniques for sequential wsn discovery, *in* 'Sensor Signal Processing for Defence (SSPD 2012)', pp. 1–5.

Zhao, F. & Guibas, L. J. (2004), *Wireless Sensor Networks - An information processing approach*, Morgan Kaufmann.

Zhao, J. & Jia, H. (2010), A hybrid localization algorithm based on dv-distance and the twice-weighted centroid for wsn, *in* 'Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on', Vol. 7, pp. 590–594.

Zhao, Y. & Smith, J. R. (2013), A battery-free rfid-based indoor acoustic localization platform, *in* 'IEEE RFID 2013'.

Zhong, Z. & He, T. (2009), Achieving range-free localization beyond connectivity, *in* 'Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems', SenSys '09, ACM, New York, NY, USA, pp. 281–294.
URL: *http://doi.acm.org/10.1145/1644038.1644066*

Zhou, Q., Jin, D., Zeng, L. & Zhou, Y. (2008), Area concentric beacons localization for wireless sensor networks, *in* 'Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE', pp. 2129–2134.

Appendix A

Project Specifications

ENG 4111/2 Research Project

Project Specification

For:	**Brad Goold**
Topic:	An innovative WSN-based intelligent system for Home Safety and Health Monitoring
Supervisor:	H. Zhou
Project Aim:	This project aims to research, design and implement an out-of-the box solution for object localisation in homes or residential buildings. It is proposed that we deliver a solution whereby the user requires minimal set-up time and technical knowledge in order to successfully interact with the WSN in the home.

Program:

1. A thorough survey on the market & similar products

2. Literature review on wireless sensing networks in homes. Including, but not limited to: current technology, needs and requirements, and social acceptance/ethical issues.

3. Additional research into what technologies are best suited for the application in regards to cost, effectiveness and ease of use/installation.

4. Study of background/foundation knowledge of technology and tools (eg. WSN/Zigbee, NesC, TinyOS, Crossbow Motes). The type of sensor, programming platform, communication protocol and operating system are dependent on the above research.

5. Create an overview of the scope of the task, its merits, constraints, challenges and finer details in an appropriately structured 'Project Appreciation Report'.

6. Design, test and evaluate the solution.

   Harware Design

   Software Design (Sensor + network coding + basic IU)

   Prototyping.

   Testing - I.e. Total design testing which includes, but is not limited to: HW, SW, adaptability and ease of use.

   Implement changes as required and reiterate the design loop in order to optimise the design.

7. Produce a formal disseratation.

As time and resources permit:

1. Program an interactive User Interface for the user PC.

2. Outline a basic design for the total product solution:

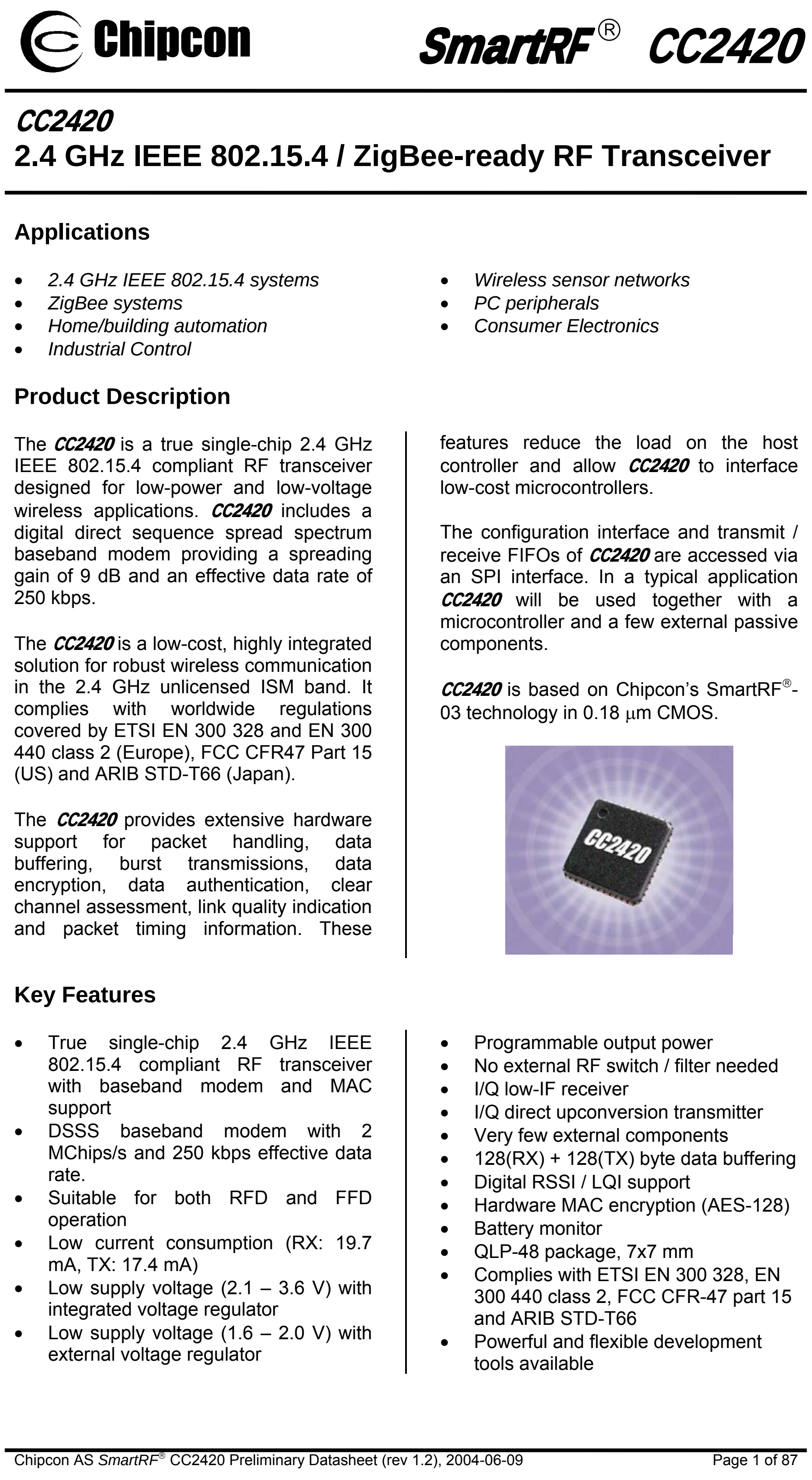
   Out of the box deliverables to potential customers. Instructions, software, installation and setup details, constraints etc.

Agreed:

Student Name:	Brad Goold
Date:	31st March 2014
Supervisor Name:	Dr. Hong Zhou
Date:	31st March 2014
Examiner:	
Date:	

Appendix B

Datasheets



Chipcon SmartRF® CC2420

CC2420

2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver

Applications

- *2.4 GHz IEEE 802.15.4 systems*
- *ZigBee systems*
- *Home/building automation*
- *Industrial Control*
- *Wireless sensor networks*
- *PC peripherals*
- *Consumer Electronics*

Product Description

The *CC2420* is a true single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver designed for low-power and low-voltage wireless applications. *CC2420* includes a digital direct sequence spread spectrum baseband modem providing a spreading gain of 9 dB and an effective data rate of 250 kbps.

The *CC2420* is a low-cost, highly integrated solution for robust wireless communication in the 2.4 GHz unlicensed ISM band. It complies with worldwide regulations covered by ETSI EN 300 328 and EN 300 440 class 2 (Europe), FCC CFR47 Part 15 (US) and ARIB STD-T66 (Japan).

The *CC2420* provides extensive hardware support for packet handling, data buffering, burst transmissions, data encryption, data authentication, clear channel assessment, link quality indication and packet timing information. These features reduce the load on the host controller and allow *CC2420* to interface low-cost microcontrollers.

The configuration interface and transmit / receive FIFOs of *CC2420* are accessed via an SPI interface. In a typical application *CC2420* will be used together with a microcontroller and a few external passive components.

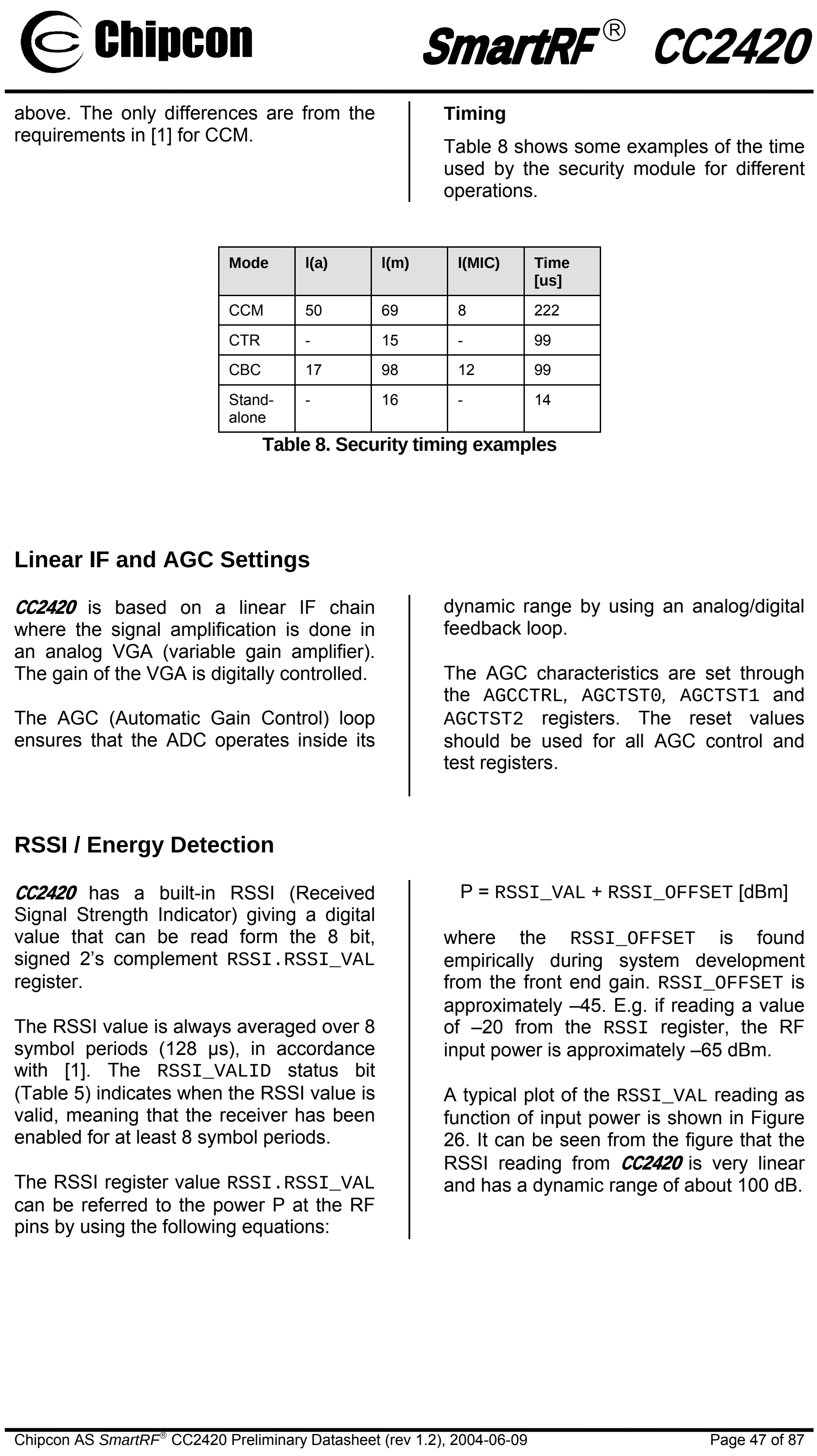
CC2420 is based on Chipcon's SmartRF®-03 technology in 0.18 μm CMOS.

Key Features

- True single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver with baseband modem and MAC support
- DSSS baseband modem with 2 MChips/s and 250 kbps effective data rate.
- Suitable for both RFD and FFD operation
- Low current consumption (RX: 19.7 mA, TX: 17.4 mA)
- Low supply voltage (2.1 – 3.6 V) with integrated voltage regulator
- Low supply voltage (1.6 – 2.0 V) with external voltage regulator
- Programmable output power
- No external RF switch / filter needed
- I/Q low-IF receiver
- I/Q direct upconversion transmitter
- Very few external components
- 128(RX) + 128(TX) byte data buffering
- Digital RSSI / LQI support
- Hardware MAC encryption (AES-128)
- Battery monitor
- QLP-48 package, 7x7 mm
- Complies with ETSI EN 300 328, EN 300 440 class 2, FCC CFR-47 part 15 and ARIB STD-T66
- Powerful and flexible development tools available

Chipcon AS *SmartRF®* CC2420 Preliminary Datasheet (rev 1.2), 2004-06-09 Page 1 of 87

Figure B.1: CC2420 Datasheet (Chi 2004)



Chipcon SmartRF® CC2420

above. The only differences are from the requirements in [1] for CCM.

Timing

Table 8 shows some examples of the time used by the security module for different operations.

Mode	l(a)	l(m)	l(MIC)	Time [us]
CCM	50	69	8	222
CTR	-	15	-	99
CBC	17	98	12	99
Stand-alone	-	16	-	14

Table 8. Security timing examples

Linear IF and AGC Settings

CC2420 is based on a linear IF chain where the signal amplification is done in an analog VGA (variable gain amplifier). The gain of the VGA is digitally controlled.

The AGC (Automatic Gain Control) loop ensures that the ADC operates inside its dynamic range by using an analog/digital feedback loop.

The AGC characteristics are set through the `AGCCTRL`, `AGCTST0`, `AGCTST1` and `AGCTST2` registers. The reset values should be used for all AGC control and test registers.

RSSI / Energy Detection

CC2420 has a built-in RSSI (Received Signal Strength Indicator) giving a digital value that can be read form the 8 bit, signed 2's complement `RSSI.RSSI_VAL` register.

The RSSI value is always averaged over 8 symbol periods (128 μs), in accordance with [1]. The `RSSI_VALID` status bit (Table 5) indicates when the RSSI value is valid, meaning that the receiver has been enabled for at least 8 symbol periods.

The RSSI register value `RSSI.RSSI_VAL` can be referred to the power P at the RF pins by using the following equations:

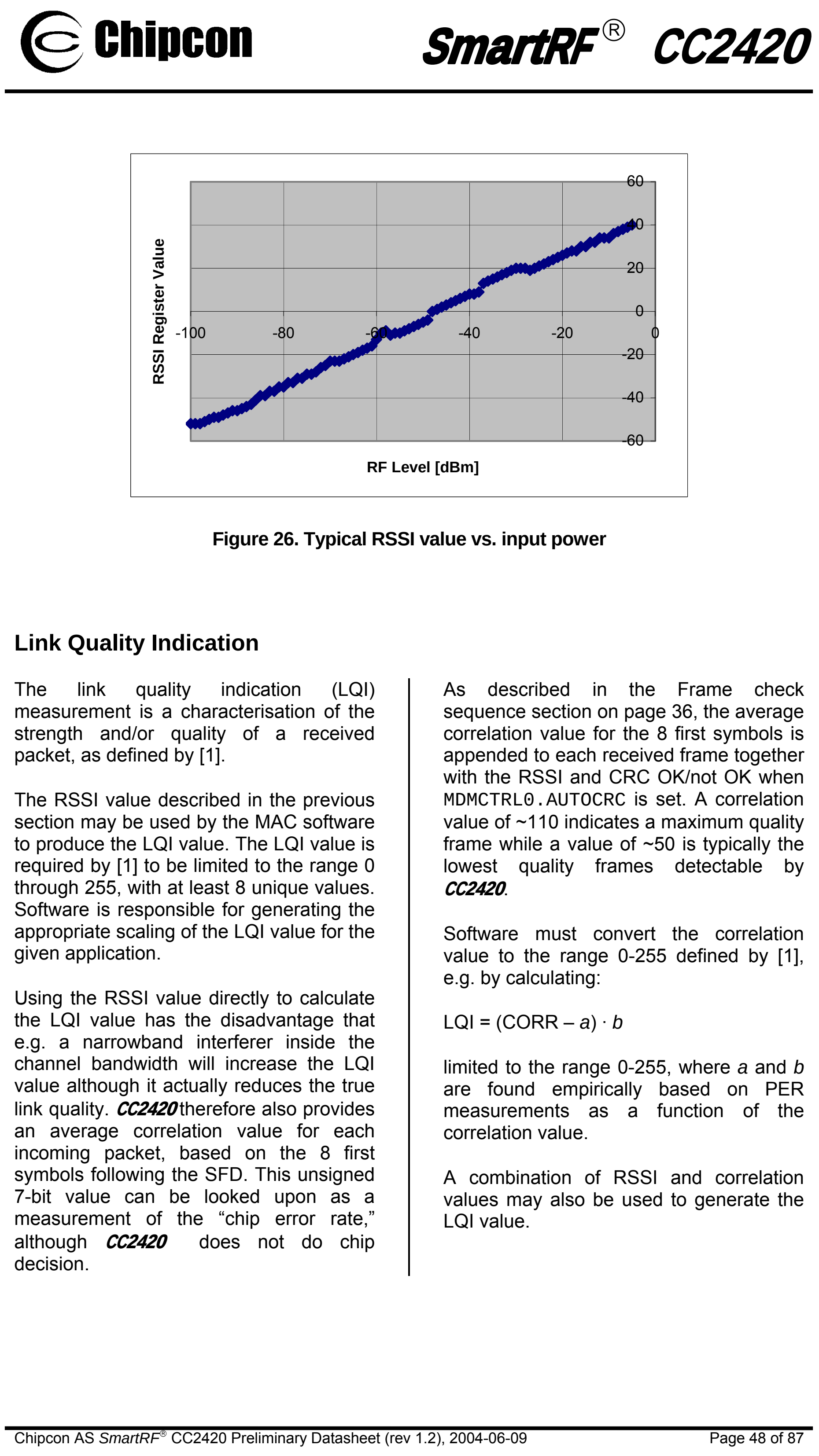
P = `RSSI_VAL` + `RSSI_OFFSET` [dBm]

where the `RSSI_OFFSET` is found empirically during system development from the front end gain. `RSSI_OFFSET` is approximately –45. E.g. if reading a value of –20 from the `RSSI` register, the RF input power is approximately –65 dBm.

A typical plot of the `RSSI_VAL` reading as function of input power is shown in Figure 26. It can be seen from the figure that the RSSI reading from ***CC2420*** is very linear and has a dynamic range of about 100 dB.

Chipcon AS *SmartRF*® CC2420 Preliminary Datasheet (rev 1.2), 2004-06-09 Page 47 of 87

Figure B.2: CC2420 Datasheet (Chi 2004)



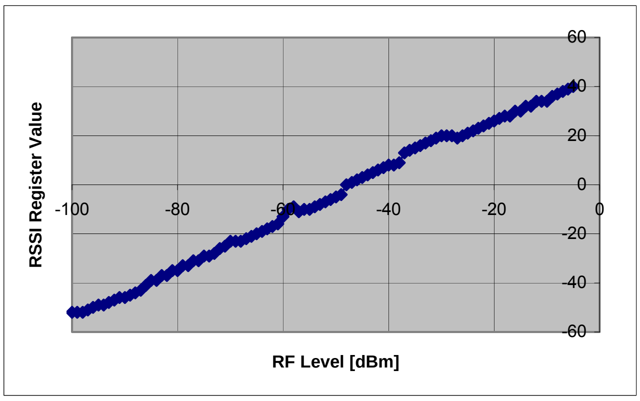


Figure 26. Typical RSSI value vs. input power

Link Quality Indication

The link quality indication (LQI) measurement is a characterisation of the strength and/or quality of a received packet, as defined by [1].

The RSSI value described in the previous section may be used by the MAC software to produce the LQI value. The LQI value is required by [1] to be limited to the range 0 through 255, with at least 8 unique values. Software is responsible for generating the appropriate scaling of the LQI value for the given application.

Using the RSSI value directly to calculate the LQI value has the disadvantage that e.g. a narrowband interferer inside the channel bandwidth will increase the LQI value although it actually reduces the true link quality. ***CC2420*** therefore also provides an average correlation value for each incoming packet, based on the 8 first symbols following the SFD. This unsigned 7-bit value can be looked upon as a measurement of the "chip error rate," although ***CC2420*** does not do chip decision.

As described in the Frame check sequence section on page 36, the average correlation value for the 8 first symbols is appended to each received frame together with the RSSI and CRC OK/not OK when `MDMCTRL0.AUTOCRC` is set. A correlation value of ~110 indicates a maximum quality frame while a value of ~50 is typically the lowest quality frames detectable by ***CC2420***.

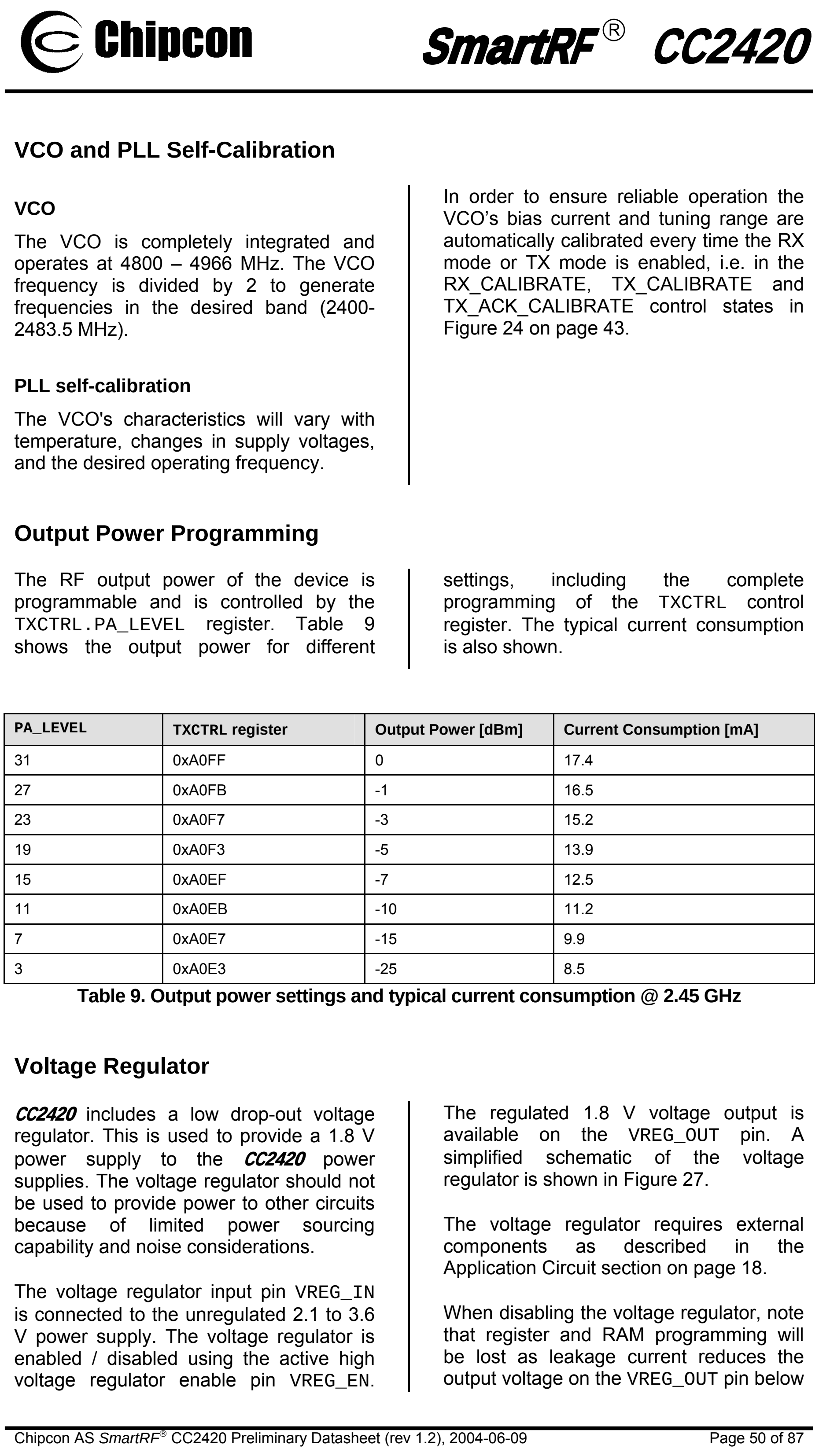
Software must convert the correlation value to the range 0-255 defined by [1], e.g. by calculating:

LQI = (CORR – a) · b

limited to the range 0-255, where a and b are found empirically based on PER measurements as a function of the correlation value.

A combination of RSSI and correlation values may also be used to generate the LQI value.

Figure B.3: CC2420 Datasheet (Chi 2004)



Chipcon SmartRF® CC2420

VCO and PLL Self-Calibration

VCO

The VCO is completely integrated and operates at 4800 – 4966 MHz. The VCO frequency is divided by 2 to generate frequencies in the desired band (2400-2483.5 MHz).

PLL self-calibration

The VCO's characteristics will vary with temperature, changes in supply voltages, and the desired operating frequency.

In order to ensure reliable operation the VCO's bias current and tuning range are automatically calibrated every time the RX mode or TX mode is enabled, i.e. in the RX_CALIBRATE, TX_CALIBRATE and TX_ACK_CALIBRATE control states in Figure 24 on page 43.

Output Power Programming

The RF output power of the device is programmable and is controlled by the `TXCTRL.PA_LEVEL` register. Table 9 shows the output power for different settings, including the complete programming of the `TXCTRL` control register. The typical current consumption is also shown.

PA_LEVEL	TXCTRL register	Output Power [dBm]	Current Consumption [mA]
31	0xA0FF	0	17.4
27	0xA0FB	-1	16.5
23	0xA0F7	-3	15.2
19	0xA0F3	-5	13.9
15	0xA0EF	-7	12.5
11	0xA0EB	-10	11.2
7	0xA0E7	-15	9.9
3	0xA0E3	-25	8.5

Table 9. Output power settings and typical current consumption @ 2.45 GHz

Voltage Regulator

CC2420 includes a low drop-out voltage regulator. This is used to provide a 1.8 V power supply to the ***CC2420*** power supplies. The voltage regulator should not be used to provide power to other circuits because of limited power sourcing capability and noise considerations.

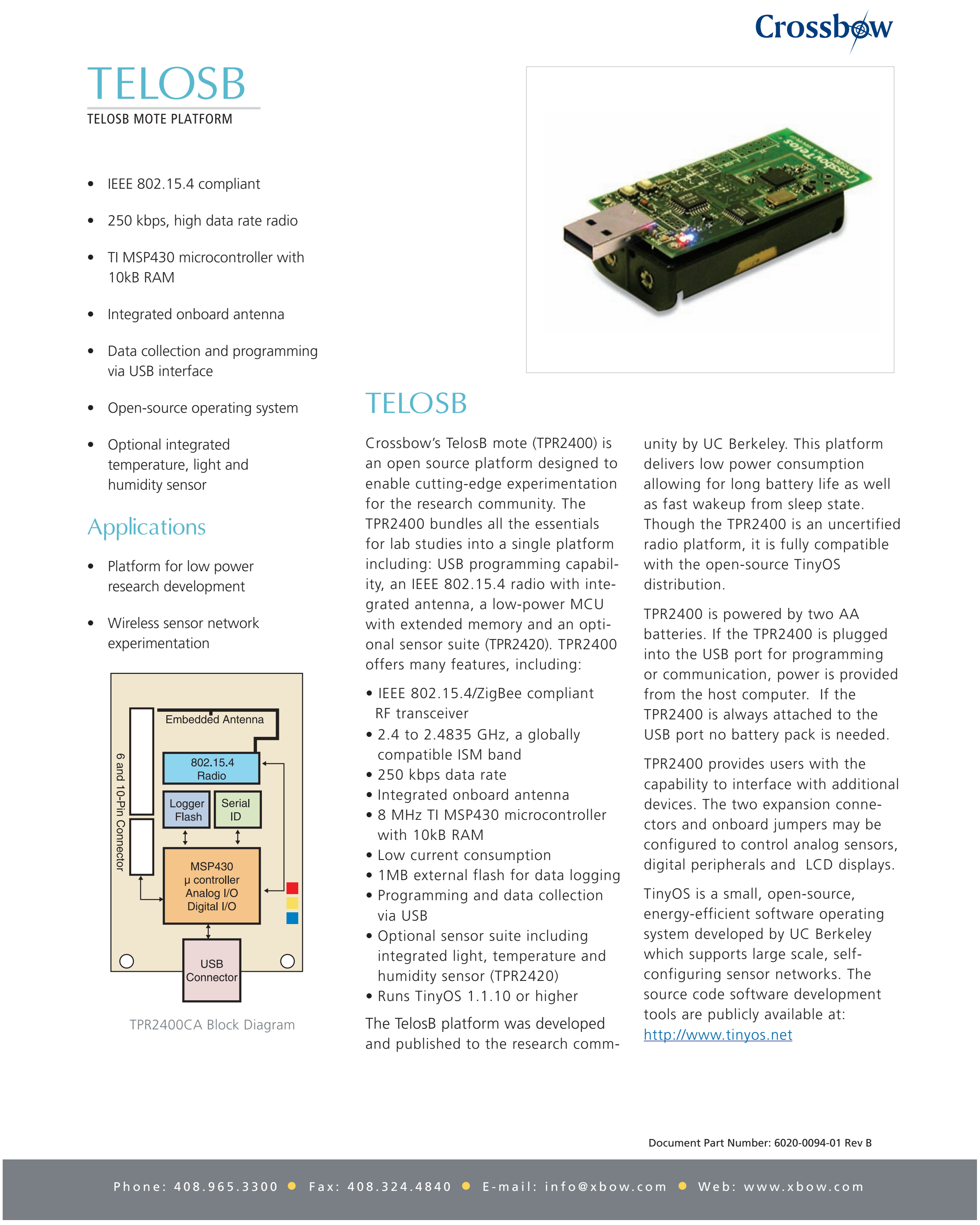
The voltage regulator input pin `VREG_IN` is connected to the unregulated 2.1 to 3.6 V power supply. The voltage regulator is enabled / disabled using the active high voltage regulator enable pin `VREG_EN`. The regulated 1.8 V voltage output is available on the `VREG_OUT` pin. A simplified schematic of the voltage regulator is shown in Figure 27.

The voltage regulator requires external components as described in the Application Circuit section on page 18.

When disabling the voltage regulator, note that register and RAM programming will be lost as leakage current reduces the output voltage on the `VREG_OUT` pin below

Chipcon AS *SmartRF*® CC2420 Preliminary Datasheet (rev 1.2), 2004-06-09 Page 50 of 87

Figure B.4: CC2420 Datasheet (Chi 2004)



Crossbow

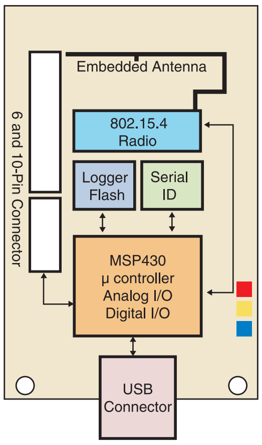
TELOSB

TELOSB MOTE PLATFORM

- IEEE 802.15.4 compliant
- 250 kbps, high data rate radio
- TI MSP430 microcontroller with 10kB RAM
- Integrated onboard antenna
- Data collection and programming via USB interface
- Open-source operating system
- Optional integrated temperature, light and humidity sensor

Applications

- Platform for low power research development
- Wireless sensor network experimentation



TPR2400CA Block Diagram

TELOSB

Crossbow's TelosB mote (TPR2400) is an open source platform designed to enable cutting-edge experimentation for the research community. The TPR2400 bundles all the essentials for lab studies into a single platform including: USB programming capability, an IEEE 802.15.4 radio with integrated antenna, a low-power MCU with extended memory and an optional sensor suite (TPR2420). TPR2400 offers many features, including:

- IEEE 802.15.4/ZigBee compliant RF transceiver
- 2.4 to 2.4835 GHz, a globally compatible ISM band
- 250 kbps data rate
- Integrated onboard antenna
- 8 MHz TI MSP430 microcontroller with 10kB RAM
- Low current consumption
- 1MB external flash for data logging
- Programming and data collection via USB
- Optional sensor suite including integrated light, temperature and humidity sensor (TPR2420)
- Runs TinyOS 1.1.10 or higher

The TelosB platform was developed and published to the research community by UC Berkeley. This platform delivers low power consumption allowing for long battery life as well as fast wakeup from sleep state. Though the TPR2400 is an uncertified radio platform, it is fully compatible with the open-source TinyOS distribution.

TPR2400 is powered by two AA batteries. If the TPR2400 is plugged into the USB port for programming or communication, power is provided from the host computer. If the TPR2400 is always attached to the USB port no battery pack is needed.

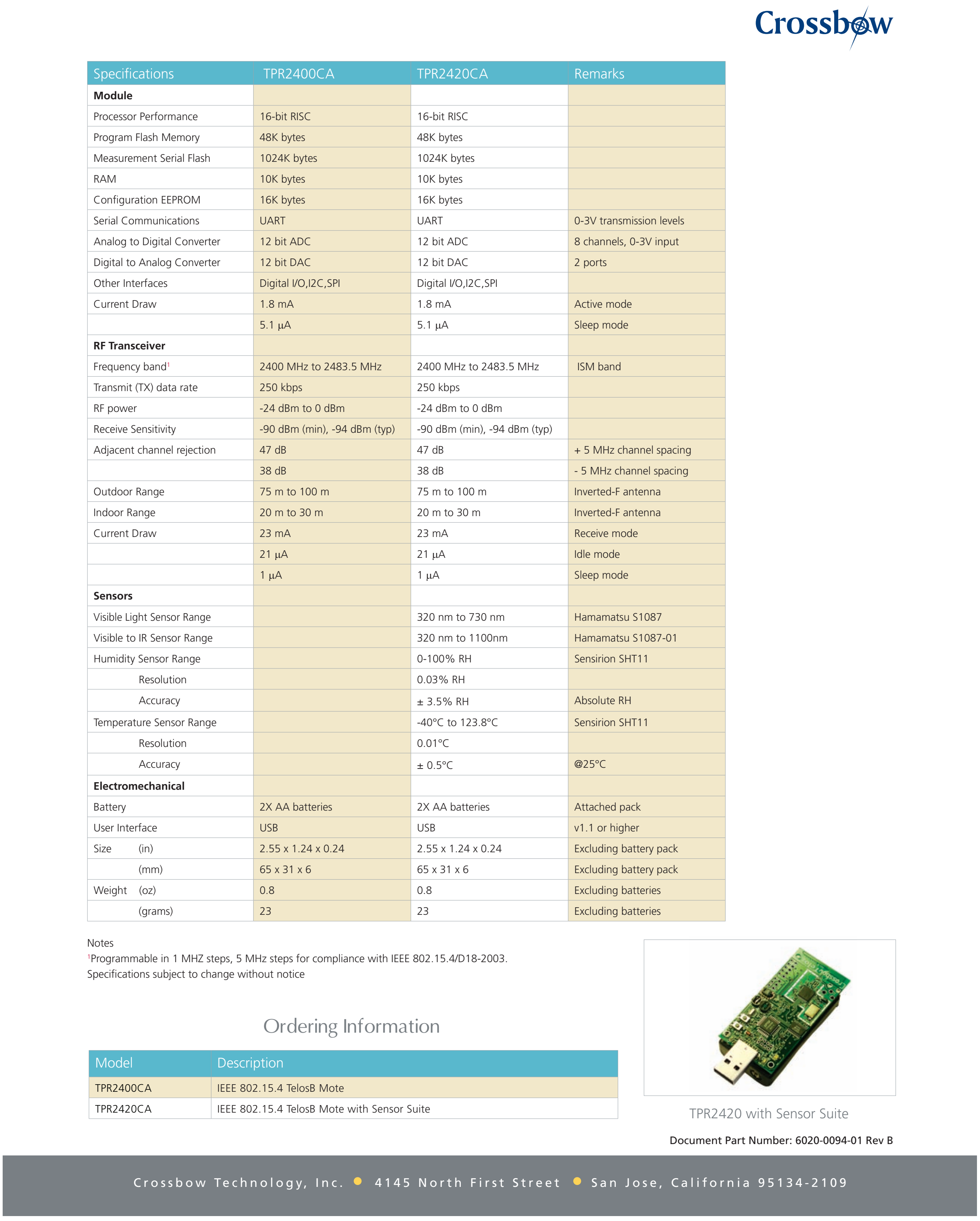
TPR2400 provides users with the capability to interface with additional devices. The two expansion connectors and onboard jumpers may be configured to control analog sensors, digital peripherals and LCD displays.

TinyOS is a small, open-source, energy-efficient software operating system developed by UC Berkeley which supports large scale, self-configuring sensor networks. The source code software development tools are publicly available at: http://www.tinyos.net

Document Part Number: 6020-0094-01 Rev B

Phone: 408.965.3300 ● Fax: 408.324.4840 ● E-mail: info@xbow.com ● Web: www.xbow.com

Figure B.5: Crossbow TelosB Datasheet (Cro 2003)



Crossbow

Specifications	TPR2400CA	TPR2420CA	Remarks
Module			
Processor Performance	16-bit RISC	16-bit RISC	
Program Flash Memory	48K bytes	48K bytes	
Measurement Serial Flash	1024K bytes	1024K bytes	
RAM	10K bytes	10K bytes	
Configuration EEPROM	16K bytes	16K bytes	
Serial Communications	UART	UART	0-3V transmission levels
Analog to Digital Converter	12 bit ADC	12 bit ADC	8 channels, 0-3V input
Digital to Analog Converter	12 bit DAC	12 bit DAC	2 ports
Other Interfaces	Digital I/O,I2C,SPI	Digital I/O,I2C,SPI	
Current Draw	1.8 mA	1.8 mA	Active mode
	5.1 µA	5.1 µA	Sleep mode
RF Transceiver			
Frequency band[1]	2400 MHz to 2483.5 MHz	2400 MHz to 2483.5 MHz	ISM band
Transmit (TX) data rate	250 kbps	250 kbps	
RF power	-24 dBm to 0 dBm	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	-90 dBm (min), -94 dBm (typ)	
Adjacent channel rejection	47 dB	47 dB	+ 5 MHz channel spacing
	38 dB	38 dB	- 5 MHz channel spacing
Outdoor Range	75 m to 100 m	75 m to 100 m	Inverted-F antenna
Indoor Range	20 m to 30 m	20 m to 30 m	Inverted-F antenna
Current Draw	23 mA	23 mA	Receive mode
	21 µA	21 µA	Idle mode
	1 µA	1 µA	Sleep mode
Sensors			
Visible Light Sensor Range		320 nm to 730 nm	Hamamatsu S1087
Visible to IR Sensor Range		320 nm to 1100nm	Hamamatsu S1087-01
Humidity Sensor Range		0-100% RH	Sensirion SHT11
Resolution		0.03% RH	
Accuracy		± 3.5% RH	Absolute RH
Temperature Sensor Range		-40°C to 123.8°C	Sensirion SHT11
Resolution		0.01°C	
Accuracy		± 0.5°C	@25°C
Electromechanical			
Battery	2X AA batteries	2X AA batteries	Attached pack
User Interface	USB	USB	v1.1 or higher
Size (in)	2.55 x 1.24 x 0.24	2.55 x 1.24 x 0.24	Excluding battery pack
(mm)	65 x 31 x 6	65 x 31 x 6	Excluding battery pack
Weight (oz)	0.8	0.8	Excluding batteries
(grams)	23	23	Excluding batteries

Notes
[1]Programmable in 1 MHZ steps, 5 MHz steps for compliance with IEEE 802.15.4/D18-2003.
Specifications subject to change without notice

Ordering Information

Model	Description
TPR2400CA	IEEE 802.15.4 TelosB Mote
TPR2420CA	IEEE 802.15.4 TelosB Mote with Sensor Suite

TPR2420 with Sensor Suite

Document Part Number: 6020-0094-01 Rev B

Crossbow Technology, Inc. ● 4145 North First Street ● San Jose, California 95134-2109

Figure B.6: Crossbow TelosB Datasheet (Cro 2003)

Appendix C

Code Listings

C.1 TelosB mote code

C.1.1 AIMSAppC.nc

Listing C.1: Aims Main Application Configuration File

```
#define CC2420_DEF_RFPOWER 20
#define LOW_POWER_LISTENING 1
#define CC2420_CHANNEL 20
configuration AIMSAppC { }
implementation {

    components MainC, AIMSC, LedsC, new TimerMilliC();

    AIMSC.Boot -> MainC;
    AIMSC.Timer -> TimerMilliC; //time after receiving beacon
    AIMSC.Leds -> LedsC; //indication of message sending/receipt

    // CTP Components
    components CollectionC as Collector,     // Collection layer
        ActiveMessageC,                       // AM layer
        new CollectionSenderC(AM_AIMS), // Sends multihop RF
        SerialActiveMessageC,                 // Serial messaging
        new SerialAMSenderC(AM_AIMS),    // Sends to the serial port
        CC2420ActiveMessageC; // to get RSSI value

    // Beacon Components
    components new TimerMilliC() as Timer0; // period of beacons
    components ActiveMessageC as BActiveMessageC;
    components new AMSenderC(AM_AIMSBEACONMSG);
    components new AMReceiverC(AM_AIMSBEACONMSG);

    // CTP Wiring
    //=========================
    AIMSC.CC2420Packet -> CC2420ActiveMessageC.CC2420Packet;
    AIMSC.RadioControl -> ActiveMessageC; //Radio
    AIMSC.SerialControl -> SerialActiveMessageC; // for base station
    AIMSC.RoutingControl -> Collector; // CTP

    AIMSC.Send -> CollectionSenderC;
    AIMSC.SerialSend -> SerialAMSenderC.AMSend;
    AIMSC.Snoop -> Collector.Snoop[AM_AIMS];
    AIMSC.Receive -> Collector.Receive[AM_AIMS];
    AIMSC.RootControl -> Collector;

    // Queue for basestation to send through UART
    //===============================================
    components new PoolC(message_t, 10) as UARTMessagePoolP,
        new QueueC(message_t*, 10) as UARTQueueP;
    AIMSC.UARTMessagePool -> UARTMessagePoolP;
    AIMSC.UARTQueue -> UARTQueueP;

    // Target Sensor Beacon Wiring
    //=============================
    AIMSC.BTimer0 -> Timer0;
    AIMSC.BPacket -> AMSenderC;
    AIMSC.BAMPacket -> AMSenderC;
    AIMSC.BAMSend -> AMSenderC;
    AIMSC.BAMControl -> BActiveMessageC;
    AIMSC.BReceive -> AMReceiverC;

}
```

C.1.2 AIMSC.nc

Listing C.2: Aims Main Application Implementation

```
#include "Timer.h"
#include "AIMS.h"
#include "AIMS_BEACON.h"

module AIMSC @safe(){
  uses {
    // Interfaces for initialization:
    interface Boot;
    interface SplitControl as RadioControl;
    interface SplitControl as SerialControl;
    interface StdControl as RoutingControl;

    // Interfaces for communication, multihop and serial:
    interface Send;
    interface Receive as Snoop;
    interface Receive;
    interface AMSend as SerialSend;
    interface CollectionPacket;
    interface RootControl;

    // Interfaces for serial comms
    interface CC2420Packet;
    interface Queue<message_t *> as UARTQueue;
    interface Pool<message_t> as UARTMessagePool;

    // Timers and read interfaces etc
    interface Timer<TMilli>;
    interface Leds;

    // Beacon interfaces
    interface Timer<TMilli> as BTimer0;
    interface Packet as BPacket;
    interface AMPacket as BAMPacket;
    interface AMSend as BAMSend;
    interface SplitControl as BAMControl;
    interface Receive as BReceive;
  }
}

implementation {
    task void uartSendTask(); //Send received packets over UART
    static void startTimer(); //Time delay after reciept of CTP msg
    // LEDs Functions
    static void fatal_problem();
    static void report_problem();
    static void report_sent();
    static void report_received();

    uint8_t uartlen; //length of uartbuf
    //buffers
    message_t sendbuf;
    message_t uartbuf;
    //flags
    bool sendbusy=FALSE, uartbusy=FALSE;

    /* Current local state - local CTP structure*/
    aims_ctp_t local;

    // BOOTED
    // ======
    event void Boot.booted() {
        // set local state for this node
        local.interval = DEFAULT_INTERVAL;
        local.id = TOS_NODE_ID;

        // Start CTP - node 8 = TARGET and doesnt use CTP
        if (TOS_NODE_ID != 8){
             if (call RadioControl.start() != SUCCESS)
            fatal_problem();

            if (call RoutingControl.start() != SUCCESS)
                fatal_problem();
        }
        // Target node
        //============
        if (TOS_NODE_ID == 8){
            call BTimer0.startPeriodic(TIMER_PERIOD_MILLI);
        }
        // Base Station doesnt receive beacons
        //====================================
        if (TOS_NODE_ID != 0){
            call        BAMControl.start();
        }
        else
        {
            call BAMControl.stop();
        }
    }

    // Radio Control has started
    //==========================
    event void RadioControl.startDone(error_t error) {
        if (error != SUCCESS)
            fatal_problem();

        // check the size of our message will fit the radio max
        // payload length
        if (sizeof(local) > call Send.maxPayloadLength())
            fatal_problem();

        // base station uses the serial
        if (TOS_NODE_ID == 0){
            if (call SerialControl.start() != SUCCESS)
                fatal_problem();
        }
    }

    // Serial control is started
    //==============================
    event void SerialControl.startDone(error_t error) {
    if (error != SUCCESS)
        fatal_problem();

    // If we are the base station, set ourselves as root
    if (local.id == 0)
      call RootControl.setRoot();

    }

    // Start one shot timer when beacon is received
    //==================================================
  static void startTimer() {
    if (call Timer.isRunning()) call Timer.stop();

    call Timer.startOneShot(local.interval);
  }

    // Dummy functions to handle unused events
  event void RadioControl.stopDone(error_t error) { }
  event void SerialControl.stopDone(error_t error) { }

    // Root event - receive messages on CTP
    //==========================================
    event message_t*
    Receive.receive(message_t* msg, void *payload, uint8_t len) {
        // root only receives messages in CTP
        aims_ctp_t* in = (aims_ctp_t*)payload;
        aims_ctp_t* out;

        // send to serial
        if (uartbusy == FALSE) {
            out = (aims_ctp_t*)call SerialSend.getPayload(&uartbuf,
                ↪ sizeof(aims_ctp_t));
            // check this is our message
            if (len != sizeof(aims_ctp_t) || out == NULL) {
                return msg; //leave if not
            }
            else {
                memcpy(out, in, sizeof(aims_ctp_t)); // copy in to
                    ↪ out
            }
            uartlen = sizeof(aims_ctp_t);
            post uartSendTask(); // call the uart task with the
                                 // message saved in 'out'
        } else {
            // The UART is busy. Put the messages in a queue
            message_t *newmsg = call UARTMessagePool.get();
            if (newmsg == NULL) {
                // no queue space.. drop it.
                report_problem();
                return msg;
            }

            //Serial port busy, so enqueue.
            out = (aims_ctp_t*)call SerialSend.getPayload(newmsg,
                ↪ sizeof(aims_ctp_t));
            if (out == NULL) {
                return msg;
            }
            memcpy(out, in, sizeof(aims_ctp_t));

            if (call UARTQueue.enqueue(newmsg) != SUCCESS) {
                //no queue space, drop.
                call UARTMessagePool.put(newmsg);
                fatal_problem();
                return msg;
            }
        }
         report_received();
        return msg;
    }

    // Send message via UART
    // task called by the scheduler. posted by Receive.receive
    //==========================================================
    task void uartSendTask() {
        if (call SerialSend.send(0xffff, &uartbuf, uartlen) !=
            ↪ SUCCESS) {
            report_problem();
        } else {
            uartbusy = TRUE;
        }
    }

    // Serial message has been sent
    //===============================
    event void SerialSend.sendDone(message_t *msg, error_t error) {
        uartbusy = FALSE;
        if (call UARTQueue.empty() == FALSE) {
            //Theres still messages in the uart queue. Repost for
            //sending those messages
            message_t *queuemsg = call UARTQueue.dequeue();
            if (queuemsg == NULL) {
                fatal_problem();
                return;
            }
            memcpy(&uartbuf, queuemsg, sizeof(message_t));
            if (call UARTMessagePool.put(queuemsg) != SUCCESS) {
                fatal_problem();
                return;
            }
            post uartSendTask();
        }
    }

    // Indicate on messages that are not for this mote
    //================================================
    event message_t*
    Snoop.receive(message_t* msg, void* payload, uint8_t len) {
        aims_ctp_t *omsg = payload;
        // message received, but not for us, toggle led
        //report_received();
        return msg;
  }

    // Timer fires shortly after receiving a Beacon message
    //======================================================
    event void Timer.fired() {
        if (!sendbusy) {
            aims_ctp_t *o = (aims_ctp_t *)call Send.getPayload(&
                ↪ sendbuf, sizeof(aims_ctp_t));
            if (o == NULL) {
                fatal_problem();
                return;
            }
            memcpy(o, &local, sizeof(local));
            if (call Send.send(&sendbuf, sizeof(local)) == SUCCESS)
                sendbusy = TRUE;
            else
```

```
                report_problem();
            }
    }

    // Message has been sent via CTP
    //=================================
    event void Send.sendDone(message_t* msg, error_t error) {
        if (error == SUCCESS){
            report_sent();
        }
        else
            report_problem();
        sendbusy = FALSE;
    }

    // ==========================================
    // BEACON Code
    // ==========================================
    uint16_t counter = 0;
    bool busy = FALSE;
    message_t pkt;

    // every timer period (100ms)
    //================================
    event void BTimer0.fired() {
        counter++; // seq number.
        if (!busy) {
            AIMSBeaconMsg_t* aims_beacon_pkt = (AIMSBeaconMsg_t*)(call BPacket.getPayload(&pkt, sizeof (AIMSBeaconMsg_t)));
            aims_beacon_pkt->nodeid = TOS_NODE_ID;
            aims_beacon_pkt->counter = counter;
            if (call BAMSend.send(AM_BROADCAST_ADDR, &pkt, sizeof(AIMSBeaconMsg_t)) == SUCCESS) {
                busy = TRUE;
            }
        }
    }

    //Beacon Messages start is done
    //=================================
    event void BAMControl.startDone(error_t err) {
        if (err == SUCCESS) {
            if (TOS_NODE_ID == 8){
                call BTimer0.startPeriodic(TIMER_PERIOD_MILLI);
            }
        }
        else {
            call BAMControl.start(); // try again
        }
    }

    event void BAMControl.stopDone(error_t err) { }

    event void BAMSend.sendDone(message_t* msg, error_t error) {
        if (&pkt == msg) {
            busy = FALSE;
            report_sent(); //flash blue led
        }
    }

    event message_t* BReceive.receive(message_t* msg, void* payload, uint8_t len) {
        // root node doesnt need to receive
        if (len == sizeof(AIMSBeaconMsg_t) && TOS_NODE_ID != 0) {
            AIMSBeaconMsg_t* aims_beacon_pkt = (AIMSBeaconMsg_t*)
                ↪ payload;
            local.rssi = call CC2420Packet.getRssi(msg);
            local.seq = aims_beacon_pkt->counter;
            startTimer(); // delay then send
        }
        return msg;
    }

    // Use LEDs to report various status issues.

    static void fatal_problem() {
        if (TOS_NODE_ID == 0 || TOS_NODE_ID == 8)
        {
            call Leds.led0On();
            call Leds.led1On();
            call Leds.led2On();
            call Timer.stop();
        }
    }

    static void report_problem() {
        if (TOS_NODE_ID == 0 || TOS_NODE_ID == 8){
            call Leds.led0Toggle();
        }
    }

    static void report_sent() {
        if (TOS_NODE_ID == 0 || TOS_NODE_ID == 8){
            call Leds.led2Toggle();
        }
    }

    static void report_received() {
        if (TOS_NODE_ID == 0 || TOS_NODE_ID == 8) {
            call Leds.led1Toggle();
        }
    }


}
```

C.1.3 AIMS.h

Listing C.3: AIMS CTP Message Header

```
#ifndef AIMS_H
#define AIMS_H

enum {
    //NREADINGS = 0,
  DEFAULT_INTERVAL = 10, //Time after receiving beacon from target
  AM_AIMS = 0x93

};

typedef nx_struct aims_ctp {
    nx_uint16_t interval; // Samping period
    nx_uint16_t id;  //id of sending mote.
    nx_uint8_t rssi; // Received Signal Strength
    nx_uint16_t seq; // Synchronisation sequence number
} aims_ctp_t;

#endif
```

C.1.4 AIMS_BEACON.h

Listing C.4: AIMS Beacon Message Header

```
#ifndef AIMS_BEACON_H
#define AIMS_BEACON_H

enum {
    AM_AIMSBEACONMSG = 6, // message type
    TIMER_PERIOD_MILLI = 100 // beacon period
};

 typedef nx_struct AIMSBeaconMsg {
   nx_uint16_t nodeid;
   nx_uint16_t counter;
} AIMSBeaconMsg_t;

#endif
```

C.2 Localisation Software Code

C.2.1 Main Application aims.c

Listing C.5: AIMS Localisaton App - Main

```
//
    ↪ ------------------------------------------------------------------------
    ↪
// Adaptable Infant Monitoring System
//
    ↪ ------------------------------------------------------------------------
    ↪
// Author: Brad Goold
// Date: 26 May 2014
// Email Address: W0085400@umail.usq.edu.au
//
// Purpose: This software interacts with a wireless sensor network
    ↪ to
// perform localisation and boundary detection of a target sensor
// inside that network via the Fingerprinting/Profiling localisation
// method.
//
// Pre: The TelosB wireless sensors must be switched on and the
// base-station must be plugged into the USB port on the host
// machine.
//
// Post:
//
    ↪ ------------------------------------------------------------------------
    ↪
// APP
#include <stdio.h>
#include <stdlib.h>
#include <mysql.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <inttypes.h>
#include <unistd.h>
#include "sfsource.h"
#include "serialsource.h"
#include <math.h>
#include <time.h>
#include "wsn_if.h"
#include "aims.h"
#include "db_if.h"
#include "keybd.h"
```

```c

// Data structure to hold packet information.
struct Packet Rec_Packet,Packet_Buffer[BUFFER_SIZE];

// Settings struct
struct settings g_settings;

//=====================================================================
// Function Prototypes
//=====================================================================
//User interface
void clearScreen(void);
void display_settings(void);
void set_boundary(const char *dev, const int baud, void (*message)(serial_source_msg problem));
void set_monitoring_samples(void);
//=====================================================================

void set_up(const char *dev, const int baud, void (*message)(serial_source_msg problem));
void profile(const char *dev, const int baud, void (*message)(serial_source_msg problem));
void monitor(const char *dev, const int baud, void (*message)(serial_source_msg problem));
void set_mon_samples(const char *dev, const int baud, void (*message)(serial_source_msg problem));
void set_algorithm(const char *dev, const int baud, void (*message)(serial_source_msg problem));

//=====================================================================
// Menu stuff
//=====================================================================
typedef struct {
    char * desc;
    void (*func)(const char *dev, const int baud, void (*message)(serial_source_msg problem));
} MenuItem;

MenuItem Menu[]  = {
    {"Profiling", profile},
    {"Monitor", monitor},
    {"Graph", mon_1},
    {"Set up", set_up},
    {"Set Algorithm Parameters", set_algorithm},
    {"Change number of samples for Monitoring", set_mon_samples},
    {"Test Accuracy", test},
    {"Set Boundaries", set_boundary},
    {NULL, NULL},
};

int print_menu(MenuItem * p)
{
    int ctr = 1;
    printf("|======|\r\n");
    printf("| MENU |\r\n");
    printf("|======|\r\n");
    while(p->desc != NULL)
    {
        printf("%d. %s\r\n", ctr++, p->desc);
        p++;
    }
    printf("'q' to QUIT or go back\r\n");
    return ctr-1;
}


```

```c
//=============================================================================

int main(int argc, char **argv)
{
    //=========================================================================
    // Menu Variables
    //=========================================================================
    MenuItem * selected_item;
    MenuItem * menu = &Menu[0];
    char key = 'y';
    int selection;
    int num_menu_items;
    //=========================================================================
    char * dev;
    int baud;
    int i;

    //====================================
    // initialise g_settings values
    //====================================
    for (i = 0; i < MAX_SENSORS; i++)
        g_settings.ids[i] = 0;

    //====================================
    // Validate cmd line args
    //====================================
    if (argc != 3)
    {
        fprintf(stderr, "Adaptable Infant Monitoring System AIMS\n");
        fprintf(stderr, "\tUsage: %s <device> <rate>\n\tdevice = basestation mote device file\n\trate = buad rate (bps)\n", argv[0]);
        fprintf(stderr, "Example: %s /dev/mote1 115200 \n", argv[0]);
        exit(2);
    }
    dev = argv[1];
    baud = atoi(argv[2]);
    system("clear");
    printf("|==================================|\r\n");
    printf("|Adaptable Infant Monitoring System|\r\n");
    printf("|==================================|\r\n");
    printf("\t    |==========|\r\n");
    printf("\t    | A I M S |\r\n");
    printf("\t    |==========|\r\n\r\n");

    //========================================
    // Check the state of the Database
    //========================================
    if(check_db_health() == EXIT_FAILURE)
        set_up(dev, baud, stderr_msg);
    //================================================================
    // MENU
    //================================================================
    printf("Welcome! Please make your selection from the menu\r\n");
    set_conio_terminal_mode(); // Set for raw input
    num_menu_items = print_menu(menu);

    while (key != 'q' && key != 'Q')
    {

        while(!kbhit()) // wait for keypress
        {
            usleep(1000); // give the processor a rest
        }
        key = getch(); // consume key
        printf("%c\r\n", key);
        selection = atoi(&key); //numerical selection

        if (selection <= num_menu_items && selection > 0)
        {
            selected_item = &menu[selection -1];
            reset_terminal_mode(); //reset raw input

            // run the function from the menu
            selected_item->func(dev, baud, stderr_msg);

            // print the menu again
            set_conio_terminal_mode();
            num_menu_items = print_menu(menu);
            //printf("asdfads %s\r\n", selected_item->desc);

        }


    }
    reset_terminal_mode();
    clearScreen();
    printf("Thank␣you␣for␣using␣A.I.M.S!\r\n");

    return 0;
    //=================================================================================

}

//====================================================================================
// HELPER FUNCTIONS
//====================================================================================
void clearScreen()
{
    const char* CLEAR_SCREE_ANSI = "\e[1;1H\e[2J";
    write(STDOUT_FILENO,CLEAR_SCREE_ANSI,12);
}

unsigned char set_g_settings(void){
    unsigned char *set[7];
    set[0] = &g_settings.num_sensors;
    set[1] = &g_settings.num_samples;
    set[2] = &g_settings.x_size;
    set[3] = &g_settings.y_size;
    set[4] = &g_settings.num_mon_samples;
    set[5] = &g_settings.k_nearest;
    set[6] = &g_settings.k_centroid;
    int ret = 0;
    unsigned int ans;
    int ii = 0;
    char * questions[7] = {
        "How␣many␣sensors␣are␣you␣using?␣␣␣",
        "How␣many␣samples␣for␣profiling?␣␣␣",
        "What␣is␣the␣x-size␣of␣the␣area?␣␣␣",
        "what␣is␣the␣y-size␣of␣the␣area?␣␣␣",
        "How␣many␣samples␣to␣take␣during␣monitoring?␣",
        "Nearest␣nodes?␣",
        "Centroid␣points?␣",
    };

    for (ii = 0; ii < 7 ; ii++)
    {
        printf("%s", questions[ii]);
        // Get user input
        ret = scanf("%i", &ans);
        // Clear the standard input stream
        scanf("%*[^\n]");
        scanf("%*c");

        while(ret != 1)
        {
            printf("need a single integer! try again: ");
            // Get user input
            ret = scanf("%i", &ans);
            // Clear the standard input stream
            scanf("%*[^\n]");
            scanf("%*c");
        }
        //ans = atoi(&ans);
        printf("ans = %d\n", ans);
        *set[ii] = ans;
    }
    // clearScreen();
    //printf("Done!\n\r");
    return 0;


}

void set_monitoring_samples(void)
{
    int ret, ans;

    printf("Number of samples for Monitoring?");
    // Get user input
    ret = scanf("%i", &ans);
    // Clear the standard input stream
    scanf("%*[^\n]");
    scanf("%*c");

    while(ret != 1)
    {
        printf("need a single integer! try again: ");
        // Get user input
        ret = scanf("%i", &ans);
        // Clear the standard input stream
        scanf("%*[^\n]");
        scanf("%*c");
    }
    //ans = atoi(&ans);
    printf("saving %d to settings\n", ans);
    g_settings.num_mon_samples = ans;

}

void display_settings(void){
    int ii;
    printf("Contents of the G_SETTINGS struct\n");
    printf("----------------------------------------\n");
    printf("Number of Sensors = %d\n", g_settings.num_sensors);
    printf("Number of Samples = %d\n", g_settings.num_samples);
    printf("Size in x direction = %d\n", g_settings.x_size);
    printf("Size in y direction = %d\n", g_settings.y_size);
    printf("number of samples for monitoring = %d\n", g_settings.num_mon_samples);
    printf("k nearest nodes = %d\n", g_settings.k_nearest);
    printf("k centroid points = %d\n", g_settings.k_centroid);

    for (ii = 0; ii < MAX_SENSORS; ii++)
        printf("Sensor ID[%d] = %d\n", ii, g_settings.ids[ii]);
    printf("----------------------------------------\n\n");

```

```
}

enum PositionHistory {
    NO_HISTORY = 1,
    FULL_HISTORY = 2,
} PositionHistory;

void stderr_msg(serial_source_msg problem)
{
  fprintf(stderr, "Note: %s\r\n", msgs[problem]);
}


//
   ↪ ===================================================================
   ↪
// HELPER FUNCTIONS FOR MENU
//
   ↪ ===================================================================
   ↪
void set_up(const char *dev, const int baud, void (*message)(
   ↪ serial_source_msg problem))
{
    clear_database(); //drop the database and recreate it
    display_settings();
    set_g_settings(); //user enters values
    get_sensor_id(dev, baud, stderr_msg); // search network for
       ↪ sensor id's
    create_tables(); // create tables based on received values
    store_g_settings(); //store the settings in the g_settings
    display_settings();
    //table
}

void set_mon_samples(const char *dev, const int baud, void (*message
   ↪ )(serial_source_msg problem))
{
    set_monitoring_samples();
    store_g_settings();
}

//
   ↪ ===================================================================
   ↪
void profile(const char *dev, const int baud, void (*message)(
   ↪ serial_source_msg problem))
{
    printf("Profiling called\r\n");
    delete_pos_data();
    profiling(dev, baud, stderr_msg);
    populate_main();
}

//
   ↪ ===================================================================
   ↪
void monitor(const char *dev, const int baud, void (*message)(
   ↪ serial_source_msg problem))
{
    set_conio_terminal_mode();
    char key = 'x';
    double * x_pos = (double *)malloc(sizeof(double));
    double * y_pos = (double *)malloc(sizeof(double));


    while(key != 'q')
    {
        while (!kbhit()) {
            /* do some work */
            mon_1(dev, baud, stderr_msg);
//          Monitoring(dev, baud, stderr_msg);
//          get_position(x_pos, y_pos);

        }
        key = getch(); /* consume the character */

    }
    free(x_pos);
    free(y_pos);
    reset_terminal_mode();
    printf("terminal reset\r\n");

}

// ==========================================================================
void set_algorithm(const char *dev, const int baud, void (*message)(serial_source_msg problem))
{
    int ret, ans;

    printf("Number of points of polygon (1-5)?");
    // Get user input
    ret = scanf("%i", &ans);
    // Clear the standard input stream
    scanf("%*[^\n]");
    scanf("%*c");

    while(ret != 1)
    {
        printf("need a single integer! try again: ");
        // Get user input
        ret = scanf("%i", &ans);
        // Clear the standard input stream
        scanf("%*[^\n]");
        scanf("%*c");
    }
    //ans = atoi(&ans);
    printf("saving %d to settings\n", ans);
    g_settings.k_centroid = ans;

    printf("Number of nearest nodes (3 or 4)?");
    // Get user input
    ret = scanf("%i", &ans);
    // Clear the standard input stream
    scanf("%*[^\n]");
    scanf("%*c");

    while(ret != 1)
    {
        printf("need a single integer! try again: ");
        // Get user input
        ret = scanf("%i", &ans);
        // Clear the standard input stream
        scanf("%*[^\n]");
        scanf("%*c");
    }
    //ans = atoi(&ans);
    printf("saving %d to settings\n", ans);
    g_settings.k_nearest = ans;


    store_g_settings();
}
```

```
//
    ↪ ====================================================================================
    ↪
```

C.2.2 WSN interface wsn_if.c

Listing C.6: AIMS WSN interface module

```
//
    ↪ ------------------------------------------------------------------------------------
    ↪
// Adaptable Infant Monitoring System
//
    ↪ ------------------------------------------------------------------------------------
    ↪
// Author: Brad Goold
// Date: 26 May 2014
// Email Address: W0085400@umail.usq.edu.au
//
// Purpose: This software interacts with a wireless sensor network
    ↪ to
// perform localisation and boundary detection of a target sensor
// inside that network via the Fingerprinting/Profiling localisation
// method.
//
// Pre: The TelosB wireless sensors must be switched on and the
// base-station must be plugged into the USB port on the host
// machine.
//
// Post:
//
    ↪ ------------------------------------------------------------------------------------
    ↪


//
    ↪ ------------------------------------------------------------------------------------
    ↪
// Included Libraries
//
    ↪ ------------------------------------------------------------------------------------
    ↪
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <termios.h>
#include <inttypes.h>
#include "sfsource.h"
#include "serialsource.h"
#include <math.h>
#include "aims.h"
#include "wsn_if.h"
#include "db_if.h"
#include <time.h>
#include <endian.h>
//
    ↪ ------------------------------------------------------------------------------------
    ↪
//Data Structures
extern struct settings g_settings;
extern struct Packet Rec_Packet;
//
    ↪ ------------------------------------------------------------------------------------
    ↪
// Function Prototypes
//
    ↪ ------------------------------------------------------------------------------------
    ↪
inline void print_packet(struct Packet P); //helper function
struct Packet * wsn_receive_packet(serial_source fd); // receive and
                                                       // parse
//
    ↪ ------------------------------------------------------------------------------------------
    ↪


// Receive and parse the packet. Creates a packet structure and
// populates the data.
// returns a pointer to the packet received.
struct Packet * wsn_receive_packet(serial_source fd){
    //Local Variables
    //---------------
    //    serial_source fd = NULL; // file descriptor for the serial
        ↪ source
    int len; //length of the recieved packet.
    uint8_t ph; //used for triangular swap
    char * packet; //pointer to the data received from the forwarder
    char local_copy_packet[PACKET_SIZE+1]; //local copy of the
        ↪ received
                                          //packet.

    //Allocate memory for the new packet structure.
    struct Packet * p = (struct Packet *)malloc(sizeof(struct Packet
        ↪ ));

    // read from the serial forwarder
    packet = read_serial_packet(fd, &len);

    // validate packet
    if (!packet || len != PACKET_SIZE || p == NULL)
    {
        fprintf(stderr, "The received packet is NULL or not the
            ↪ expected length or not enough memory\r\n");
        free(packet);
        return NULL; // the serial forwarder has failed
    }

    //make a local copy of the packet
    memcpy(local_copy_packet, packet, len+1);

    //triangular swap for wrong endianness.
    //---------------------------------
    memcpy(&ph, &local_copy_packet[14], sizeof(uint8_t));
    memcpy(&local_copy_packet[14], &local_copy_packet[13], sizeof(
        ↪ uint8_t));
    memcpy(&local_copy_packet[13],&ph, sizeof(uint8_t));
    // build Struct from packet read
    // ----------------------------
    p->source_id = local_copy_packet[SOURCE_ID];
    p->rssi = local_copy_packet[RSSI];
    memcpy(&p->seq_no, &local_copy_packet[SEQ_NO], 2);
    free(packet);
    return p;
}

// Profile the area
void profiling(const char *dev, const int baud, void (*message)(
    ↪ serial_source_msg problem)){
    // LOCAL VARIABLES
    //==================
    serial_source fd; //file descriptor
    int ii, kk, ll, x_pos = 1, y_pos = 1; // iterators
    uint16_t num_stored = 0; // Number of complete sequences stored
    unsigned char found_items[g_settings.num_sensors];
    unsigned char num_found = 0;
    struct Packet * p; // points to the recieved packet.
    int sfd; // serial file descriptor for flushing device file
        ↪ buffer

    //-----------------------------------------------------
    // Open the serial forwarder and get file descriptor
    //-----------------------------------------------------
    fd = open_serial_source(dev, baud, 0, stderr_msg);
    if (!fd)
    {
        fprintf(stderr, "Couldn't open serial port at %s:%d\r\n",dev
            ↪ , baud);
        return;
    }
    sfd = serial_source_fd(fd);
    //-----------------------------------------------------
    // loop through each position
    //-----------------------------------------------------
    for (x_pos = 1; x_pos <= g_settings.x_size; x_pos++)
    {
        for (y_pos = 1; y_pos <= g_settings.y_size; y_pos++)
        {
            printf("Please put the sensor at position X:%d, Y:%d\r\r
                ↪ \n",
                    x_pos, y_pos);
            fflush(stdin);
            printf("Confirm with ENTER:");
            // We dont care what input we get back, as long as enter
            // is pressed..
            scanf("%*c");
            //-----------------------------------------------------
            // loop through until we get the number of samples we
                ↪ need
            // for each location
            //
                ↪ -------------------------------------------------------
                ↪
            while(num_stored < g_settings.num_samples )
            {
                tcflush(sfd, TCIFLUSH); // clear the device file
                    ↪ buffer
                p = wsn_receive_packet(fd); // receive the packet
                tcflush(sfd, TCIFLUSH);

                if (p)
                {
                    //Validate Addresses (dont need base or target)
                    if (p->source_id != 0x00 &&
                        p->source_id != 0x08)
                    {
                        //shift right buffer.
                        for (kk = BUFFER_SIZE-1; kk > 0; kk--)
                        {
                            memcpy(&Packet_Buffer[kk],
                                   &Packet_Buffer[kk-1], sizeof(
                                       ↪ struct Packet));
                        }
                        // put the read value into the first element
                        memcpy(&Packet_Buffer[0], p, sizeof(struct
                            ↪ Packet));
                        // Find how many instances of each sequence
                            ↪ number in
                        // the buffer there are. If there are the name amount
                        // of instances as there are sensors, then we have a
                        // full suite and we can store these instances in the DB
                        found_items[0] = 0; //points to the first packet
                                          //in the buffer (the one we
                                          //just saved);
                        num_found = 1; // The first one is a given

                        for (ll = 0; ll < BUFFER_SIZE; ll++)
                        {
                            if (Packet_Buffer[ll].seq_no ==
                                Packet_Buffer[0].seq_no &&
                                Packet_Buffer[ll].source_id ==
                                Packet_Buffer[0].source_id &&
                                ll > 0)
                            {
                                Packet_Buffer[0].seq_no = 0xFFFF;
                                Packet_Buffer[0].source_id = 0;
                                Packet_Buffer[0].rssi = -128;
                            }

                            if (Packet_Buffer[ll].seq_no ==
                                Packet_Buffer[0].seq_no &&
                                Packet_Buffer[ll].source_id !=
                                Packet_Buffer[0].source_id)
                            {
                                num_found++;
                                found_items[num_found-1] = ll; // points
                                //to element
                            }
                        }//for ll = 0 ...

                        if (num_found == g_settings.num_sensors)
                        {
                            // we have a full set,
                            num_stored++;
                            num_found = 0;

                            for (ii = 0; ii < g_settings.num_sensors; ii++)
                            {
                                Packet_Buffer[found_items[ii]].rssi -= 45;
                                store(Packet_Buffer[found_items[ii]], x_pos, y_pos);
                            }
                            //
                        }// if (num_found...
                        num_found = 0;
                    } //validate addresses
                }// if len == packet_size
                free(p);
            }//while num_stored
            num_stored = 0; // reset for next loop
        }//for y_pos
    }//for x_pos
    close_serial_source(fd);
}

inline void print_packet(struct Packet P)
```

```
{
    printf("SEQ = %d, SOURCE_ID = %d, RSSI = %d\r\n", P.seq_no, P.
        ↪ source_id, P.rssi);
}

unsigned char get_sensor_id(const char *dev, const int baud, void (*
    ↪ message)(serial_source_msg problem)){

    serial_source fd; //file descriptor
    int ii, flag = 0, counter = 0;
    uint8_t found_ids[g_settings.num_sensors]; // storage for id's
    struct Packet * p; //received packet
    int num_ids_found = 0;

    fd = open_serial_source(dev, baud, 0, stderr_msg);
    if (!fd)
    {
        fprintf(stderr, "wsn_recieve_packet(): Couldn't open serial 
            ↪ port at %s:%d\r\n",dev, baud);
        return EXIT_FAILURE;
    }
    //initialise
    for (ii = 0; ii < g_settings.num_sensors; ii++)
    {
        found_ids[ii] = 255;
    }
    for (ii = 0; ii < MAX_SENSORS; ii++)
        g_settings.ids[ii] = 0;
    //get sensors
    while (num_ids_found < g_settings.num_sensors)
    {
        // We are only looking for specific packets*/
        p = wsn_receive_packet(fd);
        if (p != NULL)
        {
            //Validate Addresses (dont need base or target)
            if (p->source_id != 0x00 &&
                p->source_id != 0x08)
            {
                // printf("found id number %d\r\r\n", p->source_id);
                for (ii = 0; ii < g_settings.num_sensors; ii++)
                {
                    if (p->source_id == found_ids[ii])
                        flag = 1;
                }
                if (flag == 0){
                    found_ids[num_ids_found] = p->source_id;
                    g_settings.ids[num_ids_found] = found_ids[
                        ↪ num_ids_found];
                    num_ids_found++;
                }
                flag = 0;
            }
        }
        counter++;
        if (counter > 100)
        {
            printf("Failed getting sensor id's: too many tries\r\n")
                ↪ ;
            return 1;
        }
    }// while
    close_serial_source(fd);
    return 0;
}
```

```c

void Monitoring(const char *dev, const int baud, void (*message)(serial_source_msg problem)){
    serial_source fd; //file descriptor
    int ii, kk, ll; // iterators
    uint16_t num_stored = 0; // Number of complete sequences stored
    unsigned char found_items[g_settings.num_sensors];
    unsigned char num_found = 0;
    int sfd; //serial file descriptor for flushing
    struct Packet * p;
    //----------------------------------------------------------------
    // loop through until we get the number of samples we need
    // for each location
    // ----------------------------------------------------------------------
    fd = open_serial_source(dev, baud, 0, stderr_msg);
    if (!fd)
    {
        fprintf(stderr, "wsn_recieve_packet(): Couldn't open serial port at %s:%d\r\n",dev, baud);
        return;
    }
    sfd = serial_source_fd(fd);

    while(num_stored < g_settings.num_mon_samples)
    {
        // We are only looking for specific packets
        tcflush(sfd, TCIFLUSH);
        p = wsn_receive_packet(fd);
        tcflush(sfd, TCIFLUSH);  // flush device buffer.
        if (p)
        {
            fflush(stdout);
            //Validate Addresses (dont need base or target)
            if (p->source_id != 0x00 &&
                p->source_id != 0x08)
            {
                //shift right buffer.
                for (kk = BUFFER_SIZE-1; kk > 0; kk--)
                {
                    memcpy(&Packet_Buffer[kk],
                           &Packet_Buffer[kk-1], sizeof(struct Packet));
                }
                // put the read value into the first element
                memcpy(&Packet_Buffer[0], p, sizeof(struct Packet));
                // Find how many instances of each sequence number in
                // the buffer there are. If there are the name amount
                // of instances as there are sensors, then we have a
                // full suite and we can store these instances in the DB

                for (ll = 0; ll < BUFFER_SIZE; ll++)
                {
                    if (Packet_Buffer[ll].seq_no ==
                        Packet_Buffer[0].seq_no &&
                        Packet_Buffer[ll].source_id ==
                        Packet_Buffer[0].source_id &&
                        ll > 0)
                    {
                        Packet_Buffer[0].seq_no = 0xFFFF;
                        Packet_Buffer[0].source_id = 0;
                        Packet_Buffer[0].rssi = -128;
                    }
                    if (Packet_Buffer[ll].seq_no ==
                        Packet_Buffer[0].seq_no &&
                        Packet_Buffer[ll].source_id
                        != Packet_Buffer[0].source_id)
                    {
                        num_found++;
                        found_items[num_found-1] = ll; //points
                        if (num_found == g_settings.num_sensors-1)
                        {
                            // we have a full set,
                            num_stored++;
                            num_found = 0;
                            Packet_Buffer[0].rssi -= 45;
                            store_monitor(Packet_Buffer[0],
                               ↪ num_stored);
                            for (ii = 0; ii < g_settings.num_sensors
                               ↪ -1; ii++)
                            {

                                Packet_Buffer[found_items[ii]].rssi
                                   ↪ -= 45;
                                store_monitor(Packet_Buffer[
                                   ↪ found_items[ii]], num_stored);
                            }
                        }// if (num_found...
                    }// if item == search variable...
                }//for ll = 0 ...
                num_found = 0;
            } //validate addresses
        }// if len == packet_size
    }//while num_stored
    num_stored = 0; // reset for next loop
    close_serial_source(fd);

}

// get time difference from timespec structs
struct timespec diff(struct timespec start, struct timespec end)
{
        struct timespec temp;
        if ((end.tv_nsec-start.tv_nsec)<0) {
                temp.tv_sec = end.tv_sec-start.tv_sec-1;
                temp.tv_nsec = 1000000000+end.tv_nsec-start.tv_nsec;
        } else {
                temp.tv_sec = end.tv_sec-start.tv_sec;
                temp.tv_nsec = end.tv_nsec-start.tv_nsec;
        }
        return temp;
}

// Testing code for accuracy and response time.
//-------------------------------------------------
void test(const char *dev, const int baud, void (*message)(
   ↪ serial_source_msg problem))
{
    int x_pos, y_pos;
    double x,y;
    uint32_t sample = 0;
    uint8_t num_samples = 5;
    uint8_t ii;
    FILE * DataFile;
    time_t rawtime;
    struct tm * timeinfo;
    const char * filename = "Test_Data";
    char * datetime = (char *) calloc(50, sizeof(char));
```

```c
    char * fullname =(char *) calloc(50, sizeof(char));
    struct timespec ts1, ts2, ts_res;
    double time_spent, time_per_sample;
    double error = 0.0, error_sum = 0.0;
    double time_sum = 0.0;
    int gs_knn = g_settings.k_nearest; // save for later
    int gs_kc = g_settings.k_centroid; // save for later
    int knn = 0;
    int kc = 0;
    int jj = 0;
    //==================================================
    // Get time for file name
    time (&rawtime);
    timeinfo = localtime (&rawtime);
    sprintf(datetime, "%d_%d_%d_%d_%02d_%02d", timeinfo->tm_year+1900,timeinfo->tm_mon+1, timeinfo->tm_mday, timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
    strcpy(fullname, filename);
    strcat(fullname, datetime);
    strcat(fullname, ".csv");
    //==================================================
    // Open File
    DataFile = fopen (fullname,"w");
    if (DataFile==NULL)
    {
        printf("couldnt open file for writing! Ouch\r\n");
        exit(1);
    }
    // Add the headings of the columns.
    fprintf(DataFile, "SAMPLE,ACT_X_POS, ACT_Y_POS, ");
        for (jj = 0; jj < 20; jj++)
            fprintf(DataFile, "KNN, KC, EST_X_POS, EST_Y_POS, TIME_PER_SAMPLE, ERROR, ");
    fprintf(DataFile, "\n");
    // Loop through each of the positions
    for (x_pos = 1; x_pos <= g_settings.x_size; x_pos++)
    {
        for (y_pos = 1; y_pos <= g_settings.y_size; y_pos++)
        {
            sample++;
            system("clear"); // not the most elegant solution
            printf("Please put the sensor at position X:%d, Y:%d\r\n", x_pos, y_pos);
            fflush(stdin);
            printf("Confirm with ENTER:");
            // Get user input
            scanf("%*c");
            // take (num_samples * g_settings.num_mon_samples) to get lots
            // of data
            for (ii = 0; ii < num_samples; ii++)
            {
                fprintf(DataFile, "%d, %d, %d, ", sample,x_pos, y_pos);
                printf("%d, %d, %d, ", sample,x_pos, y_pos);
                // loop through the possible algorithm parameters
                for (knn = 3; knn <=6 ;knn++)
                {
                    for (kc = 1; kc <=5; kc++)
                    {
                        g_settings.k_nearest = knn;
                        g_settings.k_centroid = kc;
                        clock_gettime(CLOCK_REALTIME, &ts1); //start time
                        // Calc Position
                        Monitoring(dev, baud, stderr_msg);
                        get_position(&x, &y);

                        clock_gettime(CLOCK_REALTIME, &ts2); //end
                            ↪ time
                        //calculate position error
                        error = sqrt(pow((x-x_pos),2) + pow((y-y_pos
                            ↪ ),2));
                        error_sum += error;  // sum for average

                        //calculate response time
                        ts_res = diff(ts1,ts2); // time difference
                        //convert to double
                        time_spent = ts_res.tv_sec +
                            (double)((double)ts_res.tv_nsec/(double)
                                ↪ 1000000000);
                        time_per_sample = time_spent/(double)
                            ↪ g_settings.num_mon_samples;
                        time_sum += time_per_sample; // sum for
                            ↪ average

                        // put in CSV
                        fprintf(DataFile, "%d,%d,%2.2f,%2.2f,%2.5f,␣
                            ↪ %2.5f,␣",
                                knn,kc, x, y, time_per_sample, error
                                    ↪ );
                        printf("%d,%d,%2.2f,%2.2f,%2.5f,␣%2.5f,␣",
                                knn,kc, x, y, time_per_sample, error
                                    ↪ );
                    }
                }
                fprintf(DataFile, "\n");
                printf("\n");
            }
            fflush(DataFile); // save on every set of samples.
        }
    }

    // add averages to end of file
    fprintf(DataFile, "Average␣Error,␣%2.5f\r\nAverage␣Time,␣%2.5f\r
        ↪ \n", error_sum/(g_settings.x_size * g_settings.y_size * 5
        ↪ * 20),time_sum/(g_settings.x_size * g_settings.y_size * 5
        ↪ * 20));
    fclose(DataFile);
     g_settings.k_centroid = gs_kc; // put originals back
     g_settings.k_nearest = gs_knn; // put originals back
     return ;
}
```

C.2.3 Database interface db_if.c

Listing C.7: AIMS Database interface module

```
//
    ↪ -----------------------------------------------------------------------
    ↪
// Author: Brad Goold
// Date: 26 Jul 2014
// Email Address: W0085400@umail.usq.edu.au
//
// Purpose: Interaction with the database and the aims application.
// performs the database queries and returns the results for use by
// the application
//
    ↪ -----------------------------------------------------------------------
    ↪
//
    ↪ --------------------------------------------------------------------------------
    ↪
// Included Libraries
//
    ↪ --------------------------------------------------------------------------------
    ↪
#include <stdio.h>
#include <stdlib.h>
#include <mysql.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <signal.h>
#include <inttypes.h>
#include "sfsource.h"
#include "serialsource.h"
#include <math.h>
#include "aims.h"
#include "wsn_if.h"
#include "db_if.h"
#include "vector.h"

// States of the cells for evaluation
char * StateNames[3] = {
    "SAFE",
    "TRANS",
    "UNSAFE"
};

//
    ↪ --------------------------------------------------------------------------------
    ↪
// Database information
const char *server = "localhost";
const char *user = "root";
const char *password = "XXXXXXXXX"; // You wont be able to access if
                                    // this is not correct.
const char *database = "Aims";

// compare function for sorting.
int cmp(const void * a, const void * b)
{
    double * x = (double *)a;
    double * y = (double *)b;
    if (*x == *y)
        return 0;
    else if (*x < *y)
        return -1;
    else return 1;
}

//
    ↪ --------------------------------------------------------------------------------
    ↪
// Sum struct is used to store the error sums for each sensor
    ↪ compared
// to the current value and its position in the table (row number)
//
    ↪ --------------------------------------------------------------------------------
    ↪
struct Sum {
    double sum; ///Sum of errors
    int pos; // position in the database (row number)
};

struct s_r {
    int sensor_num;
```

```c
    int index;
    double rssi;
};

// ------------------------------------------------------------------------------------------------
// compare function for finding the best ranked positions used by
// qsort.
// ------------------------------------------------------------------------------------------------
int cmp1(const void *y1, const void *y2)
{
    struct Sum * s1 = (struct Sum *)y1;
    struct Sum * s2 = (struct Sum *)y2;

    if (s1->sum < s2-> sum)
        return -1;
    else if( s1->sum == s2->sum)
        return 0;
    else return 1;
}

// compare RSSI values for qsort
int cmp2(const void *r1, const void *r2)
{
    struct s_r * s1 = (struct s_r*)r1;
    struct s_r * s2 = (struct s_r*)r2;
    // printf("called! s1 = %2.2f, s2 = %2.2f\r\n", s1->rssi, s2->rssi);

    if (s1->rssi < s2->rssi)
        return -1;
    else if (s1->rssi == s2->rssi)
        return 0;
    else return 1;
}


unsigned char get_position(double * x_pos, double * y_pos)
{
    // Get the position by matching to the database.
    //----------------------------------------------
    MYSQL *conn;
    MYSQL_RES *res, *res2;
    MYSQL_ROW row;
    char  suf[] =  "FROM monitor";
    char middle1[512];
    char temp1[128];
    char query[512];
    int ii,jj = 0,kk = 0,ll;
    int kkk = 0;
    uint16_t num_fields = 0;
    conn = mysql_init(NULL);
    double sum1 = 0.0;
    int nnodes = g_settings.k_nearest; // nearest nodes;
    struct Sum sums1[g_settings.x_size * g_settings.y_size];
    double results1[g_settings.x_size * g_settings.y_size][nnodes];
    int zz = 0;
    int k = g_settings.k_centroid; // nearest matches.
    int error_thresh = 255; //floor (nnodes * 2.5);
    Position * best_k_pos[k];
    Position centroid;
    static double last_x = 0.0;
    static double last_y = 0.0;
    struct s_r srs[g_settings.num_sensors];

    //Initialise arrays
    //-----------------
    for (ii = 0; ii < g_settings.x_size * g_settings.y_size; ii++)
    {
        for (jj = 0; jj < nnodes; jj++)
        {
            results1[ii][jj] = 0.0;
        }
        sums1[ii].sum = 0.0;
        sums1[ii].pos = 0.0;
    }
    jj = 0;
    ii = 0;
    //Connect to the Database
    //-----------------------
    if (!mysql_real_connect(conn, server,
                            user, password, database, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    // get values from the last "monitor" call from the database
    //----------------------------------------------------------
    sprintf(middle1, "SELECT AVG(S%02d_rssi)", g_settings.ids[0]);
    for (ll = 1; ll < g_settings.num_sensors; ll++)
    {
        sprintf(temp1, ",AVG(S%02d_rssi) ", g_settings.ids[ll]);
        strcat(middle1, temp1);
    }
    strcat(middle1, suf);
    //printf("Query: '%s'\r\n", middle1);
    if (mysql_query(conn, middle1))
    {
        fprintf(stderr, "fault getting averages\r\n");
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    // Get row
    res = mysql_use_result(conn);
    row = mysql_fetch_row(res);

    //save row data in srs array
    if (row != NULL)
    {
        for (ll = 0; ll < g_settings.num_sensors; ll++)
        {

            srs[ll].sensor_num = g_settings.ids[ll];
            srs[ll].rssi = fabs(atof(row[ll]));
            srs[ll].index = ll; // index for tracking sensor later
        }
    }
    else
    {
        // row corrupt or empty.
        mysql_close(conn);
        return EXIT_FAILURE;
    }

    //Sort the array by RSSI values
    qsort(srs, g_settings.num_sensors, sizeof(struct s_r), cmp2);
    mysql_free_result(res);

    //===============
    // Algorithm to get position
    //  printf("Finding closest position\r\n");
    if (mysql_query(conn, "SELECT * from Main;"))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    res2 = mysql_use_result(conn);
    num_fields = mysql_num_fields(res2);
    row = mysql_fetch_row(res2);
    // loop through each row
    while (row != NULL)
    {
        // loop through each field in that row
        for (ii = 1; ii < num_fields; ii++)
        {
            // need to use only rows that have non-null values
            if (row[ii] != NULL)
            {
                // get nearest nodes only as calculated before
                for (kkk=0;kkk<nnodes;kkk++)
                {
                    if (srs[kkk].index == ii-1)
                    {
                        results1[jj][kkk] = srs[kkk].rssi - fabs(
                            ↪ atof(row[ii]));
                    }
                }
            }
            else
            {
                printf("Main Table has NULL Field!\r\r\n");
                exit(0);
            }
            kk++;
        }
        kk = 0;
        jj++; // increase row number
        row = mysql_fetch_row(res2);
    }

    //this code and be incorporated above.
    for (ii = 0; ii < (g_settings.x_size * g_settings.y_size); ii++)
    {
        for (jj=0;jj<nnodes;jj++)
        {
            sum1+= fabs(results1[ii][jj]);
        }
        sums1[ii].sum = sum1;
        sums1[ii].pos = ii;
        sum1 = 0.0;
    }

    mysql_free_result(res2);

    //sort the sums1 array by the error value
    qsort(sums1, g_settings.x_size*g_settings.y_size, sizeof(struct
        ↪ Sum), cmp1);

    // loop through the top k positions
```

```c
    for (zz = 0; zz < k; zz++)
    {
        best_k_pos[zz] = (Position *)malloc(sizeof(Position));
        sprintf(query, "SELECT * FROM Main ORDER BY position LIMIT %d,1", sums1[zz].pos);
        printf("%s\n\r", query);
        if (mysql_query(conn, query))
        {
            fprintf(stderr, "%s\r\n", query);
            fprintf(stderr, "%s\r\n", mysql_error(conn));
            mysql_close(conn);
            return 1;
        }

        res = mysql_use_result(conn);
        row = mysql_fetch_row(res);

        if (row != NULL)
        {
            char c = row[0][0];
            // this method only allows single digit x_size and y_size
            // and should be changed
            *x_pos = (double)atoi(&c);
            *y_pos = (double)atoi(&row[0][2]);

        }
        else
        {
            // row corrupt or empty.
            fprintf(stderr, "NULL Row in DB\r\n");
            mysql_close(conn);
            return EXIT_FAILURE;
        }

        // the best positions are put into an array of position
        // structs for use by the weighted centroid algorithm.
        best_k_pos[zz]->x = *x_pos;
        best_k_pos[zz]->y = *y_pos;
        best_k_pos[zz]->error = sums1[zz].sum;

        mysql_free_result(res);

    }

    if (best_k_pos[0]->error > error_thresh)
    {
        *x_pos = last_x;
        *y_pos = last_y;
        printf("error too high!\r\n");
    }
    else
    {
        //get_centroid(best_k_pos,k, &centroid);
        get_weighted_centroid(best_k_pos,k, &centroid);
        *x_pos = centroid.x;
        *y_pos = centroid.y;
        printf("newpos =  %2.4f, %2.4f, e=%d\r\n", *x_pos, *y_pos, best_k_pos[0]->error);

        last_x = *x_pos;
        last_y = *y_pos;
    }
    mysql_close(conn);
    printf("leaving get_pos\n\r");
    return 0;
```

```
}

// populates the main table with the averages from the individual
// tables for each position.
unsigned char populate_main(void)
{
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;
    char query[512];
    //char pos[16];
    int xx, yy, ii;
    int sensor_number = 255;
    int average = 0;
    //Initialise connection
    conn = mysql_init(NULL);

    //Connect to the Database
    if (!mysql_real_connect(conn, server,
                            user, password, database, 0, NULL, 0))
    {
        //failed
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return EXIT_FAILURE;
    }

    for (xx = 0; xx < g_settings.x_size; xx++)
    {
        for (yy = 0; yy < g_settings.y_size; yy++)
        {
            for (ii = 0; ii < g_settings.num_sensors; ii++)
            {
                sensor_number = g_settings.ids[ii];

                //get the average of the rssi's for this sensor at
                //this position
                sprintf(query, "SELECT AVG(rssi) FROM s%02d_pos%02d
                    ↪ %02d;", sensor_number, xx+1,yy+1);

                if (mysql_query(conn, query))
                {
                    fprintf(stderr, "%s\r\n", mysql_error(conn));
                    mysql_close(conn);
                    return EXIT_FAILURE;
                }

                res = mysql_use_result(conn);

                row = mysql_fetch_row(res);
                if (row != NULL)
                {

                    average = atoi(row[0]);
                }
                else
                {
                    mysql_close(conn);
                    return EXIT_FAILURE;
                }

                mysql_free_result(res);

                //insert this value into XXYY in main table

                sprintf(query, "UPDATE Main SET S%02d_rssi = %d
                    ↪ WHERE position = %02d%02d;",sensor_number,
                   ↪ average,xx+1, yy+1);
//                printf("%s\r\n", query);
                if (mysql_query(conn, query))
                {
                    fprintf(stderr, "%s\r\n", mysql_error(conn));
                    mysql_close(conn);
                    return 1;
                }
            }//for ii
        }//for jj
    }// for kk
    mysql_close(conn);

    return 0;
}

//clear the monitor table contents.
unsigned char clear_monitor(void){
    MYSQL *conn;
    char query[512];
    conn = mysql_init(NULL);

    //Connect to the Database
    if (!mysql_real_connect(conn, server,
                            user, password, database, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    sprintf(query, "DELETE FROM monitor;");
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    mysql_close(conn);
    return 0;
}

// store the values in the monitor table.
unsigned char store_monitor(struct Packet P, int sample)
{
    MYSQL *conn;
    char query[512];
    conn = mysql_init(NULL);

    //Connect to the Database
    if (!mysql_real_connect(conn, server,
                            user, password, database, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    sprintf(query, "UPDATE monitor SET S%02d_rssi = %d WHERE sample
       ↪ = %02d;", P.source_id, P.rssi, sample);
    //  printf("%s\r\n", query);
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    mysql_close(conn);
    return 0;
}

// check the database is ok and the values and tables match the
// parameters
unsigned char check_db_health(void)
{
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;
    char query[512];
    int ii;
    //Initialise connection
    conn = mysql_init(NULL);
    //Connect to the Database

    if (!mysql_real_connect(conn, server,
                            user, password, database, 0, NULL, 0))
    {
        //failed
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return EXIT_FAILURE;
    }

    //get values from g_settings table
    if (mysql_query(conn, "select * from g_settings where item = 0")
        ↪ ) {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return EXIT_FAILURE;
    }
    res = mysql_use_result(conn);

    // Store values in g_settings data stucture
    row = mysql_fetch_row(res);
    if (row != NULL)
    {
        g_settings.num_sensors = atoi(row[1]);
        g_settings.x_size = atoi(row[2]);
        g_settings.y_size = atoi(row[3]);
        g_settings.num_samples = atoi(row[4]);
        g_settings.num_mon_samples = atoi(row[5]);
        g_settings.k_nearest = atoi(row[6]);
        g_settings.k_centroid = atoi(row[7]);

    }
    else
    {
        // row corrupt or empty.
        mysql_close(conn);
        return EXIT_FAILURE;
    }

    mysql_free_result(res);
    //get values from sensors table
    if (mysql_query(conn, "select * from sensors")) {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return EXIT_FAILURE;
    }
    res = mysql_use_result(conn);

    // Store values in g_settings data stucture
    row = mysql_fetch_row(res);
    ii = 0;
    while (row != NULL)
    {
        g_settings.ids[ii++] = atoi(row[1]);
        row = mysql_fetch_row(res);
    }

    mysql_free_result(res);

    // check that the number of sensors in the table is
    // correct
    if (ii != g_settings.num_sensors)
    {
        printf("Wrong number of sensors in the table\r\n");
        //Wrong number of sensors in the table;
        mysql_close(conn);
        return EXIT_FAILURE;
    }

    //get the number of columns in Main table;
    sprintf(query, "SELECT count(*) FROM information_schema.columns 
        ↪ WHERE table_schema = '%s' AND table_name = 'Main';",
        ↪ database);
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return EXIT_FAILURE;
    }
    res = mysql_use_result(conn);
    row = mysql_fetch_row(res);
    if (row != NULL)
    {

        if ( atoi(row[0]) != (1 + (g_settings.num_sensors)) )
        {
            return EXIT_FAILURE;
            mysql_close(conn);
        }
    }
    else
    {
        // row corrupt or empty.
        mysql_close(conn);
        return EXIT_FAILURE;
    }

    mysql_free_result(res);

    //get the number of tables in Database;
    sprintf(query, "SELECT COUNT(*) FROM information_schema.tables 
        ↪ WHERE table_schema = '%s';",database);
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return EXIT_FAILURE;
    }

    res = mysql_use_result(conn);

    row = mysql_fetch_row(res);
    if (row != NULL)
    {
        if ( atoi(row[0]) != (7 + (g_settings.num_sensors *
        ↪ g_settings.x_size * g_settings.y_size)) )
        {
            mysql_close(conn);
            return EXIT_FAILURE;
        }
    }
    else
    {
        // row corrupt or empty.
        mysql_close(conn);
        return EXIT_FAILURE;
    }

    mysql_free_result(res);
    mysql_close(conn);

    return EXIT_SUCCESS;
}

unsigned char clear_database(void)
{
    MYSQL *conn;
    char query[512];
    conn = mysql_init(NULL);

    //Connect to the Database
    if (!mysql_real_connect(conn, server,
                            user, password, NULL, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));

        return 1;
    }

    // drop the database
    sprintf(query, "drop database %s", database);
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    sprintf(query, "create database %s", database);
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    sprintf(query, "use %s", database);
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;

    }

    mysql_close(conn);
    return 0;
}




unsigned char delete_pos_data(void)
```

```
{
    MYSQL *conn;
    char table_name[30];
    char query[512];
    int ii = 0, x_pos = 0, y_pos = 0;
    conn = mysql_init(NULL);
    //Connect to the Database
    if (!mysql_real_connect(conn, server,
                              user, password, database, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    for (ii = 0; ii < g_settings.num_sensors; ii++)
    {
        for (x_pos = 1; x_pos <= g_settings.x_size; x_pos++)
        {
            for (y_pos = 1; y_pos <=g_settings.y_size; y_pos++)
            {

                sprintf(table_name, "s%02d_pos%02d_%02d", g_settings
                    ↪ .ids[ii], x_pos, y_pos);
                sprintf(query, "delete␣from␣␣%s;", table_name);
                if (mysql_query(conn, query))
                {
                    fprintf(stderr, "%s\r\n", mysql_error(conn));
                    mysql_close(conn);
                    return 1;
                }
            }
        }
    }
    //printf("Query sent: %s\r\n", query);
    mysql_close(conn);
    return 0;
}

//store the packet data in the database
unsigned char store(struct Packet P, uint8_t x_pos, uint8_t y_pos){
    MYSQL *conn;
    char table_name[30];
    char query[512];

    sprintf(table_name, "s%02d_pos%02d_%02d", P.source_id, x_pos,
        ↪ y_pos);
    conn = mysql_init(NULL);
    //Connect to the Database
    if (!mysql_real_connect(conn, server,
                              user, password, database, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    sprintf(query, "insert␣into␣%s␣(rssi)␣VALUES␣␣(%d);", table_name
        ↪ , P.rssi);
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
```

```c
    mysql_close(conn);
    return 0;
}

// store the global settings.
unsigned char store_g_settings(void)
{
    MYSQL *conn;
    char query[512];
    conn = mysql_init(NULL);
    int ii;

    //Connect to the Database
    if (!mysql_real_connect(conn, server,
                            user, password, database, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        return 1;
    }
    // clear current values
    sprintf(query, "delete from g_settings where item = 0");
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    //insert values from g_settings struct
    sprintf(query, "insert into g_settings (item, x_size, y_size, num_samples, num_sensors, num_mon_samples, k_nearest, k_centroid) VALUES  (0,%d,%d,%d,%d, %d, %d, %d);", g_settings.x_size, g_settings.y_size, g_settings.num_samples, g_settings.num_sensors, g_settings.num_mon_samples, g_settings.k_nearest, g_settings.k_centroid);
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    //clears all rows in the 'sensors' table;
    sprintf(query, "delete from sensors;");
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    ii = 0;
    while(g_settings.ids[ii] != 0)
    {
        sprintf(query, "insert into sensors (item,id_number) VALUES (%d,%d);",ii+1, g_settings.ids[ii]);
        if (mysql_query(conn, query))
        {
            fprintf(stderr, "%s\r\n", mysql_error(conn));
            mysql_close(conn);
            return 1;
        }
        ii++;

    }
    mysql_close(conn);
    return 0;
```

```
}


unsigned char create_tables(void)
{

    MYSQL *conn;
    char query[256];
    int kk, ii, jj, ll;
    char middle[512], * middle1, * middle2, * middle3;
    char temp[100], temp1[100], temp2[512];

    middle1 = (char *)calloc(512, 1);
    middle2 = (char *)calloc(512, 1);
    middle3 = (char *)calloc(512, 1);

    // initialise
    for (ii = 0; ii < 512;ii++)
        middle[ii] = '\0';


    conn = mysql_init(NULL);
    /* Connect to database */
    if (!mysql_real_connect(conn, server,
                                user, password, database, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        return 1;
    }



    sprintf(query, "CREATE TABLE IF NOT EXISTS g_settings(item INT
        ↪ (4) UNIQUE PRIMARY KEY , num_sensors INT(4), x_size INT(4)
        ↪ , y_size INT(4), num_samples INT(4), num_mon_samples INT
        ↪ (4), k_nearest INT(4), k_centroid INT(4)) ENGINE = INNODB;
        ↪ ");
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    sprintf(query, "CREATE TABLE IF NOT EXISTS sensors(item INT(4)
        ↪ UNIQUE PRIMARY KEY, id_number INT(4)) ENGINE = INNODB;");
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    // initialise the grid so we can use the position values.
    for (kk = 0; kk < g_settings.num_sensors; kk++)
    {
        for (ii = 0; ii < g_settings.x_size; ii++)
        {
            for (jj = 0; jj < g_settings.y_size; jj++)
            {

                sprintf(query, "CREATE TABLE IF NOT EXISTS s%02d_pos
                    ↪ %02d_%02d(sample INT(4) AUTO_INCREMENT PRIMARY
                    ↪ KEY, rssi INT(4)) ENGINE = INNODB;",
                    ↪ g_settings.ids[kk], ii+1, jj+1);
                if (mysql_query(conn, query)) {
                    fprintf(stderr, "%s\r\n", mysql_error(conn));
                    mysql_close(conn);
                    return 1;
                }
            }

        }
    }

    const char  prefix[] = "CREATE TABLE IF NOT EXISTS Main(position INT(4) UNIQUE PRIMARY KEY ";
    const char  prefix2[] = "CREATE TABLE IF NOT EXISTS Current_positions(number INT(4) UNIQUE PRIMARY KEY AUTO_INCREMENT, x_pos DOUBLE(9,2), y_pos DOUBLE(9,2) ";
    const char  suffix[] =  ") ENGINE = INNODB;";
    const char suffix2[] =  ") ENGINE = INNODB;";
    strcat(middle, prefix);
    strcat(middle2, prefix2);
    for (ll = 0; ll < g_settings.num_sensors; ll++)
    {
        // sprintf(temp, ",S%02d_rssi INT(4), S%02d_lqi INT(4)", g_settings.ids[ll], g_settings.ids[ll]);
        // sprintf(temp2, ",S%02d_rssi INT(4), S%02d_lqi INT(4)", g_settings.ids[ll], g_settings.ids[ll]);
        sprintf(temp, ",S%02d_rssi INT(4)", g_settings.ids[ll]);
        sprintf(temp2, ",S%02d_rssi INT(4)", g_settings.ids[ll]);
        strcat(middle, temp);
        strcat(middle2,temp2);
    }
    strcat(middle, suffix);
    strcat(middle2, suffix2);

    if (mysql_query(conn, middle))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    if (mysql_query(conn, middle2))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    char  pre[] = "CREATE TABLE IF NOT EXISTS monitor(sample INT(4) UNIQUE PRIMARY KEY \0";
    char  suf[] =  ") ENGINE = INNODB;\0";
    strcat(middle1, pre);
    for (ll = 0; ll < g_settings.num_sensors; ll++)
    {
        sprintf(temp1, ",S%02d_rssi INT(4)", g_settings.ids[ll]);
        strcat(middle1, temp1);
    }
    strcat(middle1, suf);
    //  printf("%s\r\n", middle1);
    if (mysql_query(conn, middle1))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    //  printf("Inserting values into monitor\r\r\n");
    for (ii = 0; ii < g_settings.num_mon_samples; ii++)
    {
        sprintf(query, "insert into monitor(sample) VALUES  (%02d);",ii+1);
        if (mysql_query(conn, query)) {
            fprintf(stderr, "%s\r\n", mysql_error(conn));
            mysql_close(conn);
            return 1;
        }
    }
//  printf("Creating diff table\r\r\n");
    sprintf(query, "CREATE TABLE IF NOT EXISTS diff LIKE Main;");
    if (mysql_query(conn, query)) {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    for (ii = 0; ii < g_settings.x_size; ii++)
    {
        for (jj = 0; jj < g_settings.y_size; jj++)
        {
            sprintf(query, "insert into Main (position) VALUES  (%02d%02d);",ii+1, jj+1);
            if (mysql_query(conn, query)) {
                fprintf(stderr, "%s\r\n", mysql_error(conn));
                mysql_close(conn);
                return 1;
            }
            sprintf(query, "insert into diff (position) VALUES  (%02d%02d);",ii+1, jj+1);
            if (mysql_query(conn, query)) {
                fprintf(stderr, "%s\r\n", mysql_error(conn));
                mysql_close(conn);
                return 1;
            }
        }
    }


    char  pre1[] = "CREATE TABLE IF NOT EXISTS boundary(y INT(4) UNIQUE PRIMARY KEY,  1 VARCHAR(4) \0";
    char  suf1[] =  ") ENGINE = INNODB;\0";
    strcat(middle3, pre1);
    for (ll = 1; ll < g_settings.x_size; ll++)
    {
        sprintf(temp1, ", %d varchar(4)", ll+1);
        strcat(middle3, temp1);

    }
    strcat(middle3, suf1);
//  printf("%s\r\n", middle3);
    if (mysql_query(conn, middle3))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    for (ii = 0; ii < g_settings.y_size; ii++)
    {
        sprintf(query, "insert into boundary (y) VALUES  (%d);",ii+1);
        if (mysql_query(conn, query)) {
            fprintf(stderr, "%s\r\n", mysql_error(conn));
            mysql_close(conn);
            return 1;
        }
```

```c
    }
    //change all null values to "s"
    for (ii = 1; ii <= g_settings.x_size; ii++)
    {
        sprintf(query, "UPDATE boundary SET _%d = IFNULL(_%d, 's')", ii, ii);
        if (mysql_query(conn, query)) {
            fprintf(stderr, "%s\r\n", mysql_error(conn));
            mysql_close(conn);
            return 1;
        }
    }
    /* close connection */
    mysql_close(conn);
    free(middle1);
    free(middle2);
    free(middle3);
    return 0;
}

unsigned char store_current_reading(double x, double y)
{
    MYSQL *conn;
    char query[512];
    conn = mysql_init(NULL);

    //Connect to the Database
    if (!mysql_real_connect(conn, server,
                            user, password, database, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        return 1;
    }

    sprintf(query, "insert into Current_positions(x_pos,y_pos) VALUES  (%2.2f,%2.2f);", x, y);
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    mysql_close(conn);
    return 0;
}

unsigned char store_boundary(int x_pos, int y_pos, char * state)
{
    MYSQL *conn;
    char query[512];
    conn = mysql_init(NULL);

    //Connect to the Database
    if (!mysql_real_connect(conn, server,
                            user, password, database, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        return 1;
    }
    sprintf(query, "UPDATE boundary SET _%d = '%s' WHERE y = %d;", x_pos, state, y_pos);

    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    mysql_close(conn);
    return 0;
}

unsigned char get_state(int x_pos, int y_pos)
{
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;
    char *  c = (char *)calloc(1, sizeof(char));
    char query[512];
    conn = mysql_init(NULL);
    //Connect to the Database
    if (!mysql_real_connect(conn, server,
                            user, password, database, 0, NULL, 0))
    {
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }
    sprintf(query, "SELECT  %d FROM boundary where y = %d", x_pos, y_pos);
    if (mysql_query(conn, query))
    {
        fprintf(stderr, "%s\r\n", query);
        fprintf(stderr, "%s\r\n", mysql_error(conn));
        mysql_close(conn);
        return 1;
    }

    res = mysql_use_result(conn);
    row = mysql_fetch_row(res);
    if (row != NULL)
    {
        *c = row[0][0];
    }
    else
    {
        // row corrupt or empty.
        mysql_close(conn);
        return EXIT_FAILURE;
    }

    mysql_free_result(res);
    mysql_close(conn);

    switch(*c)
    {
    case 's':
        return SAFE;
    case 't':
        return TRANS;
    case 'u':
        return UNSAFE;
    default: // something went wrong
        return UNSAFE;
    }
    return UNSAFE;

}

void mon_1(const char *dev, const int baud, void (*message)(serial_source_msg problem))
```

```
{
    double * x_pos = (double *) malloc(sizeof(int));
    double * y_pos = (double *) malloc(sizeof(int));
    LocationState state;
    static LocationState last_state;
    static double last_x_pos = 3450.22;
    static double last_y_pos = 3450.22;
    int ii = 0, jj = 0;

    Monitoring(dev, baud, stderr_msg);
    get_position(x_pos, y_pos);
    state = get_state(*x_pos,*y_pos);
    printf("current state = %s\r\n", StateNames[state]);
    printf("last_state = %s\r\n", StateNames[last_state]);
    switch (last_state)
    {
    case SAFE:
    {
        if (state == UNSAFE)
        {
            // cant transition to UNSAFE without going through
            // the TRANS state... must be a bad reading. Keep
            // old values
            printf("Moving from SAFE to UNSAFE without being in
               ↪ TRANS not allowed... keeping position\r\n");
            *x_pos = last_x_pos;
            *y_pos = last_y_pos;
            state = SAFE;
        }
        break;
    }
    case TRANS:
    {
        if (state == UNSAFE)
        {
            // ALARM HERE
            printf("\a\r\n");
            printf("SENSOR IN UNSAFE AREA\r\n");
        }
        break;
    }
    case UNSAFE:
    {
        if (state == SAFE)
        {
            // cant transition to SAFE without going through
            // the TRANS state... must be a bad reading. Keep
            // old values
            printf("Moving from UNSAFE to SAFE without being in
               ↪ TRANS not allowed... keeping position\r\n");
            *x_pos = last_x_pos;
            *y_pos = last_y_pos;
            state = UNSAFE;
        }

        break;
    }
    }

    printf("%f, %f\r\n", *x_pos, *y_pos);
    store_current_reading(*x_pos, *y_pos);
    for (jj = 0; jj < g_settings.y_size; jj++)
    {
        printf("| ");
        for (ii = 0; ii < g_settings.x_size;ii++)
```

```c
        {
            if (ii == abs(*x_pos)-1 && jj == abs(*y_pos)-1)
                printf("X ");
            else printf("0 ");

        }
        printf("|\r\n");
    }
    last_state = state;
    last_x_pos = *x_pos;
    last_y_pos = *y_pos;
    free(x_pos);
    free(y_pos);

}


void set_boundary(const char *dev, const int baud, void (*message)(serial_source_msg problem)){
    int ret = 0;
    unsigned int xpos=255, ypos=255;
    char * state = (char *)calloc(2, sizeof(char));
    // int ii = 0;
    char choice = 'y';
    printf("All positions are currently set to SAFE, so\n");
    printf("you just need to select the postions that are\n");
    printf("UNSAFE or TRANSITIONAL and define them as such.\n");

    while (choice != 'x')
    {
        printf("Please input x position: ");
        // Get user input
        ret = scanf("%i", &xpos);
        // Clear the standard input stream
        scanf("%*[^\n]");
        scanf("%*c");
        while(ret != 1 || xpos < 1 || xpos > g_settings.x_size)
        {
            printf("Error: need a single integer\n");
            printf("Choose a number between %d and %d: ", 1, g_settings.x_size);
            // Get user input
            ret = scanf("%i", &xpos);
            // Clear the standard input stream
            scanf("%*[^\n]");
            scanf("%*c");
        }

        printf("ans = %d\n", xpos);
        printf("Please input y position: ");
        // Get user input
        ret = scanf("%i", &ypos);

        // Clear the standard input stream
        scanf("%*[^\n]");
        scanf("%*c");

        while(ret != 1 || ypos < 1 || ypos > g_settings.y_size)
        {
            printf("Error: need a single integer\n");
            printf("Choose a number between %d and %d: ", 1, g_settings.y_size);
            // Get user input
            ret = scanf("%i", &ypos);
            // Clear the standard input stream
            scanf("%*[^\n]");
            scanf("%*c");
        }

        printf("ans = %d\n", ypos);

        printf("(S)afe, (U)nsafe, or (T)ransitional: ");
        // Get user input
        ret = scanf("%c", state);

        // Clear the standard input stream
        scanf("%*[^\n]");
        scanf("%*c");
        *state = tolower(*state);
        if (*state == 's' || *state == 't' || *state == 'u')
        {
            printf("ans = %c\n", *state);
        }
        else
        {
            printf("must use u, s, or t!\n");
            *state = 's';
        }
        store_boundary(xpos,ypos, state);
        // Get user input
        printf("\n\n Enter 'x' to exit, any other key to keep inputting values: ");
        ret = scanf("%c", &choice);
        // Clear the standard input stream
        scanf("%*[^\n]");
        scanf("%*c");
        if (ret ==  1 && choice == 'x')
            return;
    }
    return ;

}
```

C.2.4 Centroid Algorithms vector.c

Listing C.8: AIMS Database Algorithms module

```c
//------------------------------------------------------------------------------
// Author: Brad Goold
// Date: 20 Sep 2014
// Email Address: W0085400@umail.usq.edu.au
//
// Purpose: Positioning for AIMS. Performs calculations on the
// positions and implements the Centroid and Weighted centroid algorithms
//------------------------------------------------------------------------------

//------------------------------------------------------------------------------
// Included Libraries
//------------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
//------------------------------------------------------------------------------
```

```
#include "vector.h"

#define MAX_ERROR_DIFF 20 //maximum error difference for weighted
                          //centroid calculation

typedef struct Vect {
    Position * P1;
    Position * P2;
    double angle;
    double length;
    int quad;
    double weight; //This is the weighting of the reduction for this
                   //vector. Using the error between the two
                   //positions, we create a weight factor for
                   //reduction.
} Vector;

// calculate the length of the vector.
int get_len_vector(Vector * V, double * l){
    if (V->P1 == NULL || V->P2 == NULL)
        return EXIT_FAILURE;
    else

    *l = sqrt( pow((V->P2->x - V->P1->x), 2) + pow((V->P2->y - V->P1
        ↪ ->y), 2));
    return EXIT_SUCCESS;
}

// getting quadrant REMEMBER not cartiesian coords... they relate to
// the database entry table.
int get_quad_vector(Vector * V, int * q){
    if (V->P1 == NULL || V->P2 == NULL)
        return EXIT_FAILURE;
    else
    {
        if( V->P2->x > V->P1->x && V->P2->y < V->P1->y)
            *q = 1;
        else if( V->P2->x <= V->P1->x && V->P2->y <= V->P1->y)
            *q = 2;
        else if( V->P2->x >= V->P1->x && V->P2->y >= V->P1->y)
            *q = 4;
        else if( V->P2->x <= V->P1->x && V->P2->y >= V->P1->y)
            *q = 3;
    }

    return EXIT_SUCCESS;
}

// calculates the angle of the vector. Pass in a pointer to the
    ↪ vector
// struct and a pointer to the double precision angle. This is
    ↪ usually
// the angle located inside the vector struct, but for
// interoperability, the pointer must be passed in separately.
int get_angle_vector(Vector * V, double * angle){
    if (V->P1 == NULL || V->P2 == NULL)
        return EXIT_FAILURE;
    else
    {
        *angle = atan(fabs(V->P2->y - V->P1->y) / fabs(V->P2->x - V
            ↪ ->P1->x));
    }
    return EXIT_SUCCESS;

}
```

```c
//pass in a pointer to an array of Position structs, the size and a Position
//for the return value of the centroid
int get_centroid(Position ** polygon, unsigned int num_el, Position * P){
    int ii;
    double sum_x = 0.0;
    double sum_y = 0.0;

    if (num_el == 0 || polygon == NULL)
    {

        return EXIT_FAILURE;
    }

    for (ii = 0; ii < num_el; ii++)
    {
        // printf("polygon[%d]->x = %2.4f", ii, polygon[ii]->x);
        sum_x += polygon[ii]->x;
        sum_y += polygon[ii]->y;
    }
    P->x = sum_x/num_el;
    P->y = sum_y/num_el;
    return EXIT_SUCCESS;
}

// calculates the weighting factor for the vector.
int calc_weight_vector(Vector * V, double  * w, int best_error)
{
    if (V == NULL || w == NULL)
        return EXIT_FAILURE;

//    int ii;

    *w = 1-( ((double)V->P2->error -  (double)V->P1->error)/( best_error + MAX_ERROR_DIFF) );
    if (*w < 0.5)
        *w = 0.5;
    else if (*w >= 1.01)
    {
        printf("ERROR! negative weight in calculations!\r\n");
    }

    return EXIT_SUCCESS;
}

void print_vector(Vector * V)
{
    printf("Printing Vector\n");
    printf("====================\n");
    printf("P1 = (%2.2f,%2.2f), add = \n", V->P1->x, V->P1->y);
    printf("P2 = (%2.2f,%2.2f), add = \n", V->P2->x, V->P2->y);
    printf("P1 error = %d\n", V->P1->error);
    printf("P2 error = %d\n", V->P2->error);
    printf("length = %2.2f\n", V->length);
    printf("weight = %2.8f\n", V->weight);
    printf("quadrant = %d\n", V->quad);
    printf("====================\n");
}

// Weighted Centroid.
// Calculates the weigheted centroid based on the parameters defined
// in g_settings.
int get_weighted_centroid(Position ** polygon, unsigned int num_el, Position * Centroid)
{
    int jj = 0, kk = 0;
    int best_match_error = 0;
    Vector * V = (Vector *)malloc(sizeof(Vector));
    Position * temp_pos = (Position *)malloc(sizeof(Position));
    temp_pos->x = 0;
    temp_pos->y = 0;
    temp_pos->error = 2353.2; // Just garbage to initialise

    if (polygon == NULL || num_el < 1 || Centroid == NULL)
        return EXIT_FAILURE;
    // set the best error to the first element in the array passed
    // in. This should be the closest position to the target as
    // calculated.
    best_match_error = polygon[0]->error;

    // loop through each of the points and set the positions of
        ↪ those
    // into the temporary vector V. This will loop through n(n-1)/2
    // different vectors as this defines the number of edges in a n
    // sided polygon.
    // point to the successive points in the array. Each time V is
        ↪ set
    // with new P's, move P2 back towards P1 depending on the weight
    // calculated.
    for (kk = 0; kk <  num_el-1; kk++){

        for (jj = kk; jj < num_el-1; jj++)
        {

            V->P1 = polygon[kk];
            V->P2 = polygon[jj+1];
            if (V->P1 == NULL || V->P2 == NULL )
                exit(9);

            calc_weight_vector(V, &V->weight, best_match_error);
            get_len_vector(V, &V->length);

            V->length = V->length * (V->weight);
            get_angle_vector(V, &V->angle);
            get_quad_vector(V, &V->quad);

            switch(V->quad)
            {
            case 1:
            {
                temp_pos->x = V->length*cos(V->angle);
                temp_pos->y = -1*(V->length*sin(V->angle));
                break;
            }
            case 2:
            {
                temp_pos->x =-1*( V->length*cos(V->angle));
                temp_pos->y =-1*( V->length*sin(V->angle));
                break;
            }
            case 3:
            {
                temp_pos->x =-1*( V->length*cos(V->angle));
                temp_pos->y = V->length*sin(V->angle);
                break;
            }
            case 4:
            {
                temp_pos->x = V->length*cos(V->angle);
                temp_pos->y = V->length*sin(V->angle);
                break;
                }
            }//switch

            V->P2->x = V->P1->x + temp_pos->x;

            V->P2->y = V->P1->y + temp_pos->y;


        }
        //    printf("count = %d\n", count);
    }
     get_centroid(polygon, num_el, Centroid);
    return EXIT_SUCCESS;
}
```

C.2.5 Raw Keyboard keybd.c

Listing C.9: AIMS Raw keyboard input module

```
//--------------------------------------------------------------------------------
// Author: Brad Goold
// Date: 10 Oct 2014
// Email Address: W0085400@umail.usq.edu.au
//
// Purpose: Provide a more elegant menu interface by taking in the raw
// keyboard input.
// Pre:
// Post:
//--------------------------------------------------------------------------------


//--------------------------------------------------------------------------------
// Included Libraries
//--------------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/select.h>
#include <termios.h>
#include "keybd.h"

struct termios orig_termios;

void reset_terminal_mode()
{
    tcsetattr(0, TCSANOW, &orig_termios);
}

// set the terminal mode.
void set_conio_terminal_mode()
{
    struct termios new_termios;
    /* take two copies - one for now, one for later */
    tcgetattr(0, &orig_termios);
    memcpy(&new_termios, &orig_termios, sizeof(new_termios));
    /* register cleanup handler, and set the new terminal mode */
    atexit(reset_terminal_mode);
    cfmakeraw(&new_termios);
    tcsetattr(0, TCSANOW, &new_termios);
```

```c
}

// act on keyboard event
int kbhit()
{
    struct timeval tv = { 0L, 0L };
    fd_set fds;
    FD_ZERO(&fds);
    FD_SET(0, &fds);
    return select(1, &fds, NULL, NULL, &tv);
}

// read in the char.
int getch()
{
    int r;
    unsigned char c;
    if ((r = read(0, &c, sizeof(c))) < 0) {
        return r;
    } else {
        return c;
    }
}
```

C.2.6 AIMS Makefile

Listing C.10: AIMS Makefile

```makefile
TARGET = aims
LIBS = -lm $(shell mysql_config --libs) libmote.a -lrt
CC = gcc
CFLAGS = -g -Wall $(shell mysql_config --cflags)

.PHONY: default all clean

default: $(TARGET)
all: default

OBJECTS = $(patsubst %.c, %.o, $(wildcard *.c))
HEADERS = $(wildcard *.h)

%.o: %.c $(HEADERS)
        $(CC) $(CFLAGS) -c $< -o $@

.PRECIOUS: $(TARGET) $(OBJECTS)

$(TARGET): $(OBJECTS)
        $(CC) $(OBJECTS) -Wall $(LIBS) -o $@

clean:
        -rm -f *.o
        -rm -f $(TARGET)
```

C.3 Matlab Data Analysis Code

C.3.1 Baseline Testing Analysis Code

Listing C.11: Baseline Test Analysis Matlab Code

```matlab
function [errors real_mean] = importfile(fileToRead1, tx)
%IMPORTFILE(FILETOREAD1)
%  Imports data from the specified file
%  FILETOREAD1:  file to read

%  Auto-generated by MATLAB on 20-Oct-2014 17:06:16
%  ** Modified to suit application by Brad Goold 20-Oct-2014 ****

% Import the file
newData1 = importdata(fileToRead1);

% Create new variables in the base workspace from those fields.
vars = fieldnames(newData1);
for i = 1:length(vars)
    assignin('base', vars{i}, newData1.(vars{i}))
end

A = zeros(315,11);
count = 1;
 for ii = 2:1:12
     A(:,count) = newData1.data(:,ii);
     count   = count + 1;
%      %newData1(:,ii);
 end;
real_mean = zeros(1,11);

count   = 2;
jj = 1
for ii = 1:1:3
    figure;
    hist(A(:,ii), 20);
    filename = sprintf('hist_weighted_TX%s_k%dv%d', tx, ii, jj);
    tit = sprintf('Distribution of Nearest %d and %d centroid 
        ↪ vertices for weighted algorithm at TX power %s', ii, jj,
        ↪ tx);
    title(tit);
    saveas(gcf, filename, 'png');
    close all;
    real_mean(1,ii) = raylstat(raylfit(A(:,ii)));
    count = count + 1;
end

 count   = 2;
ii = 6
for jj = 4:1:7
    figure;
    hist(A(:,jj), 20);
    filename = sprintf('hist_weighted_TX%s_k%dv%d', tx, ii, count);
    tit = sprintf('Distribution of Nearest %d and %d centroid 
        ↪ vertices for weighted algorithm at TX power %s', ii, count
        ↪ , tx);
    title(tit);
    saveas(gcf, filename, 'png');
    close all;
    real_mean(1,jj) = raylstat(raylfit(A(:,jj)));
    count = count + 1;
end
 count   = 2;
ii = 6
for jj = 8:1:11
    figure;
    hist(A(:,jj), 20);
    filename = sprintf('hist_centroid_TX%s_k%dv%d', tx, ii, count);
    tit = sprintf('Distribution of Nearest %d and %d centroid 
        ↪ vertices for centroid algorithm at TX power %s', ii, count
        ↪ , tx);
    title(tit);
    saveas(gcf, filename, 'png');
    close all;
    real_mean(1,jj) = raylstat(raylfit(A(:,jj)));
    count = count + 1;
```

```
end

figure;
filename = sprintf('rayleigh_mean__TX%s.csv', tx);
csvwrite(filename, real_mean);
figure;

hold on;
 col = {'r', 'g', 'b', 'k', 'y'};
for kk = 1:3
    g = find(A(:,kk)==0);
    c = numel(g);
    if (c == 0)
         probplot('rayleigh',A(:,kk));
    end
end

hold off;

filename = sprintf('prob_plot_KNN_C1_TX%s_',  tx);
saveas(gcf, filename, 'png');
close all;
hold on;
 col = {'r', 'g', 'b', 'k', 'y'};
for kk = 4:7
    g = find(A(:,kk)==0);
    c = numel(g);
    if (c == 0)
         probplot('rayleigh',A(:,kk));
    end
end

hold off;

filename = sprintf('prob_plot_Weighted_K6_TX%s_',  tx);
saveas(gcf, filename, 'png');
close all;
close all;
hold on;
 col = {'r', 'g', 'b', 'k', 'y'};
for kk = 8:11
    g = find(A(:,kk)==0);
    c = numel(g);
    if (c == 0)
         probplot('rayleigh',A(:,kk));
    end
end

hold off;

filename = sprintf('prob_plot_Centroid_K6_TX%s_',  tx);
saveas(gcf, filename, 'png');
close all;



 errors = A;
```

C.3.2 Residential Testing Analysis Code

Listing C.12: Baseline Test Analysis Matlab Code

```
function [errors real_mean] = importfile(fileToRead1, alg, tx)
%IMPORTFILE(FILETOREAD1)
%  Imports data from the specified file
%  FILETOREAD1:  file to read

%  Auto-generated by MATLAB on 20-Oct-2014 17:06:16
% Modified to suit Application by Brad Goold 20-Oct-2014

% Import the file
newData1 = importdata(fileToRead1);

% Create new variables in the base workspace from those fields.
vars = fieldnames(newData1);
for i = 1:length(vars)
    assignin('base', vars{i}, newData1.(vars{i}))
end

A = zeros(315,20);
count = 1;
 for ii = 9:6:123
     A(:,count) = newData1.data(:,ii);
     count  = count + 1;
%     %newData1(:,ii);
 end;
real_mean = zeros(1,20);
 count  = 1;
for ii = 3:1:6
    for jj = 1:1:5
        figure;
        hist(A(:,count), 20);
        filename = sprintf('hist_%s_TX%s_k%dv%d', alg, tx, ii, jj);
        tit = sprintf('Distribution of Nearest %d and %d centroid 
           ↪ vertices for %s algorithm at TX power %s', ii, jj, alg
           ↪ , tx);
        title(tit);
        saveas(gcf, filename, 'png');
        close all;
        real_mean(1,count) = raylstat(raylfit(A(:,count)));
        count = count + 1;
    end
end

figure;
filename = sprintf('rayleigh_mean_%s_TX%s.csv', alg, tx);
csvwrite(filename, real_mean);
figure;

hold on;
 col = {'r', 'g', 'b', 'k', 'y'};
for kk = 1:20

   g = find(A(:,kk)==0);
   c = numel(g);
   if (c == 0)


           probplot('rayleigh',A(:,kk));

   end

end

hold off;

filename = sprintf('prob_plot_%s_TX%s_', alg, tx);
saveas(gcf, filename, 'png');
close all;
errors = A;
```

Appendix D

Setting Up Development Platform

The document in this chapter was written by the author to enable further research workers of this project to set up the identical platform that was used to produce the system in this disseration. The document has been tested and the all links and steps provided were valid at the date of publishing. The files required can be sourced from the author at 'w0085400 at umail dot usq dot edu dot au' or 'brad [underscore] goold at dodo.com.au'.

Setting up an Ubuntu 12.04 VM with TinyOS, nesC and Eclipse Juno

Along with this documentation should be a folder with some binary packages and a few scripts. This procedure cannot be completed without access to these files.

You need to have some knowledge of Linux, VirtualBox and command line tools if you want to successfully complete this installation. Some of the steps here assume you know what is being implemented.

The first step is to download the Ubuntu 12.04 LTS 32-bit ISO image from Ubuntu.com. Once downloaded, create a virtual machine in VirtualBox. Point the CD/DVD drive to the ISO image and set the check box to “live CD” and run the VM. Make sure you set the network to NAT or Bridge to allow internet to your VM. Also if you would like to share folders, set up the folder sharing now and we will install the guest editions on the VM later. Follow the install procedures.

1. Boot the new Ubuntu in the VM and open the terminal
2. “sudo apt-get update && sudo apt-get upgrade”
3. “sudo apt-get install build-essential linux-headers-`uname –r` dkms”
4. “sudo apt-get install terminator default-jdk emacs subversion”
5. Add the following line to the /etc/apt/sources.list
   “deb http://tinyos.stanford.edu/tinyos/dists/ubuntu lucid main”
6. “sudo apt-get update && sudo apt-get install tinyos-2.1.2”
7. “chown –R <username>:<username>/opt/tinyos-2.1.2”
   where <username> is your user name such as “brad”
8. Copy the supplied file called “tinyos.sh” to the /etc/profile.d/ directory.
9. Log out and then back in again to get the environment variables loaded.
10. You should be able to complete the next command
    “cd $TOSROOT/support/sdk/java”
11. Install the tinyos java jni:
    “sudo tos-install-jni”
12. “make”
    “make install”

At this stage you should be able to go to the /opt/tinyos-2.1.2/apps/Blink directory and execute “make telosb”. It should compile successfully, although giving a warning that the compiler version is not correct. We will fix this later.

Now to update the VirtualBox guest additions (that you should hopefully have downloaded from VirtualBox). You will need to make sure the guest additions have been installed on the Host pc before proceeding.

1. Click on Devices in the VirtualBox menus, and click on “insert virtualbox guest additions CD”

2. Install the guest additions
3. We need to add your user to the vboxsf group so you can access the shared folders.
   Open the terminal and type:
   “sudo gpasswd –a <username> vboxsf”
4. Reboot

Now we need to update the gcc compiler to 4.6.3.

The easiest way to do this is to set the drag and drop settings in the devices menu to “Host to Guest” and drag and drop all of the files supplied with this document into the ~/Downloads folder

Double click on each of the files below starting from the first one and working down. This installs all of the dependencies for the compiler, then the compiler itself. The packages listed here are only available in higher versions of Ubuntu, but this is the latest LTS version, which is the most stable. Each double click should open the package manager and should allow you to install the package. It may tell you that it will “upgrade”, this is OK.

1. Libmpfr4-3.1.1-2
2. Libmpc3
3. Msp430libc-20120224
4. Msp430mcu-20120406
5. Gcc-msp430-4.6.3
6. Reboot
7. Check that the compiler is the correct version by typing
8. “gcc -- version”
9. Reboot

Now to install eclipse and the yeti2 plugin.

This is just a case of extracting the supplied eclipse tar.gz file.

1. Extract the tarball to /opt/. You will need to change the owner of the /opt/ directory in order to create the folder.
2. Copy the eclipse.desktop file (provided with this document) to the folder /usr/share/applications/
3. Open this folder in the file manager and locate the eclipse configuration file. Drag this file to the launcher bar on the left of the screen.. it should drop itself there permanently.
4. Click on the eclipse launcher to test eclipse opens.
5. Click help -> install new software
   Click on Add and type into the location box:
   “http://tos-ide.ethz.ch/update/site.xml”
   This may take a few minutes to update.
6. Click on all the checkboxes except for the one that has to do with Cygwin and windows. Then click “install”
7. Eclipse will ask to restart, do so.
8. Click on window->preferences->tinyOS->TinyOS-2.xUnix, then click the button “update default”. Apply.
9. Click on window-> open perspective->other->TinyOS

Ok, so this should be it. Try to import a project by following the following steps.

1. Click on file->import->tinyOs-filesystem.
2. Find /opt/tinyos-2.1.2/apps/blink and click open
3. Click on the project on the left and open the “src” folder.
4. On the bottom right of the screen is a box with the build directives. Right click on the line that shows the Blink application and click “build”
   The console box at the bottom of the screen should show a successful compilation.

Appendix E

Test Data

Table E.1: Baseline Test Data

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
1	4.12	1.00	4.12	0.86	5.22	2.14	2.41	4.84	1.58	3.07	2.69
1	4.12	3.61	4.12	0.87	5.28	1.96	2.87	5.64	1.58	3.07	2.69
1	4.12	1.00	4.12	0.86	5.39	1.75	2.35	4.68	1.58	3.07	2.69
1	4.12	1.00	4.12	0.86	5.34	1.83	2.95	5.53	1.58	3.07	2.69
1	4.12	1.00	4.12	0.86	5.38	1.82	2.87	5.88	1.58	3.07	2.69
2	0.00	0.00	0.00	1.60	1.13	2.07	0.67	3.88	1.25	0.95	1.12
2	0.00	0.00	0.00	1.69	2.57	2.06	0.74	4.28	1.25	0.95	1.12
2	0.00	0.00	0.00	1.68	1.41	2.11	0.78	3.88	1.25	0.95	1.12
2	0.00	0.00	0.00	1.65	1.09	2.07	0.80	2.78	1.25	0.95	1.12
2	0.00	0.00	0.00	1.60	1.16	2.09	0.82	3.88	1.25	0.95	1.12
3	1.00	1.00	2.00	2.25	0.33	0.92	1.02	5.57	3.05	1.49	1.41
3	1.00	1.00	5.83	2.25	0.53	0.92	1.00	5.57	3.05	1.49	1.41
3	1.00	1.00	2.00	2.25	0.47	0.94	1.00	5.57	3.05	1.49	1.41
3	1.00	1.00	1.00	2.25	0.51	0.94	1.03	5.57	3.05	1.49	1.41
3	1.00	1.00	5.83	2.24	0.61	0.94	1.03	5.57	3.05	1.49	1.41
4	0.00	3.16	2.83	1.80	0.54	2.01	3.09	2.72	3.64	3.15	3.04
4	2.24	3.16	2.83	1.64	0.59	2.06	2.93	2.43	3.64	3.15	3.04
4	0.00	3.16	2.83	1.69	0.56	2.20	3.15	1.46	3.64	2.13	2.69
4	3.61	3.16	2.83	1.69	0.56	2.07	2.95	2.43	3.64	2.13	2.69
4	0.00	3.16	5.39	1.69	0.56	2.18	2.86	2.04	3.64	3.15	2.69
5	6.08	6.08	1.41	0.64	2.97	1.04	1.01	2.78	3.26	1.66	0.71
5	6.08	4.00	1.00	0.58	2.88	0.97	1.01	3.13	3.26	1.66	0.71
5	6.08	4.00	2.00	1.14	2.86	0.96	0.99	4.10	3.50	1.66	0.71
5	6.08	4.00	5.83	0.58	2.84	1.01	0.98	4.40	3.51	1.66	0.71
5	6.08	6.08	2.24	0.58	2.83	1.04	0.89	4.10	3.51	1.66	0.71
6	2.24	1.41	2.00	1.58	2.05	0.59	1.85	2.51	0.25	1.41	2.55
6	2.00	1.41	2.00	1.56	2.02	0.69	1.80	2.51	0.25	1.41	1.58
6	2.00	1.41	2.00	1.57	2.03	0.71	1.83	2.78	0.25	1.41	1.58
6	2.00	1.41	2.00	1.56	2.01	0.56	1.82	2.95	0.25	1.41	1.58
6	2.00	1.41	2.00	1.56	1.67	0.61	1.82	2.78	0.25	2.11	1.58
7	1.00	1.00	1.00	0.70	0.95	0.91	0.86	2.41	0.75	1.05	1.50
7	1.00	1.00	1.00	0.71	0.98	1.08	0.73	2.09	0.75	1.05	1.50
7	1.00	1.00	1.00	1.29	0.94	0.79	0.85	2.41	1.52	1.05	1.50
7	1.00	1.00	1.00	1.29	0.95	0.88	0.79	2.41	1.52	1.05	1.50
7	1.00	1.00	1.00	0.63	0.96	1.15	0.78	2.41	0.75	1.05	1.50
8	0.00	0.00	0.00	1.18	0.55	1.09	2.40	2.13	0.75	3.54	2.50
8	2.00	0.00	0.00	1.21	0.54	1.11	2.43	2.06	0.75	1.66	2.50
8	0.00	0.00	0.00	1.21	0.59	1.40	2.34	1.89	0.75	1.66	2.50
8	0.00	0.00	2.24	1.23	0.57	1.41	2.26	1.00	0.75	1.66	2.50

Continued on next page

Table E.1 – continued from previous page

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
8	2.00	0.00	1.00	1.18	0.55	1.11	2.28	2.06	0.75	3.54	2.50
9	1.41	1.00	1.00	5.12	2.98	5.97	3.24	3.61	5.83	4.77	3.64
9	7.07	7.07	2.00	5.12	2.97	6.59	3.20	3.61	5.83	4.77	3.64
9	1.41	7.07	2.00	5.23	2.99	6.20	3.23	3.61	5.83	4.77	1.41
9	1.41	1.00	3.61	5.53	2.97	5.91	3.09	4.22	5.83	2.13	3.64
9	1.41	1.00	0.00	5.33	3.03	5.76	3.14	3.61	5.83	3.61	3.64
10	1.00	7.07	7.62	2.83	4.93	4.50	1.51	5.39	3.78	2.69	8.54
10	1.00	5.10	7.62	2.83	5.02	4.55	1.47	5.88	3.78	4.86	2.12
10	1.00	5.10	7.62	2.85	4.92	4.55	1.53	5.88	3.78	2.69	4.03
10	1.00	7.07	3.16	2.85	4.90	4.55	1.92	5.39	3.78	4.53	5.59
10	1.00	5.10	7.62	2.83	4.78	4.53	1.82	5.88	3.78	3.33	4.03
11	0.00	0.00	4.24	3.16	2.43	2.37	1.98	2.91	2.80	0.75	1.50
11	0.00	0.00	0.00	3.21	2.37	2.31	2.09	2.91	1.82	0.75	1.50
11	4.12	6.08	5.10	3.21	2.39	2.65	1.98	2.91	2.55	0.75	1.50
11	0.00	6.08	5.10	2.77	2.39	2.68	1.83	2.91	2.80	0.75	1.50
11	0.00	6.08	5.10	2.73	2.43	2.67	1.98	2.91	3.54	0.75	1.50
12	3.16	3.16	1.00	1.25	5.14	3.37	3.08	2.41	3.82	1.33	2.12
12	3.16	3.16	3.16	1.66	4.97	2.44	3.08	2.41	3.82	0.75	2.12
12	3.16	3.16	1.00	1.66	5.00	2.26	2.46	2.41	3.82	4.86	2.12
12	4.00	3.16	1.00	1.66	5.04	3.21	2.41	2.41	2.15	2.85	2.12
12	4.00	3.16	1.00	1.66	4.91	3.60	2.54	2.13	3.82	2.85	2.12
13	4.24	2.24	2.24	3.54	1.88	2.19	1.40	2.61	1.35	3.07	1.50
13	5.66	2.24	2.24	3.54	1.69	2.14	1.33	2.88	1.35	3.07	2.12
13	4.24	2.24	2.24	3.54	1.74	2.76	1.32	3.61	1.35	3.07	2.12
13	5.39	2.24	2.24	2.61	1.66	2.14	1.34	2.97	1.35	2.13	3.54
13	3.61	4.00	2.24	3.00	1.66	2.18	1.63	3.40	1.35	2.13	1.58
14	1.41	2.24	2.83	1.67	1.10	1.41	1.66	2.28	0.35	1.67	2.55
14	1.41	2.24	2.83	1.58	1.07	1.46	1.66	1.46	0.25	2.67	2.55
14	2.24	2.24	2.83	1.58	1.08	1.51	1.61	1.00	0.35	2.67	2.55
14	2.24	0.00	2.83	1.54	1.20	1.53	1.64	1.46	2.06	2.67	2.55
14	2.24	1.41	3.16	1.62	1.07	1.47	1.63	1.17	0.79	2.67	2.55
15	3.16	3.61	3.16	1.64	1.33	2.24	1.33	1.90	2.70	1.95	1.80
15	3.16	3.61	3.16	1.68	1.73	2.31	2.66	1.00	1.60	1.05	3.35
15	3.16	3.61	3.16	1.68	1.33	2.76	1.23	1.90	1.27	1.05	3.35
15	3.16	3.61	3.16	1.84	1.51	2.35	1.24	1.00	1.82	1.05	3.35
15	3.16	3.61	3.16	1.76	1.74	2.17	1.27	1.98	1.60	1.95	3.35
16	0.00	1.00	0.00	0.33	0.38	0.57	0.47	3.41	0.35	0.33	0.50
16	0.00	1.00	0.00	0.38	0.51	0.54	0.60	3.00	0.35	0.33	0.50
16	0.00	1.00	0.00	1.61	0.53	0.60	0.60	3.28	0.00	0.33	0.50
16	0.00	1.00	0.00	0.39	0.38	0.49	0.59	2.56	0.00	0.33	2.12
16	0.00	1.00	0.00	0.38	0.33	0.19	0.43	3.74	0.00	0.33	1.12
17	1.00	1.00	1.00	0.49	0.41	1.03	3.23	3.05	2.85	0.47	6.32
17	1.00	1.00	0.00	0.47	0.43	1.19	4.96	3.23	2.85	0.47	3.16
17	1.00	1.00	0.00	0.49	0.42	0.93	2.52	4.62	2.85	0.47	3.16
17	1.00	1.00	0.00	0.47	0.41	1.05	4.30	4.20	2.85	0.47	5.59
17	1.00	1.00	1.00	0.50	0.42	1.22	3.59	3.05	2.85	0.47	6.32
18	1.00	2.24	2.24	3.04	3.49	3.04	2.95	3.03	2.51	3.16	1.80
18	1.00	1.00	2.24	1.49	3.43	3.01	2.50	3.54	2.51	4.34	4.03
18	2.24	1.00	2.24	2.83	3.47	2.98	2.42	3.13	2.25	4.34	4.12
18	1.00	1.00	2.24	3.04	3.55	2.95	1.94	3.85	2.51	3.16	4.03
18	1.00	1.00	2.24	3.04	3.55	3.04	2.37	2.68	2.51	4.34	5.41
19	6.71	5.00	1.41	0.81	6.03	5.82	4.08	4.43	3.01	1.80	0.71
19	6.71	5.00	1.41	0.76	5.93	5.84	4.08	3.94	3.01	3.07	0.71
19	6.71	5.00	1.41	0.74	5.86	5.82	4.07	5.19	3.01	1.80	0.71
19	6.71	7.00	3.61	0.71	6.01	5.84	3.96	3.93	2.02	3.16	0.71
19	5.00	8.00	1.41	0.68	4.89	5.80	3.84	4.12	2.02	2.69	0.71
20	6.71	4.00	4.00	3.42	2.19	2.73	4.63	3.79	6.80	3.40	1.00
20	6.40	4.00	4.00	3.35	3.21	2.59	4.74	2.61	6.80	3.40	1.00
20	6.40	4.00	4.00	3.82	3.23	3.44	4.18	3.28	4.98	3.40	1.00
20	6.40	4.00	4.00	3.83	2.24	3.68	4.04	5.19	4.03	3.40	1.00
20	6.40	4.00	4.00	3.83	3.23	2.92	4.67	3.26	6.80	4.34	1.00
21	1.00	1.00	1.00	0.06	0.43	1.51	1.14	1.84	1.27	0.67	0.71
21	1.00	1.00	1.00	0.06	0.48	1.47	1.12	0.57	3.01	0.67	0.71

Continued on next page

Table E.1 – continued from previous page

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
21	1.00	1.00	1.41	0.06	0.42	1.51	1.11	1.61	1.27	0.67	0.71
21	1.00	1.00	1.00	0.03	0.39	1.51	1.13	1.44	1.27	0.67	0.71
21	1.00	1.00	1.41	0.06	0.43	1.51	1.04	2.91	1.27	0.95	0.71
22	0.00	0.00	0.00	1.89	0.58	0.69	0.59	1.80	1.03	0.00	1.12
22	0.00	4.12	0.00	0.43	0.45	0.26	0.55	1.70	1.03	0.00	1.80
22	0.00	2.24	0.00	0.43	0.88	0.18	0.62	2.61	1.03	0.00	1.12
22	0.00	4.12	0.00	0.42	0.52	0.27	0.68	2.61	1.03	1.05	1.12
22	0.00	1.41	0.00	0.42	0.88	0.30	0.72	1.70	1.03	0.00	0.50
23	4.12	4.12	1.41	3.91	3.83	3.45	3.47	2.33	2.93	0.75	0.71
23	4.12	4.12	1.41	3.91	3.78	3.22	3.31	1.44	1.35	3.68	0.71
23	4.12	4.12	4.24	3.91	3.81	3.15	3.50	1.90	3.51	0.95	1.12
23	4.12	4.12	4.24	1.80	3.45	3.39	3.51	1.60	2.76	2.03	0.71
23	4.12	4.12	1.41	4.00	3.80	3.40	3.47	1.90	1.68	2.98	1.50
24	2.24	3.16	3.16	3.49	2.14	2.10	1.26	0.45	1.03	1.05	0.50
24	2.24	3.16	3.16	3.47	1.02	2.14	2.05	0.89	1.03	0.67	0.50
24	2.24	3.61	4.24	3.46	1.08	2.51	2.15	1.08	1.03	0.67	1.80
24	2.24	3.16	3.16	3.47	1.20	2.52	2.57	0.89	1.03	0.67	3.64
24	1.41	3.16	3.61	3.51	0.78	1.76	1.35	0.89	1.03	0.47	1.80
25	1.00	4.24	0.00	1.75	3.46	2.19	1.10	0.40	0.75	1.05	1.12
25	1.00	5.00	0.00	0.39	2.95	1.38	1.35	0.89	0.75	1.05	1.12
25	1.00	4.24	0.00	0.34	4.58	2.27	1.91	1.65	0.75	1.05	1.12
25	0.00	4.24	0.00	1.61	3.01	2.28	1.31	1.28	0.75	1.05	1.12
25	1.00	4.24	0.00	1.63	3.09	2.28	2.05	1.00	0.75	1.05	1.12
26	5.00	1.41	3.61	0.45	1.66	0.95	1.94	4.95	3.75	1.41	3.00
26	5.00	1.41	1.00	0.44	1.66	1.20	2.06	4.95	3.75	1.95	2.12
26	2.24	1.41	3.61	3.16	1.62	0.74	2.10	5.65	3.75	1.41	3.00
26	5.00	1.41	3.61	3.16	1.65	1.13	2.06	4.71	3.75	0.67	4.61
26	1.41	1.41	5.66	0.44	1.59	1.13	2.15	4.71	2.66	1.41	6.80
27	0.00	2.24	0.00	0.47	1.80	3.54	1.57	5.80	2.57	4.02	1.12
27	0.00	1.00	1.41	0.44	1.96	3.36	1.97	5.22	2.51	4.24	2.92
27	0.00	1.00	0.00	0.50	1.93	2.43	1.70	5.20	2.57	4.02	2.92
27	0.00	1.00	0.00	0.44	1.90	1.97	1.89	5.20	2.36	4.02	2.92
27	0.00	1.00	0.00	0.42	1.96	2.43	1.92	5.89	1.77	4.02	2.92
28	8.00	7.28	2.24	8.38	8.19	3.80	5.09	2.61	5.84	5.59	1.80
28	8.25	7.28	2.24	8.38	6.41	3.74	4.89	3.01	5.84	5.59	1.80
28	8.25	7.28	8.25	8.38	6.44	2.21	4.79	1.34	5.84	5.59	1.80
28	8.25	7.28	2.24	8.37	6.34	3.74	5.25	1.34	5.84	5.59	4.92
28	8.00	7.28	3.00	6.01	6.52	3.71	5.19	2.20	5.84	5.59	1.80
29	7.07	1.00	4.12	1.14	1.76	2.86	1.95	1.80	2.61	2.54	5.85
29	7.28	1.00	3.61	1.81	1.61	3.48	2.17	1.72	2.61	2.54	4.03
29	7.07	7.07	7.07	1.07	1.53	3.40	1.96	2.44	2.93	2.54	4.03
29	7.07	1.00	7.07	1.47	1.54	3.31	1.97	1.00	2.61	2.54	4.03
29	7.28	1.00	4.12	1.14	1.54	3.44	2.02	1.79	2.61	2.54	4.03
30	0.00	1.41	1.00	2.56	1.36	1.69	1.42	0.82	2.57	3.90	2.50
30	0.00	1.41	1.00	2.47	1.30	1.79	1.47	0.57	1.90	3.90	2.24
30	0.00	1.41	1.41	2.55	1.35	1.68	1.40	0.60	2.57	3.48	2.24
30	0.00	1.41	1.00	2.47	1.27	1.72	1.42	0.60	2.57	3.48	2.50
30	0.00	2.00	1.41	2.38	1.33	1.76	1.37	0.57	2.57	3.90	2.50
31	4.47	3.16	2.00	0.97	1.41	1.10	1.10	2.04	1.12	1.37	1.50
31	4.47	3.16	2.00	0.97	1.49	0.86	1.16	3.06	1.12	0.33	1.50
31	4.47	3.16	2.00	0.97	1.44	0.95	1.07	2.24	0.75	0.33	1.50
31	4.24	3.16	2.00	1.03	1.41	0.99	0.75	2.78	1.12	0.33	1.50
31	4.47	3.16	2.00	1.03	1.44	0.63	0.88	2.24	1.12	1.37	1.50
32	3.16	3.16	2.24	1.53	1.43	1.19	0.93	1.52	1.06	0.67	1.58
32	0.00	3.16	2.24	1.47	1.06	1.21	0.75	1.52	1.06	1.20	1.80
32	3.16	3.16	2.24	1.43	1.04	1.02	0.58	1.52	1.06	0.67	0.50
32	0.00	3.16	2.24	1.46	1.02	1.22	0.96	1.81	1.06	1.20	1.00
32	3.16	3.61	2.24	1.47	1.06	0.99	0.96	1.52	1.06	1.20	1.58
33	2.00	1.41	4.47	1.21	2.80	2.33	2.62	0.85	2.30	2.68	5.22
33	3.16	1.41	4.47	1.23	2.99	2.20	2.57	1.00	3.04	2.68	5.22
33	3.16	1.41	4.47	1.17	3.25	2.04	2.52	1.00	3.04	3.73	3.50
33	3.16	1.41	3.61	1.18	2.41	2.16	2.52	1.26	3.04	2.68	3.50
33	1.41	2.00	3.61	1.25	2.42	2.54	2.41	1.26	3.04	2.68	5.22

Continued on next page

Table E.1 – continued from previous page

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
34	5.00	3.61	2.24	1.57	1.24	1.62	1.39	0.63	0.56	0.95	2.00
34	5.00	3.61	2.24	1.65	1.40	1.44	1.37	5.00	1.03	1.37	1.12
34	5.00	2.24	2.24	1.59	1.24	1.63	1.57	3.93	1.06	1.37	2.00
34	5.00	2.24	2.24	1.74	1.25	1.63	1.42	5.06	1.06	1.37	2.00
34	1.41	3.61	2.24	1.54	1.32	1.71	1.49	5.00	1.06	1.70	1.80
35	1.41	1.41	3.61	1.49	1.93	2.62	2.98	3.80	2.25	3.07	1.50
35	1.41	1.41	3.61	1.13	1.96	2.64	2.97	3.85	1.46	2.61	1.50
35	2.00	1.41	3.61	1.54	2.80	2.64	2.98	3.85	2.25	3.07	1.80
35	1.41	1.41	3.61	1.61	1.74	2.61	2.97	3.80	2.25	3.07	1.80
35	1.41	1.41	5.00	1.14	1.75	2.57	2.98	3.80	1.46	3.07	1.50
36	1.00	0.00	8.54	0.84	3.78	3.31	6.89	2.04	4.12	4.38	1.00
36	5.83	3.16	8.54	0.85	4.16	2.60	6.15	2.20	4.12	4.34	1.00
36	1.00	0.00	8.54	0.84	4.44	2.26	6.44	3.16	3.01	6.67	1.00
36	1.00	0.00	8.54	0.84	4.13	2.21	7.24	3.49	5.34	4.38	1.00
36	1.00	0.00	0.00	0.59	3.95	2.27	5.99	3.86	2.30	4.38	1.00
37	2.24	3.16	1.00	0.72	2.58	1.59	1.33	3.86	2.30	1.20	0.71
37	2.24	4.12	1.00	0.72	2.34	1.57	1.32	3.13	2.30	1.20	0.71
37	3.00	4.12	1.00	0.72	2.52	1.56	1.29	4.08	2.30	1.20	0.71
37	2.24	4.12	3.00	4.07	2.46	1.95	1.33	4.31	2.30	1.20	0.71
37	2.24	4.12	3.00	3.70	2.11	1.69	1.39	5.00	2.30	1.20	0.71
38	3.61	1.41	5.00	3.24	5.86	3.11	3.49	4.44	2.61	0.75	3.20
38	3.61	1.41	5.00	3.20	4.84	3.12	3.26	4.20	2.61	0.75	3.20
38	3.61	1.41	5.00	3.29	5.90	3.10	3.03	4.20	2.61	2.40	3.20
38	1.41	2.00	3.16	3.24	5.86	3.11	2.97	4.20	2.61	0.75	3.20
38	3.61	1.41	5.00	3.24	5.00	3.11	3.08	3.01	2.61	2.40	1.58
39	5.83	5.83	1.00	1.33	0.91	1.11	0.43	1.72	1.35	1.37	2.00
39	5.83	5.83	1.00	1.54	1.27	1.10	0.40	2.26	1.25	1.05	2.00
39	5.83	5.83	1.00	1.30	1.03	1.13	0.92	2.26	0.90	1.05	2.00
39	3.16	0.00	1.00	1.27	1.25	1.06	0.36	2.26	0.35	1.37	1.00
39	5.83	5.83	1.00	1.56	1.25	1.08	0.38	2.26	0.50	1.05	2.00
40	2.00	2.00	0.00	0.63	1.57	0.77	1.10	1.89	0.25	0.75	1.12
40	2.00	2.00	0.00	0.67	0.96	0.77	1.18	1.41	0.56	1.20	1.58
40	2.00	2.00	0.00	0.64	2.54	0.77	1.17	2.09	0.25	0.75	1.12
40	2.00	2.00	0.00	3.03	2.56	0.77	1.16	1.41	0.56	0.75	1.12
40	2.00	2.00	0.00	0.76	0.93	0.77	0.77	1.41	0.56	0.75	1.12
41	2.24	2.00	2.24	2.46	0.47	2.60	2.51	0.40	3.36	3.34	2.50
41	2.24	2.00	3.61	2.50	2.00	2.77	2.95	1.13	3.36	3.34	2.50
41	2.24	2.00	3.61	2.52	0.41	2.83	2.66	1.34	2.70	3.34	2.50
41	2.24	2.00	2.24	3.48	0.44	2.78	2.57	0.40	3.36	3.34	2.50
41	2.24	3.61	2.24	2.44	0.78	2.69	2.58	1.26	3.36	3.34	2.50
42	1.41	2.24	4.47	3.10	2.25	2.17	1.66	2.00	1.77	2.00	2.50
42	1.41	3.61	4.47	2.14	2.25	2.23	1.77	1.80	2.00	2.00	2.50
42	4.47	4.24	4.47	2.32	1.83	1.90	1.70	1.20	2.00	2.00	2.50
42	4.47	2.24	4.47	2.32	2.08	2.23	1.91	1.20	2.00	2.00	2.50
42	4.47	3.61	1.41	2.33	2.06	2.33	1.88	2.00	1.77	2.00	1.80
43	1.41	1.41	1.41	0.75	1.88	3.69	1.00	0.89	1.12	1.20	0.71
43	0.00	1.41	1.41	0.71	1.88	3.85	0.93	2.01	1.12	1.20	0.71
43	1.41	1.41	1.41	0.66	1.87	3.36	0.97	2.61	1.12	1.20	0.71
43	0.00	1.41	1.41	0.66	1.84	3.89	1.00	1.40	1.12	1.20	0.71
43	0.00	1.41	1.41	0.83	1.84	3.85	0.91	2.00	1.12	1.20	0.71
44	3.16	3.16	3.16	4.43	5.33	1.92	5.71	3.06	3.01	5.38	3.54
44	6.32	1.41	3.16	4.01	5.04	1.73	5.65	2.41	4.47	4.96	6.50
44	6.32	1.41	7.28	4.35	5.02	1.66	5.72	2.81	3.01	5.38	6.50
44	4.12	1.41	1.00	2.92	4.94	1.75	5.80	3.01	4.47	5.38	3.54
44	1.00	1.41	7.28	3.90	4.15	1.75	5.76	3.61	4.47	5.39	3.54
45	3.61	3.00	3.00	2.05	3.60	3.53	2.68	5.76	3.01	2.69	4.27
45	4.47	3.00	3.00	3.92	2.26	3.19	2.44	5.37	2.02	2.75	4.00
45	4.47	1.00	3.00	2.05	3.94	3.53	2.58	4.68	3.01	2.00	4.03
45	1.41	3.00	3.00	3.78	3.64	3.85	2.60	5.41	1.82	3.07	4.03
45	1.41	3.00	3.00	3.72	3.40	3.93	2.51	5.19	1.82	3.07	4.00
46	2.24	8.06	1.00	0.68	6.76	1.72	1.60	3.62	1.68	1.37	3.61
46	2.24	8.06	1.00	0.49	6.92	1.40	1.55	3.62	1.68	1.88	2.12
46	2.24	5.39	1.00	1.00	6.76	1.42	1.75	4.51	1.68	3.15	2.12

Continued on next page

Table E.1 – continued from previous page

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
46	2.24	2.24	1.00	1.48	6.91	1.75	1.63	4.24	1.68	1.88	1.12
46	2.24	5.39	1.00	5.05	6.86	1.77	1.83	4.24	1.68	2.98	2.12
47	1.00	1.00	1.00	0.45	0.19	0.70	0.45	2.41	1.60	0.75	2.50
47	1.00	1.00	1.00	0.43	0.20	0.24	0.49	3.31	1.60	0.00	2.50
47	1.41	1.00	1.00	0.40	0.16	0.71	0.42	3.05	1.60	0.75	2.50
47	1.41	1.00	1.00	0.46	0.17	0.82	0.20	3.44	1.60	0.75	2.50
47	7.21	1.00	1.00	0.45	0.21	0.82	0.44	3.44	1.60	0.75	2.50
48	3.16	3.61	2.24	0.52	1.96	1.06	1.57	1.90	1.46	2.11	2.92
48	3.16	4.47	5.66	0.53	2.05	1.04	1.70	2.01	1.46	2.11	2.92
48	3.16	3.61	5.66	0.52	1.96	1.05	1.70	2.04	1.46	2.11	2.92
48	4.47	3.61	5.66	0.53	1.96	1.09	1.75	2.04	1.46	2.11	2.92
48	3.16	4.47	5.66	0.53	1.87	1.07	1.71	2.04	1.46	2.11	2.92
49	0.00	2.83	2.24	3.30	1.87	2.09	2.48	3.82	1.60	2.42	3.54
49	0.00	2.83	2.24	3.15	0.93	2.02	2.62	3.26	1.27	2.42	2.83
49	0.00	0.00	2.24	2.07	0.98	1.97	2.18	3.69	1.60	2.42	2.92
49	0.00	2.83	2.24	2.08	0.95	2.02	2.66	3.41	1.27	2.24	2.92
49	0.00	2.24	2.24	2.02	1.14	2.32	2.53	3.69	1.58	2.42	2.92
50	2.00	4.24	5.39	3.54	0.23	0.43	1.60	2.00	3.16	2.87	4.03
50	2.00	2.00	5.39	3.50	0.12	0.23	1.61	3.11	3.35	2.24	4.03
50	2.24	5.39	5.39	3.51	0.59	0.49	1.67	3.61	3.82	2.24	3.81
50	5.66	2.00	5.39	3.49	0.56	0.60	1.58	3.28	3.61	2.87	3.81
50	5.66	2.00	5.39	3.48	0.14	0.60	1.71	2.61	3.61	1.95	4.03
51	0.00	0.00	0.00	0.74	0.03	0.35	0.47	1.71	0.90	0.33	0.71
51	0.00	0.00	4.24	0.72	0.00	0.36	0.46	1.28	1.46	0.33	0.71
51	0.00	0.00	4.24	0.68	0.00	0.31	0.74	2.42	1.25	0.33	0.71
51	0.00	0.00	4.24	0.74	0.00	0.28	0.46	2.24	1.46	0.33	0.71
51	0.00	0.00	0.00	0.75	0.00	0.52	0.46	2.16	1.25	0.33	0.71
52	1.00	1.41	1.41	0.41	0.24	0.42	0.58	3.16	0.50	0.47	0.50
52	1.00	1.41	1.41	1.67	0.22	0.36	0.74	3.26	0.50	1.49	1.80
52	1.00	0.00	1.41	0.40	0.22	0.36	0.63	3.01	0.50	1.49	0.50
52	1.00	0.00	1.41	0.41	0.23	0.47	0.73	2.42	0.50	0.47	1.80
52	1.00	0.00	1.41	0.37	0.23	0.36	0.88	1.97	0.50	0.47	0.50
53	1.00	3.16	3.16	1.51	1.00	0.82	0.41	3.56	2.02	2.03	1.12
53	3.16	3.16	3.16	1.57	1.00	0.88	0.25	3.61	3.16	2.11	1.58
53	1.00	3.16	1.00	5.59	1.00	0.85	0.48	2.56	3.16	2.11	1.12
53	3.16	3.16	3.16	5.51	1.00	0.88	0.37	4.05	3.16	2.11	1.12
53	1.00	3.16	3.16	3.04	1.00	0.88	0.42	5.01	3.16	2.11	1.12
54	3.00	3.16	0.00	3.37	6.27	4.68	4.05	2.42	2.85	1.49	4.12
54	3.00	3.16	0.00	4.31	6.27	4.64	4.10	3.82	2.26	3.80	5.02
54	3.00	3.16	8.06	3.57	6.23	4.67	3.52	2.41	2.26	0.47	5.50
54	3.00	3.16	0.00	4.12	6.27	4.69	4.12	3.72	2.75	3.07	4.12
54	3.00	3.16	8.06	4.12	3.71	4.71	3.34	4.84	2.26	3.43	5.50
55	2.24	1.41	3.61	3.82	1.72	2.32	2.09	4.69	2.15	3.34	2.12
55	2.24	1.41	3.61	4.12	1.74	2.43	2.32	4.74	4.16	3.54	2.50
55	2.24	1.41	2.24	3.99	1.73	2.63	2.14	5.39	2.24	3.54	2.50
55	2.24	1.41	2.83	3.91	1.74	2.60	2.18	4.94	2.15	3.54	2.12
55	2.24	1.41	3.61	3.35	1.75	2.42	2.33	4.67	2.15	3.54	2.12
56	5.00	0.00	0.00	2.25	2.04	2.57	2.85	6.49	2.46	1.66	1.80
56	5.00	0.00	0.00	2.64	2.17	2.57	2.91	4.53	2.46	1.88	3.64
56	5.00	1.00	0.00	3.99	1.97	2.47	3.02	3.69	0.79	2.13	6.52
56	5.00	1.00	0.00	2.25	1.97	2.56	3.01	3.82	5.10	2.13	6.52
56	5.00	0.00	0.00	2.09	1.97	2.51	2.79	3.82	2.46	2.13	6.52
57	2.24	2.24	4.47	1.83	0.82	1.99	1.41	5.52	1.82	2.54	2.92
57	2.24	2.24	4.47	2.20	0.81	1.93	1.24	4.67	2.12	2.54	2.69
57	2.24	2.24	4.47	1.41	0.75	1.93	1.35	4.67	1.82	2.54	2.69
57	2.24	2.24	4.47	2.76	0.77	1.93	1.37	6.08	2.12	2.54	2.69
57	2.24	2.24	4.47	1.80	0.79	1.97	1.49	6.08	1.25	2.54	2.69
58	5.39	5.39	4.24	4.36	2.73	3.89	3.10	2.88	3.09	2.87	2.24
58	5.83	7.07	4.24	6.17	2.93	3.37	3.01	4.00	3.09	2.87	2.24
58	7.81	1.41	0.00	6.13	2.90	3.27	2.98	2.86	3.09	2.87	2.24
58	7.07	1.41	4.24	5.27	2.92	3.42	2.99	2.86	3.09	2.87	2.24
58	5.39	1.41	0.00	6.53	2.88	3.35	3.09	3.28	3.09	2.87	2.24
59	5.00	1.41	5.83	4.08	1.86	1.58	1.74	1.72	3.01	2.40	2.69

Continued on next page

Table E.1 – continued from previous page

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
59	3.00	1.41	3.61	4.03	1.84	1.44	1.25	1.26	3.36	2.40	3.91
59	5.00	1.41	5.83	3.58	1.84	1.82	1.78	3.13	3.36	2.40	2.50
59	5.00	1.41	5.83	3.66	1.84	1.82	1.92	0.72	3.36	2.40	2.69
59	5.00	1.41	5.83	3.19	1.84	1.55	1.21	0.72	3.72	2.40	2.69
60	0.00	0.00	0.00	1.70	1.43	1.13	0.88	2.00	0.79	2.40	2.00
60	0.00	0.00	0.00	1.72	1.43	1.10	0.88	2.28	0.79	2.40	2.00
60	0.00	0.00	0.00	1.17	1.43	1.16	0.89	2.44	0.79	2.40	3.16
60	0.00	0.00	0.00	1.72	1.43	1.18	0.89	2.44	0.79	3.15	1.50
60	0.00	0.00	0.00	1.14	1.43	1.10	0.86	2.28	0.79	3.15	3.16
61	1.00	1.00	1.00	1.37	1.01	1.06	0.96	3.61	1.58	1.37	2.06
61	1.00	1.00	1.41	0.59	1.04	1.14	0.95	2.61	1.80	1.37	3.00
61	1.00	1.00	1.00	0.54	1.06	1.10	0.94	3.11	1.80	0.75	2.06
61	1.00	1.00	1.00	0.99	1.02	1.12	0.98	2.41	2.46	1.05	3.00
61	1.00	1.00	1.00	1.61	1.00	1.11	1.00	3.11	2.46	1.05	1.12
62	7.62	5.10	2.83	5.61	4.49	6.15	4.00	2.44	5.83	4.24	1.80
62	3.00	5.10	2.83	5.42	4.59	4.60	2.12	2.41	6.95	4.24	3.54
62	7.62	1.41	2.83	5.48	5.08	6.06	2.09	2.34	5.59	4.27	2.00
62	1.41	1.41	2.83	5.45	5.16	6.09	2.11	2.34	6.95	4.24	1.80
62	5.00	4.12	2.83	5.55	5.03	6.09	2.10	2.41	6.95	4.24	1.80
63	3.61	8.60	3.61	3.84	2.52	3.93	3.12	4.83	3.36	3.34	3.81
63	3.61	8.60	3.61	3.82	2.57	4.13	3.15	3.54	3.36	3.43	3.81
63	3.61	4.12	3.61	3.85	2.46	4.07	3.07	4.22	2.70	3.43	1.80
63	3.61	8.60	3.61	3.84	2.66	3.86	2.63	3.33	3.36	3.34	3.54
63	6.40	8.60	3.61	3.85	2.52	4.20	2.53	3.00	3.36	5.37	3.81

Table E.2: Residential Test Data

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
1	7.21	8.60	8.60	5.77	6.72	6.04	5.42	6.50	6.13	6.66	7.27
1	7.21	7.81	8.60	6.95	7.20	6.46	6.43	3.54	4.22	3.95	4.12
1	7.21	7.21	8.60	6.80	7.01	7.09	7.21	5.22	4.12	4.92	5.10
1	7.21	7.21	5.39	6.95	6.57	6.80	7.03	6.73	7.21	6.37	7.55
1	8.60	5.39	8.60	7.07	7.20	7.29	7.17	5.00	6.01	5.15	6.28
2	1.00	7.07	2.83	2.49	2.01	2.53	2.31	7.21	6.62	6.05	5.16
2	1.00	4.47	2.83	2.51	2.10	2.52	2.56	7.21	6.62	6.05	5.44
2	4.47	4.47	2.83	2.54	2.09	2.60	2.88	5.70	6.62	5.67	4.53
2	1.00	4.47	4.47	2.96	3.37	3.30	3.76	7.21	6.62	6.72	5.94
2	3.00	7.07	4.47	3.01	2.98	3.28	2.81	3.54	3.80	3.61	4.56
3	6.40	1.41	2.24	1.83	2.08	2.89	2.75	2.50	4.07	3.75	4.24
3	3.61	1.41	2.83	1.86	3.00	2.32	2.54	3.20	2.60	2.36	2.44
3	2.83	7.21	2.83	4.53	3.24	2.46	2.74	2.83	2.36	3.40	3.96
3	2.83	7.21	2.24	1.85	3.06	1.78	2.29	4.47	4.07	3.91	2.84
3	2.83	1.41	2.83	1.36	1.43	2.51	2.73	2.50	4.07	3.40	3.22
4	1.00	3.61	2.00	1.57	1.93	2.05	1.87	0.50	2.36	1.46	1.00
4	1.00	1.00	2.00	1.49	1.64	2.33	1.89	3.91	2.36	2.30	2.41
4	3.16	2.24	2.24	1.50	1.64	2.35	1.57	1.12	1.20	2.47	2.28
4	3.16	3.16	2.24	1.53	1.70	2.17	2.32	1.12	1.05	1.25	1.71
4	3.16	3.16	2.00	1.48	2.28	1.88	2.37	1.80	2.43	3.40	3.54
5	1.00	5.00	4.00	2.01	2.42	1.52	2.09	2.50	2.13	2.85	3.05
5	1.00	2.24	5.00	2.84	2.72	4.01	4.23	2.50	2.13	2.36	2.51
5	2.24	3.61	3.61	4.27	3.47	4.11	3.87	2.50	2.13	1.25	2.00
5	2.24	2.24	2.24	3.38	3.24	3.84	3.60	1.58	2.13	2.30	1.56
5	2.24	2.24	0.00	3.54	3.77	4.30	3.82	2.50	2.13	2.83	2.88
6	3.61	0.00	3.00	1.01	0.49	0.69	1.37	1.80	2.00	1.52	2.00
6	3.61	3.61	0.00	0.60	1.31	2.29	3.21	2.69	2.60	2.14	0.72
6	3.61	3.00	5.39	1.34	2.31	2.17	2.20	1.41	0.94	1.27	1.61
6	3.00	3.16	2.00	1.34	1.37	0.76	2.05	1.12	0.94	0.50	2.28
6	3.00	3.00	3.16	0.56	0.84	1.18	1.38	2.24	0.94	1.35	2.04
7	1.00	1.00	2.83	2.00	1.90	2.02	1.85	1.58	1.37	2.02	2.09
7	2.24	2.83	6.32	1.46	2.19	2.39	1.82	1.58	1.67	1.06	1.13
7	1.00	4.12	6.32	2.70	2.83	2.32	1.72	0.00	1.05	1.00	2.41

Continued on next page

Table E.2 – continued from previous page

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
7	2.83	6.32	1.41	1.89	1.45	2.06	1.80	1.58	1.37	1.00	1.00
7	1.00	3.16	1.00	0.71	2.51	1.84	1.95	1.80	1.41	1.25	1.56
8	4.12	1.00	2.24	2.66	2.59	3.28	2.94	0.71	1.49	2.14	1.84
8	2.24	4.00	4.12	2.54	4.36	1.66	2.41	2.00	0.75	1.35	0.72
8	3.16	2.24	3.00	3.89	1.49	1.79	2.01	2.69	1.80	0.79	0.63
8	1.41	3.00	3.16	2.50	2.08	2.43	3.52	3.16	1.94	2.80	2.63
8	4.12	5.10	1.00	2.68	2.45	2.21	1.90	1.58	1.89	2.14	2.47
9	2.00	1.41	1.41	2.46	2.49	2.18	2.75	2.50	2.69	2.15	2.28
9	2.00	3.16	2.00	2.26	2.08	2.29	1.90	1.50	3.73	3.25	3.16
9	1.41	1.00	2.00	1.17	1.30	1.23	1.15	2.50	2.69	2.66	2.88
9	1.41	1.41	2.00	1.63	1.23	1.33	1.34	0.50	1.20	2.80	1.26
9	1.00	1.00	1.41	1.41	2.02	2.25	2.08	3.54	3.33	2.47	2.28
10	1.41	1.41	1.00	2.91	1.91	2.58	1.96	3.35	2.60	2.02	1.89
10	3.61	3.61	3.61	1.67	1.98	2.10	2.26	2.50	2.98	3.01	2.72
10	6.40	8.06	8.06	1.62	1.79	1.94	2.24	2.92	2.69	2.30	3.05
10	6.40	8.06	8.06	2.78	2.50	2.45	3.12	2.50	0.94	1.41	2.00
10	3.61	3.61	8.06	2.18	2.09	2.58	3.11	2.92	2.36	3.95	1.89
11	6.40	7.07	5.00	2.79	2.81	3.43	2.93	2.69	2.69	2.26	2.83
11	3.61	7.21	2.00	2.66	3.51	3.35	3.47	1.12	1.89	2.36	1.08
11	6.71	7.21	2.00	2.24	2.67	3.37	3.40	2.12	3.77	3.54	2.56
11	7.21	7.21	2.24	0.61	0.41	0.17	0.40	5.32	3.54	2.66	2.83
11	0.00	0.00	1.00	1.66	1.64	2.24	2.73	2.12	3.77	3.18	2.56
12	1.00	1.00	0.00	3.00	3.32	2.93	2.49	3.00	3.28	2.46	3.05
12	1.00	2.00	0.00	3.27	2.71	2.51	2.30	0.71	0.75	1.27	1.22
12	1.00	2.00	2.00	0.04	1.44	0.86	1.94	3.00	2.03	2.50	2.51
12	1.00	1.00	1.00	1.19	0.37	1.03	1.28	1.12	2.24	2.06	2.40
12	1.00	1.00	2.24	2.92	1.52	1.39	1.07	1.80	1.80	2.30	2.00
13	1.00	0.00	3.00	0.81	0.76	0.94	0.63	2.50	2.00	1.12	1.89
13	2.00	1.00	3.00	2.73	2.33	2.26	2.05	2.00	2.03	1.77	0.80
13	3.00	3.00	3.00	2.48	1.65	2.00	1.95	2.50	2.00	1.35	1.02
13	3.00	3.00	2.00	2.58	1.62	1.39	1.23	2.69	2.75	2.51	1.40
13	3.00	3.00	3.00	2.71	2.06	2.18	1.76	1.58	2.03	1.27	1.22
14	4.00	4.00	3.61	2.46	2.57	2.60	2.56	4.12	4.35	2.83	1.65
14	3.00	3.00	3.61	2.52	3.37	2.92	3.30	2.50	3.33	2.12	2.51
14	2.83	3.61	3.61	3.24	3.23	2.91	3.15	2.50	3.33	3.35	2.33
14	2.83	5.66	1.00	3.14	3.40	3.70	2.42	3.54	3.67	3.25	3.45
14	2.24	2.24	3.61	3.35	3.52	3.24	3.80	2.50	2.43	1.35	1.84
15	1.41	1.41	1.41	1.07	1.31	1.44	1.56	4.12	1.80	2.61	2.33
15	1.41	2.24	1.41	1.13	1.25	1.35	1.41	4.00	2.69	2.93	2.33
15	2.00	2.24	1.41	3.15	1.74	1.74	1.61	2.50	1.94	1.77	1.84
15	1.41	1.41	1.41	1.63	1.70	1.67	2.23	4.00	2.69	2.93	2.61
15	1.41	5.83	5.83	1.58	1.76	1.46	1.76	2.50	2.85	2.24	2.61
16	1.00	2.00	2.00	1.42	1.39	1.28	0.99	3.54	2.33	1.35	0.80
16	1.00	1.00	2.00	1.26	1.07	1.02	0.87	1.58	0.75	1.46	0.89
16	1.00	1.00	3.61	1.48	1.10	0.96	0.86	1.50	1.70	1.35	1.02
16	1.00	3.61	3.61	3.33	2.68	2.64	2.23	1.50	0.47	0.90	0.72
16	1.00	2.00	1.00	2.90	3.29	1.75	0.76	1.50	1.00	0.71	0.45
17	0.00	3.16	0.00	5.02	3.46	3.52	1.66	1.50	1.05	1.50	1.20
17	0.00	0.00	1.00	0.95	0.71	3.69	2.06	0.50	1.00	0.79	0.57
17	0.00	3.00	1.41	0.91	0.44	0.40	0.43	1.80	1.05	1.50	1.41
17	1.41	1.00	0.00	0.06	0.30	0.16	0.29	1.50	2.03	1.50	1.41
17	0.00	1.00	1.00	1.50	0.88	0.65	0.55	3.00	3.35	2.37	2.41
18	1.00	2.24	2.24	1.59	1.85	1.80	1.91	0.50	0.75	1.82	1.84
18	1.00	2.24	2.24	1.58	1.40	1.55	1.26	1.12	1.67	1.41	1.61
18	1.00	1.00	1.00	1.54	0.50	1.42	1.59	1.12	0.75	1.12	2.16
18	1.00	1.00	1.00	1.08	0.80	0.73	1.13	1.58	1.89	2.46	1.56
18	1.00	0.00	1.00	0.56	0.94	0.98	1.08	1.80	1.67	1.90	2.00
19	3.16	4.12	4.12	3.66	2.89	2.44	2.48	3.64	3.14	3.55	2.34
19	4.12	4.12	7.62	1.52	2.48	1.72	2.18	3.91	3.89	3.55	2.56
19	1.00	4.12	2.24	1.75	1.73	1.70	1.15	2.06	3.14	2.57	2.42
19	4.12	0.00	0.00	0.40	0.71	0.62	1.36	2.83	3.14	2.57	2.42
19	4.12	4.12	2.00	1.90	1.66	1.91	1.96	1.41	1.70	1.25	1.34
20	4.12	0.00	0.00	3.33	3.02	2.08	1.89	1.58	1.37	1.12	1.41

Continued on next page

Table E.2 – continued from previous page

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
20	0.00	3.16	3.16	1.92	1.54	1.88	1.59	2.06	0.94	0.50	0.82
20	3.16	0.00	3.16	2.50	1.72	2.11	2.08	1.12	0.33	1.03	0.82
20	2.24	0.00	0.00	0.52	0.70	1.12	1.54	0.50	1.20	0.50	0.82
20	0.00	0.00	0.00	0.64	0.42	0.61	0.45	1.58	1.37	1.00	1.17
21	0.00	0.00	0.00	1.06	0.77	1.23	1.12	1.80	1.20	1.12	1.41
21	0.00	3.00	0.00	0.88	0.56	0.85	0.88	1.80	0.75	1.46	1.28
21	0.00	0.00	0.00	1.09	0.52	0.44	0.55	2.12	0.47	0.56	0.60
21	0.00	0.00	0.00	0.39	0.25	0.69	0.95	0.50	0.67	1.06	0.89
21	0.00	0.00	0.00	0.27	0.59	0.85	1.06	0.00	0.75	0.79	1.26
22	3.61	2.24	3.61	2.36	1.95	2.21	1.09	2.55	3.16	2.14	1.84
22	2.24	3.61	2.24	1.09	2.82	1.83	2.13	1.12	1.80	1.27	1.81
22	4.12	2.24	4.47	2.50	1.52	2.85	1.94	0.50	1.80	0.79	0.72
22	3.61	4.00	3.61	3.62	0.78	1.45	1.92	1.00	1.20	1.12	0.82
22	3.61	2.24	3.61	3.58	1.35	1.62	1.72	1.50	2.11	1.27	1.26
23	4.47	3.16	4.47	1.41	1.22	1.02	1.02	2.24	2.98	3.55	3.39
23	4.47	4.47	4.47	1.00	0.98	1.47	0.88	3.64	4.03	3.02	3.40
23	4.47	2.24	4.47	3.11	1.98	1.13	2.47	3.64	2.13	2.85	3.49
23	2.00	3.16	2.24	0.62	1.63	2.01	1.56	3.91	3.43	3.16	3.39
23	2.83	2.83	2.83	0.88	1.37	1.11	1.97	3.91	3.28	3.20	2.95
24	2.24	2.24	4.47	2.24	1.12	1.42	0.76	2.83	2.87	0.90	0.72
24	1.41	1.41	3.16	1.50	0.46	1.49	1.58	1.80	1.49	0.90	1.17
24	2.24	4.47	3.16	1.46	3.72	2.88	1.88	2.24	1.67	1.60	2.72
24	2.24	3.16	2.24	1.55	2.28	2.38	2.21	3.04	1.89	1.68	1.97
24	3.16	4.47	3.16	3.28	2.73	3.08	2.31	2.24	2.03	2.25	2.63
25	4.24	1.41	4.24	2.47	1.90	1.65	0.68	1.50	1.37	0.79	0.63
25	2.00	1.00	2.00	0.46	0.84	0.37	0.45	0.50	1.00	1.35	1.44
25	1.00	2.00	1.00	0.46	1.40	0.85	0.67	0.50	1.33	0.90	2.51
25	2.24	3.61	1.00	0.54	2.32	1.95	2.22	1.12	0.75	0.35	0.28
25	2.00	2.00	2.00	1.05	1.12	2.04	1.98	0.50	0.75	0.71	0.45
26	1.41	2.24	1.41	1.16	1.97	1.66	1.53	3.35	3.00	3.05	3.69
26	2.00	1.41	2.24	1.12	1.03	1.05	1.01	1.50	1.49	0.75	1.65
26	1.41	2.24	2.24	0.77	1.07	1.32	1.49	2.06	1.67	1.35	0.80
26	1.41	1.00	1.00	1.04	0.41	0.37	1.57	3.81	1.20	1.35	0.80
26	2.24	1.00	1.41	3.06	1.32	0.44	0.50	4.74	2.75	2.55	2.00
27	2.00	2.00	1.00	0.53	0.63	0.93	1.01	1.12	0.67	0.90	1.00
27	1.00	1.00	1.00	0.56	0.62	0.99	1.07	1.12	1.67	1.12	1.00
27	1.00	1.00	2.00	1.13	0.77	1.11	1.03	1.12	1.67	1.12	1.00
27	2.00	2.00	1.00	0.49	1.10	1.14	0.59	2.06	1.33	1.12	2.01
27	8.25	2.83	2.83	0.93	0.64	0.87	0.96	2.06	1.33	1.12	1.79
28	2.83	2.83	2.83	2.00	1.64	1.43	1.49	2.83	3.07	2.06	2.34
28	2.83	2.83	2.83	1.92	1.52	1.40	1.64	2.83	1.67	1.95	1.28
28	2.83	2.83	2.83	2.07	2.22	1.96	1.73	1.50	1.67	1.95	1.84
28	2.83	2.83	2.83	1.98	1.87	1.77	1.70	2.50	2.24	1.35	1.84
28	2.83	2.83	2.83	1.89	1.67	1.57	1.75	1.50	1.67	2.76	2.33
29	3.16	3.16	1.41	0.46	0.84	0.85	1.04	2.55	0.67	1.03	1.41
29	1.00	3.16	3.16	1.21	1.71	1.78	1.52	1.50	0.67	1.35	1.81
29	3.16	3.16	3.16	1.16	1.22	0.81	0.42	0.50	0.00	1.03	0.60
29	1.41	2.24	2.24	1.05	1.57	1.26	1.14	1.12	0.00	0.56	1.08
29	2.24	1.00	2.24	1.63	1.03	1.62	1.76	0.50	1.37	1.50	1.22
30	2.00	2.00	2.00	1.79	1.97	1.63	1.59	2.50	1.67	0.25	0.63
30	2.00	2.00	2.24	2.50	1.30	2.38	1.96	1.58	1.00	0.25	1.22
30	2.00	2.00	2.24	1.53	2.66	1.65	1.79	1.12	1.00	0.25	0.63
30	2.00	2.00	3.00	2.51	2.71	2.14	1.74	1.58	0.47	0.25	0.63
30	2.00	2.00	2.00	3.04	2.33	1.61	1.84	2.24	1.89	1.95	1.52
31	3.16	1.41	1.41	3.52	1.90	1.89	1.97	1.41	0.94	1.00	1.46
31	1.41	1.41	3.16	1.58	2.84	1.66	1.21	1.41	0.75	1.25	1.17
31	2.24	2.00	3.00	1.45	1.69	1.72	1.80	1.41	0.75	0.56	1.46
31	1.41	3.16	1.41	3.02	2.63	2.37	1.84	1.50	0.75	2.25	2.42
31	3.16	2.24	3.16	2.48	1.81	2.67	2.18	2.55	0.94	2.00	1.34
32	3.00	2.24	2.24	0.78	0.45	0.91	0.80	2.69	1.00	2.06	1.00
32	4.12	2.24	0.00	1.40	0.45	1.80	1.23	2.92	2.60	1.90	1.61
32	4.12	2.24	2.24	1.07	0.37	0.62	1.05	2.24	2.60	2.61	2.72
32	4.12	0.00	0.00	1.12	0.47	1.94	0.53	2.83	2.60	2.85	2.24

Continued on next page

Table E.2 – continued from previous page

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
32	4.12	0.00	3.00	0.54	1.34	2.17	1.03	2.06	2.03	2.61	1.70
33	4.47	4.47	4.47	0.91	2.15	1.61	1.84	0.50	0.33	0.71	2.53
33	4.47	4.47	4.47	4.37	2.54	2.45	2.59	1.80	1.67	2.66	2.13
33	4.47	4.47	3.61	2.24	4.18	3.20	2.92	1.00	0.75	1.25	1.20
33	4.47	5.10	3.61	4.03	2.29	2.33	2.22	3.20	3.90	3.29	3.23
33	4.47	5.10	3.61	3.22	3.27	1.83	2.11	1.12	1.05	3.51	2.56
34	1.00	1.00	2.24	0.66	0.82	0.47	0.50	2.92	3.33	1.90	1.02
34	2.24	2.83	2.83	1.29	1.32	1.21	1.03	3.54	2.75	2.51	2.88
34	2.24	2.24	2.24	2.03	1.66	0.92	1.34	1.50	3.16	2.50	2.41
34	2.24	2.24	2.24	0.81	1.06	1.92	1.05	2.69	2.75	2.50	2.47
34	2.83	2.24	2.24	0.28	0.80	1.17	1.04	1.50	2.24	1.95	2.72
35	4.47	7.07	1.41	3.04	5.52	2.99	4.05	2.12	2.36	1.52	2.01
35	5.83	1.41	1.41	2.53	1.89	1.40	1.97	1.80	2.36	1.58	1.26
35	6.32	2.24	4.24	2.64	1.87	0.93	0.84	1.00	1.37	2.30	1.56
35	1.41	1.41	1.41	1.01	1.62	1.94	1.61	3.04	0.47	0.79	0.20
35	1.41	1.41	1.41	1.02	1.64	1.68	0.88	4.12	2.67	2.24	2.00
36	2.00	5.00	2.00	2.95	2.35	2.11	2.24	3.54	2.60	1.95	1.71
36	2.00	2.00	1.00	0.24	0.09	0.73	1.05	2.69	1.80	1.52	1.97
36	1.00	1.00	1.00	2.02	1.34	1.05	0.92	2.24	0.94	1.12	1.02
36	1.00	1.00	1.00	0.59	0.96	1.87	1.25	1.80	1.67	1.58	1.84
36	1.00	3.61	1.00	0.31	0.44	1.04	1.21	1.50	0.94	1.06	1.34
37	2.24	2.83	2.83	1.49	2.28	2.88	3.04	2.50	2.36	2.50	1.97
37	2.24	4.12	2.83	2.96	2.36	2.52	2.90	2.50	2.36	1.68	1.97
37	2.24	2.24	2.83	3.02	2.45	2.43	3.07	2.50	2.36	1.68	1.97
37	2.24	2.83	2.83	3.05	2.53	2.56	2.85	2.50	2.36	2.36	1.97
37	2.24	2.83	4.12	2.89	2.60	2.55	2.15	2.50	1.70	1.68	1.34
38	1.41	1.00	1.41	0.64	2.20	2.05	1.67	1.00	1.05	0.35	0.63
38	1.41	1.41	1.41	1.52	1.94	1.32	0.83	2.06	1.37	0.35	1.22
38	1.41	1.41	2.24	2.11	1.12	1.57	2.30	1.00	0.00	1.25	0.82
38	1.41	1.41	2.24	0.68	0.30	0.95	1.21	1.00	0.00	0.35	0.40
38	1.41	1.00	2.24	1.97	2.59	2.32	0.90	1.00	0.00	0.35	1.61
39	2.24	2.00	1.41	1.58	1.70	1.57	1.47	2.06	1.05	2.15	2.86
39	2.24	2.24	1.41	1.44	1.56	1.34	1.17	1.58	1.67	1.25	0.63
39	2.00	2.24	1.41	1.57	1.65	1.47	1.39	1.58	1.94	1.60	1.44
39	2.00	1.00	1.41	1.37	1.57	1.27	1.23	1.58	1.94	2.15	1.26
39	2.24	1.00	2.00	1.60	1.66	1.61	1.41	1.58	1.94	1.82	2.24
40	2.83	2.83	2.83	2.45	2.05	2.05	1.98	5.70	4.63	3.02	2.34
40	2.83	2.83	2.83	2.52	1.69	2.02	2.11	1.80	0.67	1.52	2.34
40	2.83	2.83	2.83	1.99	1.91	1.76	1.68	4.03	4.07	4.14	4.47
40	2.83	1.00	2.83	1.90	1.57	1.44	1.31	1.58	1.20	1.75	2.34
40	2.83	2.83	2.83	1.80	1.48	1.26	1.33	1.58	1.20	1.75	2.34
41	5.00	2.24	4.00	3.02	2.75	2.58	2.39	1.41	1.37	0.75	1.41
41	5.00	3.61	4.00	3.38	3.05	4.00	3.79	0.71	1.05	1.27	1.40
41	4.47	4.00	4.00	3.10	2.30	2.19	2.47	0.71	0.75	2.25	1.40
41	4.47	5.00	5.00	3.56	3.56	3.39	3.12	1.41	1.37	0.35	0.57
41	5.00	5.00	5.00	4.00	4.57	4.21	3.90	1.41	1.05	1.25	0.63
42	4.24	2.24	3.00	0.96	1.19	0.30	0.68	2.50	2.54	2.85	1.56
42	3.00	2.24	1.00	1.76	0.49	0.11	0.49	3.61	2.60	1.46	1.56
42	4.24	2.24	1.00	1.85	0.82	1.05	1.28	1.12	2.24	1.12	1.52
42	4.24	2.24	3.00	1.66	0.63	0.19	0.15	3.61	2.60	2.50	2.00
42	4.24	2.24	3.00	0.50	0.47	0.22	0.75	0.71	0.94	1.46	1.56
43	4.00	4.00	4.00	3.67	2.99	2.65	2.72	0.50	2.13	2.83	2.28
43	4.00	4.00	4.00	2.46	2.27	2.15	2.21	0.50	2.13	2.83	2.24
43	4.00	4.00	4.00	3.80	3.21	3.07	2.66	1.41	2.75	2.30	2.24
43	4.00	4.00	4.00	2.90	2.27	2.16	2.34	0.50	2.13	1.35	1.98
43	4.00	4.00	4.00	2.70	2.55	2.39	2.42	0.50	0.75	1.25	1.08
44	1.00	3.16	0.00	0.48	2.20	2.20	2.05	3.81	3.40	3.51	3.61
44	3.00	1.41	3.00	3.60	3.45	3.20	3.69	2.06	2.75	2.93	3.10
44	1.00	4.12	4.12	2.99	3.56	2.65	2.83	3.81	4.27	3.91	3.01
44	3.00	4.12	2.83	1.67	2.06	3.77	3.26	5.32	3.61	2.70	2.84
44	4.12	4.12	3.61	4.58	3.14	1.92	1.84	4.74	3.80	3.34	4.12
45	3.16	1.00	0.00	0.71	1.96	1.96	2.04	2.50	1.33	3.18	2.15
45	3.00	3.00	4.47	3.53	3.63	3.05	2.84	1.50	1.67	2.30	2.53

Continued on next page

Table E.2 – continued from previous page

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
45	3.16	3.16	4.47	3.79	3.26	3.17	2.76	0.50	1.05	1.77	2.33
45	3.00	3.00	4.47	3.67	3.37	2.63	2.42	0.50	2.87	2.30	2.44
45	3.16	4.47	0.00	4.29	4.04	2.69	3.53	2.50	1.67	2.30	2.53
46	2.00	2.00	2.00	2.63	1.93	1.61	1.01	1.50	1.80	1.35	1.41
46	2.00	1.00	1.00	0.94	1.00	0.96	1.13	1.58	1.67	1.77	1.65
46	0.00	1.00	2.24	0.93	1.48	1.04	1.05	1.80	1.00	1.25	1.00
46	1.00	1.00	1.00	0.86	0.97	0.91	1.00	1.58	1.70	1.52	1.02
46	0.00	1.00	2.24	1.14	1.39	1.32	1.30	1.12	1.67	0.75	1.00
47	1.00	1.00	1.00	0.73	0.31	0.35	0.18	0.50	0.00	0.56	0.82
47	1.00	1.41	1.00	0.51	0.54	0.20	0.27	1.12	1.41	2.06	1.84
47	1.41	1.00	1.00	1.08	0.29	0.15	0.23	0.71	0.33	0.50	0.45
47	1.00	1.00	1.00	0.73	0.49	0.35	0.16	1.12	0.75	0.75	0.45
47	1.00	1.00	1.00	0.49	0.35	0.05	0.20	1.12	0.94	0.35	0.20
48	1.41	1.41	1.41	0.60	0.57	0.72	1.05	1.12	1.05	0.56	0.45
48	1.00	1.41	2.24	0.61	0.41	0.59	0.53	1.12	1.20	1.03	0.45
48	0.00	1.00	0.00	0.49	0.44	0.53	0.42	1.58	2.33	1.03	1.34
48	0.00	0.00	1.00	0.75	0.57	0.61	0.56	1.58	1.33	0.25	0.28
48	2.24	1.41	2.24	0.79	0.76	0.81	0.62	4.72	1.05	0.79	2.15
49	1.00	1.41	0.00	0.68	0.91	1.09	0.42	3.91	2.36	1.60	1.34
49	2.24	0.00	0.00	2.16	1.87	2.04	2.21	0.50	0.75	2.02	1.41
49	0.00	0.00	0.00	0.63	1.56	1.88	1.86	4.61	0.67	0.25	3.30
49	0.00	0.00	0.00	2.12	1.80	1.69	1.80	3.20	3.07	2.30	2.33
49	0.00	0.00	0.00	0.57	1.05	1.19	0.79	0.71	0.75	0.25	1.26
50	3.16	1.00	1.00	0.03	0.33	1.11	1.01	1.00	0.47	1.00	1.17
50	1.00	1.00	1.00	0.73	1.82	0.74	1.37	0.71	0.33	0.56	1.79
50	4.00	1.00	1.00	0.07	0.31	1.22	1.03	0.71	0.33	1.25	0.63
50	1.00	1.00	1.00	0.03	0.30	1.12	0.98	0.00	0.33	1.25	0.72
50	1.00	1.00	1.00	0.07	0.38	0.81	1.19	1.58	2.33	1.27	1.41
51	5.00	4.47	5.00	2.18	2.48	3.02	2.94	3.20	3.28	3.91	4.00
51	3.00	3.16	4.12	4.74	3.47	3.78	3.59	3.20	3.07	3.20	3.67
51	5.00	5.00	5.00	3.27	3.08	3.22	3.29	2.69	2.11	1.95	1.84
51	5.00	5.00	5.00	3.20	3.64	3.35	3.74	2.24	2.60	1.41	1.44
51	5.00	5.00	5.00	3.28	3.15	3.31	3.50	2.24	1.37	1.60	2.97
52	2.24	2.24	2.24	1.91	2.41	2.15	2.43	3.20	2.69	3.02	3.16
52	2.24	1.00	2.24	1.97	1.80	2.38	2.42	3.20	2.11	2.93	3.03
52	2.24	2.24	2.24	2.73	3.08	2.01	1.76	4.12	3.90	3.16	2.86
52	2.24	2.24	2.24	1.88	1.77	2.32	1.81	2.69	2.11	1.58	1.00
52	2.24	2.24	2.24	2.92	1.78	2.40	2.94	1.58	1.33	1.82	1.40
53	4.12	5.10	5.10	2.54	3.48	1.63	1.70	1.12	2.75	3.29	2.81
53	4.12	1.00	3.00	2.75	3.04	2.43	3.46	1.00	1.37	2.50	2.41
53	5.00	3.00	3.00	3.15	3.55	2.12	2.90	2.00	1.05	1.25	1.84
53	5.10	5.10	5.10	3.29	1.55	2.48	2.65	2.00	1.80	1.25	1.56
53	4.00	4.12	3.61	3.21	2.81	2.89	1.96	2.55	2.54	3.25	3.23
54	0.00	4.47	4.47	1.04	1.43	1.33	2.26	0.71	0.94	0.56	1.41
54	4.47	1.00	1.41	1.70	2.55	1.96	2.81	0.71	0.94	0.56	0.89
54	1.00	1.00	1.00	0.54	1.90	1.97	1.69	1.12	0.75	1.35	1.22
54	1.00	0.00	1.00	2.70	2.47	2.45	2.51	1.12	0.75	0.25	0.45
54	1.00	1.00	0.00	0.40	1.28	1.82	2.04	0.50	0.75	0.56	0.45
55	2.00	3.16	1.00	1.41	1.18	1.92	2.18	1.41	1.20	1.06	1.28
55	2.00	2.00	6.32	2.36	2.85	5.35	3.69	1.41	1.67	1.46	1.79
55	2.00	2.00	7.81	1.59	1.46	2.11	1.85	2.12	3.14	3.35	3.69
55	2.00	1.00	1.00	1.92	1.94	2.70	2.37	2.55	2.60	2.30	1.98
55	1.00	1.00	1.00	1.52	1.49	2.27	2.39	2.12	1.94	3.40	3.05
56	2.24	2.24	4.00	1.27	1.96	1.22	1.95	4.03	5.47	4.37	5.12
56	2.24	2.24	2.24	0.97	0.66	0.79	1.19	3.04	2.36	2.80	3.23
56	0.00	0.00	0.00	0.48	0.69	0.67	0.86	3.04	2.54	2.06	2.16
56	2.24	0.00	0.00	0.54	0.83	1.12	1.13	2.12	1.41	1.90	2.16
56	1.00	0.00	0.00	0.95	1.74	1.61	1.03	2.12	2.60	2.70	2.16
57	2.00	2.00	2.00	1.10	0.99	1.60	1.25	1.12	1.41	2.02	2.15
57	1.00	1.00	2.00	1.12	1.04	1.27	1.45	1.12	1.41	2.02	1.61
57	1.00	1.00	1.00	0.52	1.00	1.17	1.39	1.58	1.41	2.02	1.61
57	2.00	2.00	1.41	1.11	1.02	1.21	1.11	1.12	1.41	1.68	1.56
57	1.00	2.00	1.41	2.09	2.01	1.57	1.57	2.24	1.41	2.02	1.61

Continued on next page

Table E.2 – continued from previous page

Sample	k3	k4	k5	wc62	wc63	wc64	wc65	c62	c63	c64	c65
58	6.40	6.08	7.81	6.66	6.06	5.66	5.16	3.00	2.00	2.76	2.47
58	7.81	7.81	5.66	6.65	6.32	5.43	5.77	0.50	2.00	2.06	1.61
58	7.81	7.81	5.66	6.73	5.36	4.52	4.50	3.54	2.75	3.29	3.10
58	6.40	7.81	7.81	6.67	6.36	4.54	4.79	4.47	3.43	3.35	3.44
58	6.08	7.81	7.81	4.72	4.77	5.74	5.58	3.81	3.80	3.69	3.22
59	0.00	3.16	0.00	3.89	2.72	3.14	3.55	3.20	2.98	2.66	2.83
59	1.41	1.41	0.00	1.46	2.69	3.76	2.90	1.58	1.70	2.47	3.26
59	2.83	2.83	2.83	2.94	3.39	1.30	1.80	0.71	4.68	3.58	2.83
59	2.83	0.00	2.83	3.22	2.90	2.87	3.07	3.81	2.11	2.30	2.83
59	0.00	2.83	0.00	2.86	2.83	2.33	2.43	2.83	2.69	2.46	1.84
60	2.24	1.00	2.24	3.82	2.35	2.72	2.27	1.00	0.94	0.79	1.08
60	5.83	3.16	1.00	2.83	1.80	2.12	2.36	0.71	0.75	0.79	1.60
60	5.00	2.24	2.24	2.72	3.30	3.09	3.96	1.50	0.75	0.79	1.60
60	5.00	1.00	2.24	2.85	4.19	3.98	2.97	1.50	0.33	2.24	2.78
60	2.24	2.24	1.00	2.09	2.00	2.83	2.51	1.50	0.75	0.75	1.90
61	2.83	2.83	2.00	1.78	1.82	2.17	2.60	5.00	2.60	4.07	2.42
61	2.24	2.83	2.83	2.50	3.05	3.01	2.64	2.50	1.20	1.35	2.83
61	2.83	2.00	2.83	3.44	1.83	1.85	1.35	2.24	2.13	1.35	4.88
61	2.83	1.00	1.00	1.80	1.92	1.25	1.39	2.24	1.80	3.40	3.33
61	2.83	2.00	2.00	1.11	1.96	1.26	1.47	3.00	4.47	3.81	3.10
62	2.00	2.24	2.24	2.00	2.37	2.82	3.10	4.03	2.33	2.25	1.81
62	5.10	5.10	6.08	4.98	3.78	4.20	3.93	1.00	0.75	2.26	2.40
62	5.10	6.08	6.08	4.44	3.78	3.44	3.03	1.58	1.05	1.50	1.26
62	2.00	2.00	2.00	3.17	3.10	3.53	2.99	2.06	1.37	2.51	2.40
62	2.24	2.24	2.24	3.29	3.02	2.63	3.37	3.81	0.75	0.90	1.56
63	4.12	2.00	1.00	1.44	0.76	0.94	1.34	0.50	1.70	2.61	2.06
63	4.00	1.00	1.00	1.50	1.17	1.77	1.31	1.80	1.49	2.02	2.06
63	3.00	1.00	1.00	1.27	0.86	0.74	1.04	2.06	1.70	1.68	1.52
63	3.00	0.00	1.00	1.37	0.95	0.84	0.78	1.80	2.54	2.02	2.06
63	1.00	3.00	1.00	0.56	0.78	1.65	1.93	2.06	2.54	2.46	2.06

Table E.3: Orientation Test of RSSI for TelosB mote

	Angle															
Sample No.	**0**	**22.5**	**45**	**67.5**	**90**	**112.5**	**135**	**157.5**	**180**	**202.5**	**225**	**247.5**	**270**	**292.5**	**315**	**337.5**
1	-69	-70	-72	-69	-75	-89	-86	-80	-85	-83	-79	-83	-81	-88	-80	-79
2	-70	-69	-71	-70	-75	-87	-86	-79	-82	-78	-77	-79	-80	-86	-87	-81
3	-69	-69	-70	-71	-75	-88	-80	-77	-83	-81	-77	-83	-83	-82	-84	-79
4	-71	-71	-71	-74	-77	-84	-82	-79	-86	-82	-80	-81	-88	-79	-88	-78
5	-71	-70	-71	-76	-76	-85	-84	-79	-85	-84	-84	-84	-84	-79	-83	-81
6	-71	-69	-73	-71	-76	-88	-85	-79	-80	-82	-85	-87	-86	-77	-77	-77
7	-71	-70	-71	-71	-75	-86	-85	-78	-74	-86	-82	-86	-87	-77	-76	-79
8	-70	-67	-71	-71	-75	-84	-86	-78	-79	-81	-77	-84	-90	-79	-74	-77
9	-69	-69	-71	-71	-86	-79	-88	-79	-80	-81	-77	-78	-87	-80	-77	-77
10	-67	-70	-70	-72	-86	-85	-81	-79	-78	-80	-79	-78	-89	-78	-78	-71
11	-68	-71	-70	-73	-83	-86	-83	-80	-80	-82	-79	-79	-84	-78	-77	-70
12	-69	-70	-69	-74	-86	-86	-82	-77	-78	-86	-80	-80	-85	-77	-78	-71
13	-69	-70	-70	-74	-87	-83	-84	-79	-81	-88	-84	-83	-82	-77	-80	-71
14	-69	-70	-70	-76	-87	-80	-80	-83	-84	-83	-84	-88	-81	-77	-79	-71
15	-70	-70	-71	-77	-86	-78	-81	-85	-83	-79	-84	-88	-84	-77	-84	-71
16	-69	-71	-71	-75	-85	-78	-80	-84	-80	-78	-84	-83	-81	-77	-80	-71
17	-69	-74	-70	-75	-87	-79	-82	-83	-82	-79	-85	-81	-79	-78	-76	-72
18	-69	-73	-70	-73	-87	-80	-79	-86	-85	-82	-82	-79	-80	-80	-73	-72
19	-70	-71	-71	-73	-86	-83	-80	-88	-81	-77	-80	-79	-78	-79	-75	-72
20	-71	-71	-73	-72	-84	-87	-77	-86	-81	-75	-79	-81	-79	-76	-75	-71
21	-69	-71	-73	-72	-80	-84	-79	-86	-80	-75	-80	-77	-81	-77	-77	-72
22	-68	-71	-72	-73	-78	-84	-79	-82	-81	-76	-83	-80	-86	-79	-78	-72
23	-69	-72	-73	-74	-80	-83	-77	-83	-80	-77	-87	-85	-83	-79	-77	-74
24	-70	-73	-75	-71	-78	-82	-75	-82	-77	-79	-82	-88	-78	-78	-75	-73
25	-70	-71	-75	-73	-77	-86	-73	-85	-80	-84	-81	-87	-80	-75	-75	-71
26	-70	-72	-72	-73	-78	-84	-75	-85	-86	-87	-82	-85	-80	-75	-77	-71
27	-71	-73	-73	-74	-79	-86	-75	-86	-83	-78	-79	-81	-80	-75	-75	-71
28	-72	-72	-73	-75	-79	-85	-74	-86	-81	-79	-78	-81	-80	-76	-75	-71
29	-73	-70	-74	-74	-80	-84	-74	-84	-82	-77	-80	-81	-79	-75	-76	-71
30	-73	-71	-75	-75	-82	-81	-75	-82	-84	-84	-83	-83	-79	-75	-75	-71
31	-72	-73	-75	-74	-82	-79	-77	-84	-84	-86	-84	-83	-81	-74	-75	-71
32	-71	-72	-73	-74	-80	-78	-80	-77	-81	-84	-83	-82	-80	-75	-73	-71
33	-72	-71	-73	-73	-82	-79	-85	-79	-82	-86	-84	-82	-78	-75	-73	-70
34	-71	-71	-71	-73	-80	-80	-78	-82	-84	-88	-84	-81	-76	-75	-75	-70
35	-71	-71	-71	-73	-82	-79	-77	-75	-81	-83	-84	-79	-76	-76	-74	-69
36	-71	-72	-71	-75	-84	-80	-82	-78	-79	-77	-84	-77	-79	-77	-72	-69
37	-71	-73	-71	-74	-83	-82	-79	-82	-84	-83	-86	-74	-82	-79	-73	-70
38	-70	-74	-71	-73	-86	-78	-79	-76	-86	-85	-82	-74	-84	-77	-75	-70
39	-68	-73	-71	-75	-86	-77	-78	-77	-83	-88	-80	-76	-85	-78	-74	-71
40	-69	-73	-71	-77	-84	-79	-79	-75	-86	-84	-79	-79	-84	-79	-75	-73
41	-70	-73	-72	-81	-81	-80	-79	-76	-84	-80	-88	-82	-80	-78	-75	-72
42	-70	-74	-73	-80	-82	-79	-80	-77	-84	-81	-79	-86	-79	-79	-77	-71
43	-69	-74	-73	-81	-79	-80	-82	-77	-82	-86	-78	-87	-77	-80	-76	-70
44	-68	-73	-73	-75	-78	-78	-84	-77	-83	-84	-78	-87	-77	-82	-75	-69
45	-69	-72	-73	-75	-84	-80	-84	-77	-83	-80	-79	-87	-78	-81	-75	-67
46	-69	-73	-71	-78	-85	-82	-84	-79	-84	-84	-80	-86	-78	-83	-77	-67
47	-69	-73	-71	-77	-82	-88	-86	-80	-87	-83	-77	-82	-79	-86	-76	-69
48	-70	-74	-71	-76	-79	-87	-83	-84	-86	-82	-76	-81	-78	-86	-77	-69
49	-69	-75	-71	-77	-83	-83	-82	-82	-85	-84	-75	-78	-78	-88	-77	-69
50	-69	-76	-71	-82	-83	-79	-78	-83	-88	-81	-78	-79	-81	-85	-75	-71
51	-69	-77	-72	-79	-82	-78	-77	-82	-85	-77	-77	-85	-87	-84	-76	-71
52	-67	-77	-71	-76	-80	-79	-77	-88	-86	-76	-77	-83	-84	-82	-74	-72
53	-69	-75	-72	-73	-81	-83	-79	-83	-87	-79	-76	-80	-85	-75	-72	-71
54	-70	-76	-73	-72	-79	-84	-80	-81	-87	-87	-81	-79	-88	-75	-74	-71
55	-69	-75	-75	-73	-80	-84	-79	-85	-89	-79	-84	-82	-82	-75	-75	-71
56	-69	-75	-75	-73	-80	-84	-80	-86	-88	-77	-85	-85	-77	-75	-75	-69
57	-71	-77	-72	-74	-80	-83	-78	-85	-84	-87	-88	-85	-77	-76	-76	-68
58	-69	-77	-72	-75	-82	-84	-78	-87	-84	-88	-79	-83	-80	-77	-76	-70
59	-69	-73	-71	-73	-84	-81	-78	-80	-87	-85	-81	-83	-79	-77	-77	-71
60	-70	-71	-71	-75	-85	-84	-77	-79	-88	-82	-84	-85	-83	-80	-78	-71

Continued on next page

Table E.3 – continued from previous page

	Angle															
Sample No.	**0**	**22.5**	**45**	**67.5**	**90**	**112.5**	**135**	**157.5**	**180**	**202.5**	**225**	**247.5**	**270**	**292.5**	**315**	**337.5**
61	-71	-71	-71	-77	-81	-80	-79	-81	-87	-87	-86	-86	-83	-82	-81	-70
62	-71	-71	-71	-75	-79	-79	-77	-83	-80	-79	-88	-86	-79	-79	-78	-70
63	-70	-73	-70	-75	-81	-79	-75	-84	-77	-83	-84	-86	-79	-79	-79	-70
64	-70	-73	-70	-75	-83	-80	-80	-82	-79	-81	-87	-86	-79	-81	-82	-70
65	-71	-72	-71	-75	-83	-79	-79	-79	-82	-85	-83	-81	-81	-82	-79	-70
66	-71	-73	-73	-74	-83	-83	-80	-86	-82	-85	-79	-79	-83	-79	-79	-69
67	-71	-73	-72	-74	-82	-84	-77	-83	-87	-82	-77	-80	-84	-79	-77	-70
68	-70	-71	-72	-74	-82	-82	-79	-89	-80	-76	-79	-83	-87	-80	-75	-70
69	-71	-71	-71	-75	-87	-83	-77	-86	-77	-79	-80	-79	-87	-84	-75	-71
70	-70	-71	-69	-74	-84	-82	-79	-86	-76	-85	-80	-82	-82	-84	-74	-71
71	-69	-70	-69	-73	-86	-81	-82	-86	-75	-89	-79	-80	-77	-85	-73	-70
72	-69	-69	-71	-72	-83	-84	-81	-86	-77	-83	-82	-78	-84	-86	-73	-69
73	-68	-70	-71	-72	-83	-83	-84	-88	-81	-80	-79	-78	-85	-86	-77	-70
74	-69	-71	-71	-72	-83	-85	-81	-83	-82	-79	-83	-81	-84	-85	-78	-70
75	-68	-72	-71	-73	-86	-84	-80	-85	-80	-87	-81	-80	-85	-88	-78	-72
76	-66	-72	-71	-73	-85	-85	-83	-83	-87	-80	-82	-84	-86	-84	-78	-71
77	-66	-72	-70	-73	-86	-84	-84	-86	-84	-85	-85	-79	-84	-79	-80	-71
78	-67	-74	-71	-73	-87	-83	-85	-84	-82	-85	-83	-78	-84	-79	-81	-71
79	-68	-74	-71	-73	-88	-86	-88	-88	-78	-84	-84	-83	-77	-80	-82	-72
80	-69	-75	-70	-73	-85	-86	-87	-86	-77	-77	-88	-84	-79	-83	-80	-71
AVERAGE		**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**
	69.6	**72.2**	**71.6**	**74.2**	**82.0**	**82.5**	**80.3**	**82.1**	**82.4**	**82.0**	**81.4**	**82.0**	**81.7**	**79.5**	**76.9**	**71.5**