Improving Reliability In Body Area Sensor Network Communications For eHealth Monitoring

by

Jacqueline Callanan, B.Sc.

Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the Degree of

Master of Science in Computer Science

(Mobile and Ubiquitous Computing)

School of Computer Science and Statistics

University of Dublin, Trinity College

August 2013

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Jacqueline Callanan

30th August, 2013

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Jacqueline Callanan

30th August, 2013

Acknowledgements

"*Out of life's school of war: What does not destroy me, makes me stronger.*"

— Twilight of the Idols by Friedrich Nietzsche (1844-1900)

First and foremost, I would like to thank my supervisor, Jonathan Dukes, not only for suggesting an interesting topic for my project originally but also for being such a reliable source of guidance and support. I am indebted to him for his encouragement, patience and positive attitude. I am also very grateful to Stefan Weber for being my second reader and to Tom Kearney for superb hardware modifications, not to mention an emergency battery repair.

I would like to thank Ciarán McGoldrick for his helpful advice when I started this course and Meriel Huggard for her assistance and inspiration. Special thanks are also due to some friends who helped me out in my times of need this year: Seán Duddy, Declan O'Sullivan, Mary Tangney and Mark Jones. I would also like to thank my brother John for proof-reading this document for me.

And last but certainly not least, I thank my husband Richard Varney for being my rock and for holding everything together for me under difficult circumstances. It has been a tough year on family life and I really appreciate the numerous sacrifices that he and our daughters, Lucy and Edel, had to make so that I could study.

I would like to dedicate this dissertation to the memory of my parents Tom and Deirdre Callanan who made this year possible for me.

JACQUELINE CALLANAN

University of Dublin, Trinity College,
August 2013

Improving Reliability In Body Area Sensor Network Communications For eHealth Monitoring

Jacqueline Callanan, M.Sc.
University of Dublin, Trinity College, 2013

Supervisor: Dr. Jonathan Dukes

Abstract

We are living in an era where advances in sensor technologies, mobile devices and wireless communications are facilitating what some are calling a healthcare revolution. Self-monitoring is already commonplace with inexpensive sensor devices readily accessible to consumers. In addition, there are numerous applications available for our smartphones and tablets that allow us to manage our calorie intake, track how much exercise we have taken and tell us how well we have slept. A phone's camera can even be used to measure how fast someone's heart is beating. Some doctors are beginning to prescribe applications for self-monitoring instead of, or in conjunction with, medication.

eHealth is prevalent in the sports and fitness industry, with some professional teams employing physical performance experts who can monitor each team member's physical activities in real-time through the use of mobile wireless devices such as GPS trackers, heart rate monitors and accelerometers. Injuries and illnesses may be detected early and even prevented in some cases.

Increasing healthcare costs, an aging population and a shortage of medical staff are some drivers for eHealth and telemedicine in remote monitoring scenarios. In the case of assisted living for the elderly, given the choice, many would prefer to remain living in the comfort of their own smart homes instead of being hospitalised. However, in order for remote monitoring to be safe, it is imperative that critical situations are detected promptly and urgent alerts are forwarded to a Remote Monitoring System (RMS) as quickly as possible, so that the appropriate action can be instigated. A lost or significantly delayed alert could potentially mean the difference between life and death. Reliability is therefore of paramount importance and this project focuses on improving the reliability of communications between a Body Area Sensor Network (BASN) and a RMS.

A BASN is a collection of sensors that are worn on, and sometimes implanted within, the human body. These devices can share information by communicating wirelessly with each other and with other computing devices within range. Sewable electronic components, conductive thread and eTextiles all allow nodes in a BASN to be worn as part of someone's clothing. Microcontrollers provide the computing capabilities and lightweight batteries supply the power. The network is mobile, it moves with the wearer, and should have minimal intrusion on the user's activities of daily living.

This dissertation proposes a design for a remotely monitored, reliable and autonomous BASN to support independent living for the elderly. The emphasis of the design is to facilitate the development of a robust and reliable system that remains operational even when communication failures occur. A communications configuration and protocol are proposed to support these objectives. A prototype is designed and implemented that uses a BASN of movement, temperature and heart rate sensors as well as a panic button. Experiments are designed with test suites to evaluate the prototype. The goal of the project is to improve reliability in communications between a BASN and a RMS, with the foremost objective of alerts not being lost when communications errors occur. In order to reduce the risk of an external communications outage, a secondary communications link is required as backup so that alerts generated by a BASN can still reach a RMS if the primary communications link is not working. The proposed communications protocol is designed to work with an event-based architecture and a generic set of sensors. In the event of a total communications outage between the BASN and the RMS, the protocol ensures all outstanding delayed alerts are forwarded to the RMS as soon as connectivity is re-established.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

eHealth is the use of information and communication technologies (ICT) for health. Examples include treating patients, conducting research, educating the health workforce, tracking diseases and monitoring public health[1]. The importance of eHealth can be seen in the fact that it is a major area of project work for the World Health Organisation (WHO). In support of this, WHO has established a designated Collaborating Centre in consumer health informatics. eHealth was the subject of a special issue of the public health journal The Bulletin [1] in which remote patient monitoring was noted as being an improvement to health systems' performance. Other benefits of eHealth for patient care included improved access to health advice, access to remote consultations and telemedicine and quicker access to emergency services. It was also noted that eHealth technologies facilitate radically new ways of delivering and monitoring care.

WHO publish a number of interesting facts about ageing including the following[2]:

> The world population is rapidly ageing.
> Between 2000 and 2050, the proportion of the world's population over 60 years will double from about 11% to 22%. The absolute number of people aged 60 years and over is expected to increase from 605 million to 2 billion over the same period.

As the global population ages, it is to be expected that new and innovative services will become more commonly available for the elderly, as eHealth technologies develop to support pervasive and ubiquitous health monitoring. Remote monitoring is one such service that can help patients to remain at home while receiving healthcare.

[1] http://www.who.int/topics/ehealth/en/ viewed on 28/08/2013

[2] http://www.who.int/ageing/about/facts/en/index.html viewed on 27/08/2013

Another eHealth article [2] in The Bulletin includes results from a trial of 5,715 patients who had suffered heart failure. The cost per patient of treatment including remote monitoring was €300 - €1,000 cheaper than the cost of conventional treatment. If this cost-saving becomes typical of healthcare treatments delivered through remote monitoring then it can be expected to be a further incentive to governments and healthcare providers to invest in eHealth technology and remote monitoring services.

Advances in technology in recent years in the areas of wireless communication, biometric sensors, mobile devices and batteries are helping eHealth and mHealth to become a reality. Services supporting panic buttons, remote monitoring for security and health purposes, implantable and wearable biosensors and smartphones are already pervasive in society for many age groups.

One definition[3] of the word reliable is:

> "consistently good in quality or performance; able to be trusted"

In order for any system to be worthy of trust where there is the possibility of human life being at risk, it is essential that the system is reliable and safe. For remote monitoring to become ubiquitous, one area where it must be reliable is in terms of generating alerts, whenever emergency services may be required. Not detecting a critical health-related event or not alerting the relevant authorities on time could mean the difference between life and death. Therefore it is critical that serious health-related events do not go undetected and it is highly desirable that the system does not generate false alarms.

Distributed systems are notoriously complex [3]. There is always the possibility of some unexpected sequence of events or conditions occurring that can cause a failure. In this regard, it is unlikely that any system is 100% immune to all possible combinations of failures. Therefore every effort must be made to avoid single points of failure (one smart phone cannot be responsible for all external communications for example) and to use redundancy techniques to improve reliability so as to minimise the impact of any individual failure on the system as a whole.

[3] Retrieved from http://oxforddictionaries.com/definition/english/reliable?q=reliability#reliable__7 on 25/08/2013

1.1 Research Area

The area of research covered in this dissertation concerns the reliability of communications between a Body Area Sensor Network (BASN) and a Remote Monitoring System (RMS). Of specific interest is how to make BASN communications more reliable for eHealth Monitoring. In the case of remote health monitoring, it is essential to ensure that any critical alerts that are generated by a BASN reach a Remote Monitoring System and that no alerts are lost, even if communication links fail. Redundant communication channels are required to improve reliability.

1.2 Scope

The general context for the project was clear from the beginning, the remit was associated with the design of a fault-tolerant Remote Monitoring System to assist independent living for the elderly. The precise scope of the project is:

> To improve the reliability in communications between a Body Area Sensor Network and a Remote Monitoring System for the purposes of eHealth, in order to assist the elderly to live independently.

1.3 Objectives

The main research goal of this project is to identify a system design or technical architecture for a wireless sensor network that supports the development of a fault-tolerant, anti-fragility monitored system that is both safe and reliable and suitable for use in an elderly person's home. The remote monitoring healthcare system to support independent living for the elderly can be seen as a proof-of-concept or case study for the project.

The specific aims of the project are:

1) To research the State of the Art in areas associated with Body Area Sensor Networks and eHealth in the context of remote monitoring to promote assisted living for the elderly.
2) To investigate a suitable communications topology and technology for a Body Area Sensor Network (BASN).
3) To propose a communications configuration to support the reliable communication of alerts, that are generated by a BASN, to a Remote Monitoring System (RMS).

4) To propose a communications protocol to help improve the reliable communication of alerts from a BASN to a RMS.
5) To implement a prototype that uses the proposed protocol to communicate alerts from a BASN to a dummy RMS. This includes:
    - Selecting and configuring hardware components such as microcontrollers and sensors as well as communications equipment for a BASN.
    - Designing a system architecture and implementing software components to run on the selected equipment.
6) To specify and execute experiments to evaluate the prototype.
7) To identify any issues or areas for future work.

1.4 Exclusions

Although the project concerns the communication of alerts between a Body Area Sensor Network and a Remote Monitoring System, the main focus is on the communication protocol between the BASN nodes and local gateway nodes. These gateway nodes act as intermediaries between the BASN and the RMS and are discussed in detail in **Section 3.4**. The gateways have external communication capabilities, using broadband, 3G or other wireless links.

Details of the precise physical communication links used between the gateway nodes and the RMS are not covered in this dissertation. However, communications between the gateway nodes and the RMS should be secure and reliable and would use a fully acknowledged transport protocol such as TCP (Transmission Control Protocol). The protocol proposed in **Chapter 3** to improve the reliability of communication between the BASN and the RMS requires application-layer acknowledgements to be returned by the RMS upon receipt of an alert from the BASN.

Furthermore, the detailed design of a Remote Monitoring System is beyond the scope of this project. Although the RMS would be expected to be able to filter out duplicate alerts, process alerts from multiple BASNs simultaneously, return acknowledgements promptly, provide secure and reliable communications options, have high availability and be able to invoke the appropriate action to deal with alerts in real-time – particularly if emergency services had to be called.

1.5 Dissertation Structure

This dissertation document is organised into a number of chapters, appendices and a bibliography. The full Table of Contents is given on pages vii – x. The following table gives a high level overview of the whole document for clarification purposes.

Chapter		Contents
1	**Introduction**	Overview of the project, its scope and objectives.
2	**State of the Art**	Discussions and summaries of research areas that are closely related to the project e.g. Wireless Personal Area Network communications, ZigBee failure issues, Middleware.
3	**Design**	This chapter summarises the requirements for the prototype and proposes a communications configuration and protocol to improve the reliability of the communication of alerts between a Body Area Sensor Network and a Remote Monitoring System.
4	**Implementation**	Full details of the implementation of a prototype adhering to the design recommendations and the proposed protocol in Chapter 3.
5	**Evaluation**	Specification of experiments to be run to prove the prototype meets its requirements. Due to technical issues that will be explained, the experiments have not yet been carried out.
6	**Conclusions**	Summary of achievements and suggestions for future work.
Appendix		**Contents**
A	**Abbreviations and Definitions**	There are many acronyms and abbreviations used throughout the document. This appendix attempts to define and/or briefly explain terms which some readers may find helpful.
B	**Bibliography**	There are a number of citations throughout the document. These appear as numbers within brackets such as [1] etc. The bibliography lists full details of all the documents that have been cited. Some documents were obtained from web sites in which case the retrieval date is also included.

Table 1 Dissertation Structure

Chapter 2 State of the Art

The elderly can be supported to live independently in their own home environment with the assistance of an autonomous monitoring system which uses a body sensor network to collect physiological data and then automatically decides whether or not to generate alerts. These alerts could be sent to a central monitoring station or to the emergency services, depending on the nature of the issue that has been detected.

A number of healthcare monitoring systems have already been implemented to various degrees, and are being prototyped, to support independent living for the elderly in smart home and assisted living environments. One examples is an Australian remote health monitoring service provider called Patient Connect[4]. Another example is an American remote patient monitoring service provided by Guardian eHealth Solution[5]. All such systems share a common issue regarding reliability and the generation of false alarms [4].

There are also issues regarding reliable fall detection mechanisms [5] and this specific event is prone to false alarms [6]. This particular area is heavily researched and is not covered in this dissertation.

The following table gives an overview of the topics that are reviewed in this chapter.

Section	Topic
2.1	Remote Monitoring eHealth Systems
2.2	Multiple Sensor Platforms
2.3	Wireless Body Area Network communications standards and technologies

[4] http://www.patientconnect.com.au/ viewed on 28/08/2013

[5] http://gehs.net/remote-patient-monitoring/ viewed on 28/08/2013

Section	Topic
2.4	Challenges in Body Area Networks
2.5	Reliability issues relating to ZigBee networks
2.6	Middleware
2.7	Summary of Reliability terminology
2.8	Chapter summary highlighting where this dissertation's research area fits in to current research.

Table 2 Summary of Sections in Chapter 2

2.1 Remote Monitoring eHealth Systems

There are many research papers covering remote monitoring systems. This section summarises points from a survey journal article [7].

A number of eHealth Body Area Network (BAN) applications and prototypes already exist. A survey of such solutions suggest a strong potential for the improved quality of medical healthcare provided remotely through the ubiquitous deployment of wireless sensor networks.

Common design considerations include unobtrusiveness, scalability, energy-efficiency and security. Real-time availability and reliability of communications is a major concern.

Many of the existing commercial and prototype BANs connect to backbone networks via gateways allowing healthcare professionals to monitor patients remotely in real-time. These gateways often provide the only interface between a BAN and a remote monitoring network and therefore can be the weakest link in the communications chain.

Reliability issues are categorised as follows:

- Reliable data measurement
- Reliable data communications. This is particularly difficult to ensure when using BANs as low transmission power requirements can increase error rates.
- Reliable data analysis

2.2 Multiple Sensor Platforms

This section discusses an example of recent research [8] that used a Minimally Invasive Monitoring Sensor (MIMS) platform to continually monitor the human movements and physiological signals of older adults with cognitive difficulties. The MIMS platform can be used to build a comprehensive and customisable real-time health monitoring system.

This research notes that passive monitoring systems that use cameras and on/off sensors on doors, beds, chairs etc. are useful only in detecting events after they have happened. These systems can violate the user's privacy, collect a large amount of redundant data and often miss incidents - if the user falls where there is no floor sensor for example.

MIMS is used to analyse physiological data preceding potential emergency events in order to predict them quickly and to help prevent emergency incidents. MIMS consists of wearable, active sensor devices such as accelerometers, gyroscopes and biosensors. Each sensor in a wearable device can pre-process and filter redundant data so that only useful information is provided. This also reduces traffic and extends sensor battery life. There is an optional graphical user interface to allow users to interact with the sensor devices and perform self-diagnosis and check-up.

A virtual hub that can reside on a desktop computer, laptop, smart phone, PDA or distributed system, acts as a gateway between the monitored user and the caregivers. Passive sensing devices can also connect to the hub so that correlation analysis can be carried out on the active and passive data. When used in a home monitoring scenario, the virtual hub connects the house with the external world. The hub has a Caregivers Interface to connect to health care application systems that are used by hospitals and caregivers. This is a two-way communications channel allowing caregivers to receive alarms from MIMS and also allowing caregivers to remotely monitor the user and perform online queries or check heart-rate or blood pressure etc.

Older people are more likely to fall at home than in nursing homes for example so MIMS can be used to provide a fall prevention and detection system. A sensor algorithm was developed to detect falls. False positives were reduced by simultaneously detecting and analysing activities of daily living (ADL) e.g. lying down or sitting quickly on a chair.

In order to help prevent night-time fall incidents, a smart-hat solution was used to monitor brain activity in conjunction with sleep monitors. Signal processing algorithms were used to extract pre-fall signal data.

2.3 Wireless Body Area Networks

Standards for short distance wireless networks, also called Wireless Personal Area Networks (WPANs), are defined by the **IEEE 802.15** Working Group. This includes the 802.15.1/ Bluetooth and 802.15.4/ZigBee protocols. These ad hoc networks operate over short ranges at low power, with ZigBee being more suited to lower-powered and lower-data-rate applications than Bluetooth.

2.3.1 IEEE 802.15.1 and Bluetooth

The link and physical layers of the **IEEE 802.15.1** standard [9] are based on an older Bluetooth specification for Personal Area Networks. Bluetooth was originally designed as a direct replacement for asynchronous TTL serial communication. There are a number of Bluetooth profiles supported e.g. the **Serial Port Profile** (SPP) is suitable for serial cable replacement links.

802.15.1 networks operate in the 2.4GHz unlicensed radio band. Time Division Multiplexing (TDM) is used to allow a sender to transmit on one of 79 channels. Data rates up to 4 Mbps are supported.

802.15.1 networks require no network infrastructure and devices organise themselves into a master-slave configuration. The supported topology is called a Piconet which is the same as a star (see **Section 3.2.1**). There is one master device with a number of active and parked slave devices. The master device is a single point of failure as slaves can only communicate with the master node.

2.3.2 IEEE 802.15.4

The **IEEE 802.15.4** standard [10] defines the Physical (PHY) and Medium Access Control (MAC) layers of the communications protocol stack. These layers provide the physical data transport. Standards for the ZigBee network and application layers are specified separately by the ZigBee Alliance (see **Section 2.3.3**) and run above 802.15.4 layers. The 802.15.4 PHY layer supports radios operating at 2.4 GHz or 868/915 MHz.

2.3.3 ZigBee

ZigBee is a low-power standard for personal wireless networks. The ZigBee Alliance define ZigBee application and network layers [11] that run above the PHY and MAC layers defined by

802.15.4. ZigBee supports a number of different topologies, including meshes (see **Section 3.2.4**). A ZigBee network is self-healing as it uses AODV (Ad hoc on-demand vector) routing – if a node on an established route has failed then a new route can be found through node discovery. Routing around failed nodes is a very important reliability feature. ZigBee defines channel rates of 20, 40, 100 and 250 Kbps – all slower than Bluetooth.

ZigBee is a robust protocol (packets are acknowledged). However every ZigBee network requires exactly one coordinator to start the network. The coordinator may act as a router once the network is formed. There can be any number of routers and end nodes in a ZigBee network. End nodes are low power as they can sleep.

The ZigBee Alliance recommends ZigBee as an Assistive Technology (AT) for medical devices as it is easy to use, allows users to maintain their independence and mobility and is optimised for remote monitoring [12]. For example, ZigBee Health Care was designed for use by assistive devices operating in non-invasive health care and is a standard for data exchange.

The main reliability issues with ZigBee networks is the coordinator as it can be a single point of failure. This is discussed further in **Section 2.5**.

2.3.4 IEEE 802.15.6

IEEE 802.15.6 is the standard for Body Area Network (BAN) technologies [13]. The focus is on ultra-low power and short-range wireless devices that operate on, in, or around the human body to serve a variety of applications including health care. This standard:

- allows devices to operate on very low transmit power for safety reasons so as to minimise the Specific Absorption Rate (SAR) into the body and increase battery life;
- supports quality of service (QoS), for example, to provide for emergency messaging;
- provides for robust security, since some communications can carry sensitive information.

A Body Area Network (BAN) consists of a number of sensors that are placed on or inside the human body that communicate their readings back to a nearby hub device, such as a smart phone. There are a number of communications challenges with BANs, particularly at the Medium Access Control (MAC) level. Reliability and energy efficiency are key considerations as some of the sensors may be monitoring vital life signs and sensors cannot operate without battery power.

2.4 Challenges in Body Area Networks

There are a number of common challenges when monitoring the health of elderly persons living independently e.g. size and weight of hardware components, cost, compatibility and perceived value associated with body area sensor networks [4]. Further research challenges include scalability (in terms of data rate, power consumption and duty cycle), antenna design, interference mitigation, coexistence, QoS, reliability, security, privacy and energy efficiency [14].

The following is a list of the four key challenges faced by BANs [15]:

1) **Energy Efficiency**
   BAN sensor nodes must be small and have small batteries that last for days or years, depending on the application. Therefore, BAN nodes must be extremely frugal in their energy usage. There is a trade-off between energy efficiency and the rate at which data is sampled.

2) **Reliable Wireless Communications**
   The wireless signals may be severely attenuated by the human body as they are transmitted from sensors to the hub, particularly when the user is mobile.

3) **Interference**
   If multiple people wearing BANs come into range of each other it can be difficult or even impossible to coordinate the nodes in each BAN as separate networks.

4) **Performance requirements**
   BANs used for medical applications need to support a wide range of throughput rates (1 Kbps to 10 Mbps) while still delivering high reliability and low-latency.

2.5 Reliability in ZigBee Networks

As mentioned in **Section 2.3.3**, there must be exactly one coordinator per ZigBee network. The coordinator is needed to initialise the network at start up time. ZigBee network can function in certain circumstances without the coordinator, specifically if it is not required for routing purposes.

From a reliability point of view, the ZigBee coordinator can be a major vulnerability. Therefore the ZigBee network topology is particularly important from a reliability standpoint. Using ZigBee

routers and a full mesh topology (see **Section 3.2.4)** can considerably enhance the reliability of a ZigBee network.

One idea to help improve reliability is to keep a backup coordinator on standby on a separate PAN and to force surviving nodes without a coordinator (on a different PAN) to modify their PAN Identifier and to force them to migrate to the standby's PAN. This is suggestion for future work in **Section 6.2**.

2.5.1 Fault Tolerance in ZigBee Wireless Sensor Networks

One paper by a NASA research centre [16] considers fault tolerance in ZigBee mesh networks for space projects when there is RF interference or a Wireless Sensor Network (WSN) node failure. The research goal was to create a single fault-tolerant WSN for aerospace applications.

A number of experiments were carried out by inducing faults in router and sensor nodes at the Network layer. Some faults were induced by causing RF interference at the PHY layer. Nodes were cut off from the network and timings were taken to see how quickly alternative routes were discovered.

WSN reliability concerns included in-band RF interference, multipath distortion and node failures. Interference was generated through the use of a Wireless Local Area Network (WLAN) access point that operated on the same frequency as the WSN. Multipath distortion was caused by running the WSN in a closed metal environment. A number of different reliability metrics were measured including throughput, recovery time, packet loss and Received Signal Strength (RSS). Measurements for PAN construction and reconstruction times were also taken for various scenarios using mesh networks with different numbers of hops. Test results showed that the ZigBee 2007 protocol [11] uses robust failover mechanisms.

An interesting fault tolerant WSN architecture was proposed with two coordinator nodes in a balanced mesh using redundancy techniques for fault tolerance. There is still the potential for some data loss as there could be a short loss of streaming data during reconfiguration.

However the exact method of setting up the parallel coordinators was not fully identified and is an outstanding design issue. This area is worth of further research and is included in **Section 6.2**.

2.6 Middleware

Some reasons for having a Middleware software layer, that logically sits in-between a higher level eHealth application layer and a BAN's sensor and gateway nodes, are to:

- Provide a standard interface to eHealth applications that wish to use a variety of heterogeneous bio and ambient sensors.
- Shield applications from details of the underlying network topology and communication protocols.
- Provide common services such as data security, a real time clock, event/message logging, event escalation/alarm generation etc.
- Simplify configuration changes like the addition and removal of sensors and/or eHealth apps.
- Facilitate the implementation of an intelligent algorithm across nodes in the BAN to promote operational reliability e.g. nodes could be grouped to help detect and potentially resolve operational issues such as comms failures, depleted power/battery resources, missed events, critical alerts which have not generated alarms etc.

The following subsections review some research papers concerning Middleware and sensor networks.

2.6.1 Integrating Sensors with the Cloud

A paper [17] that researches a generic architecture to integrate any Wireless Sensor Networks (WSN) with Cloud Computing (CC) uses a lightweight component model. WSNs are resource-constrained and yet have high demands for real-time data transmission and processing. These requirements can be met by CC services which can provide processing and storage on demand. However there are two issues when using the Cloud as a backend for WSNs: latency and the ability of the Cloud to support periodic events.

The component model is used with dynamic proxies to connect any lightweight WSNs (including those used for medical monitoring) to the Cloud. It is noted that WSNs can gather a large amount of irrelevant information and that raw data needs to be filtered, aggregated and processed. Cloud services that offer Infrastructure as a Service (IaaS) are considered as these provide low-layer processing and data storage which are the computational resourced needed by WSNs. Hybrid clouds that are partially public and partially private are best for WSNs as these provide both

security and resource potential. Experiments were also run to show that CC services have sufficient elasticity to process the collected data from periodic WSN events.

Six different approaches were considered for integrating WSNs with the Internet. These include message-oriented communication, SOAP-based web services and the HTTP RESTful paradigm. The proposed architecture uses a publically-available, component-based model called LooCI as it supporting middleware. LooCI provides a networking framework and an event bus abstraction and has already been deployed on a variety of platforms.

The proposed architecture consists of three tiers:

- Sensor tier – uses a sink node to send aggregated data from the WSN to a local gateway. It is the central node that bridges the WSN with the local proxy and as such is a potential single point of failure.
- Gateway tier – runs the Middleware on a local proxy and is independent of the WSN platform
- Cloud tier – uses the LooCI event bus to support publish and subscribe services

Experiments were run using a ZigBee XBee module as the sink node and an Arduino Uno with eight temperature sensors. Results were promising and showed that only 0.6 msec of latency is added.

2.6.2 Lightweight Middleware

A Springer journal article [18] examines options for a lightweight middleware for use in Wireless Medical Body Area Networks designed to run on mobile devices and act as a gateway to both receive sensor data and control sensor devices. The middleware presented provides:

- Data acquisition
- Dynamic plug and play (ad-hoc addition or removal of sensor nodes)
- Security to protect sensor data
- A Lightweight implementation to run at low power on mobile devices with limited processing power and memory
- On-the-fly reconfiguration of key parameters such as sample rates
- Resource control and management such as sensor sleep/reactivation a1nd battery levels

The article provides a comprehensive overview of different categorises of middleware and recommends a hybrid adaptive application-driven Middleware solution that is both Application reactive and Application proactive. The Middleware infrastructure uses a single gateway and does not address reliability concerns.

2.7 Reliability Terminology

There are a number of inter-related terms that are frequently used when discussing robust distributed systems. The following brief descriptions are based on explanations given in [3] and [19].

Availability	System is ready to be used immediately. Probability that the system is working correctly at any particular time and is therefore available to be used.
Fault	Something (system condition or state) that causes an error
Fault Tolerant	Dependable system that can tolerate faults i.e. it can still provide correct services when faults occur.
Maintainability	How easily a failed system can be repaired
Redundancy	Key technique for masking faults. With physical redundancy, additional software or hardware components are used as "spares" for backup purposes in case the primary component fails.
Reliability	The system can run continuously without failure. Defined in terms of a time interval. A system can still be reliable despite failures if it is fault tolerant through the use of redundancy techniques for example.
Risk	Probability of Failure x Severity.
Safety	When a safe system fails to operate correctly there is no disastrous impact. A totally safe system is free from risk.
Six 9's	System is available 99.9999% of the time

2.8 Chapter Summary

This chapter summarises the State of the Art in relation to issues relevant to eHealth and remote monitoring communications. Particular attention is focused on reliability and single points of failure in proposed solutions. An overview of Wireless Body Area Network communications technologies is also included.

As shown in the following table, this dissertation fits into a research gap covering redundant gateway communications without a single point of failure.

Topic **Paper**	**Event Bus**	**Gateway**	**Generic Sensor Network**	**No Single Points of Failure**	**Redundant Gateway Comms**
Middleware **2.6**	√	√	√	X Sink Node or Sensor Gateway is SFP	X
ZigBee Fault Tolerance **2.5.1**		√	√		√ Outstanding issue re dual coordinators
This dissertation	√	√	√	√	√

Table 3 Dissertation research area

Chapter 3 Design

This chapter discusses design issues and proposes a protocol that is designed to improve the reliability of communications between a Body Area Sensor Network (BASN) and a Remote Monitoring System (RMS). A communications configuration is also proposed with primary and secondary independent links between the BASN and the RMS; the secondary link acts as a backup to the primary. Both the proposed communications protocol and network configuration are designed to improve reliability so that alerts are not lost when communication failures occur.

Due to advances in technology there are many options to be considered when designing a BASN and as a result a number of hardware, software and communications decisions must be made when implementing a prototype. The primary concern of this project is reliability and this requirement, above all others, was the critical factor for all decisions that had to be made when designing the system.

The sections in this chapter are organised as follows:

Section	Contents
3.1	Outline of the overall requirements that should be met by the design for a prototype and additional requirements that a production system would need to consider.
3.2	Discussion regarding potential topologies for the BASN.
3.3	Outline of the communications configuration proposed to meet the requirements.
3.4	A detailed description of the proposed protocol for providing reliable communication between a BASN and RMS. This section includes explanation of the functions of gateway nodes within the BASN and the messages that are sent between nodes.
3.5	Failure Mode and Effects Analysis.
3.6	Chapter summary.

Table 4 Sections in Chapter 3

3.1 Requirements

The project's key reliability requirement can be summarised succinctly as follows: all significant health-related events must be communicated to an external Remote Monitoring Service (RMS) in a timely fashion.

With any system, there are times when delays are inevitable e.g. when communication links are down or not fully operational. Wireless communications based on radio-frequency electromagnetic waves also suffer from additional problems caused by interference and signal attenuation. No specific system performance goals were set for the prototype. For a production system, specific performance targets would need to be set to state what acceptable timings would be for an alert to reach a Remote Monitoring System (RMS) under normal and worst case operating conditions.

A production system would also have many additional requirements that would have to be taken into account when making design decisions e.g.

- Safety – any system that makes human contact must be safe to use and comply with SAR (Specific Absorption Rate) regulations.

- Fault Tolerance – this requirement is closely related to the reliability requirement as a system cannot be reliable if it does not tolerate faults. In particular, an important design goal os to avoid Single Points of Failure whenever possible.

- Mobility – when selecting hardware components it is important to consider aspects such as power supply, size, weight and wired connections as the user's freedom of movement should not be restricted and they should be able to continue their normal activities of daily living without too much intrusion. In short, the hardware components and associated communications equipment should be as unobtrusive as possible and user friendly in terms of ease of fitting, putting on and taking off.

- Interoperability – the BAN must be able to operate in close proximity to other systems. In particular the BAN communications must not be affected adversely by other systems commonly found in the home e.g. Wi-Fi, microwave ovens.

- Power Consumption – like all embedded systems, battery power is a limited resource and it is essential that energy is conserved as batteries may not be easily replaced or recharged. The

prototype implements an energy-aware protocol by trying to reduce repetitive broadcast rates whenever possible. Monitoring within the BASN is carried out repeatedly. Therefore status-checking must also be performed in an efficient manner so as not to exhaust battery power.

3.2 Wireless BASN Topology Options

The project encompasses a Body Area Sensor Network. The proposed communications protocol is designed to work with a generic BASN configuration. Nevertheless it is important that the BASN itself is reliable. One of the first decisions to make when considering the design of a BASN is its network topology as the topology influences what communication technologies are appropriate.

In addition to deciding what network topology to use, it is also necessary to consider what radio frequency (RF) bands the wireless network uses. Reliability may suffer if the selected radio frequencies are jammed or blocked by other networks or devices in the home, or even within the same BASN.

Although there is research into non-RF communication methods for BASNs, such as ultrasound [20], this dissertation only considers RF technologies as none of the prototype sensors are implanted and RF equipment is readily available.

When any network is set up or being reconfigured, the nodes are physically and logically organised into a particular structure. In wired networks it is particularly important to decide what the physical network topology is in advance, as the nodes are connected together by cables. In wireless networks, even if the nodes are mobile, the topology is also important as it affects what communication technologies can be used and how nodes can communicate with each other. Some common topologies are shown in the following figure and are discussed in the subsections below. There are many other topologies and hybrid combinations which are not covered here.

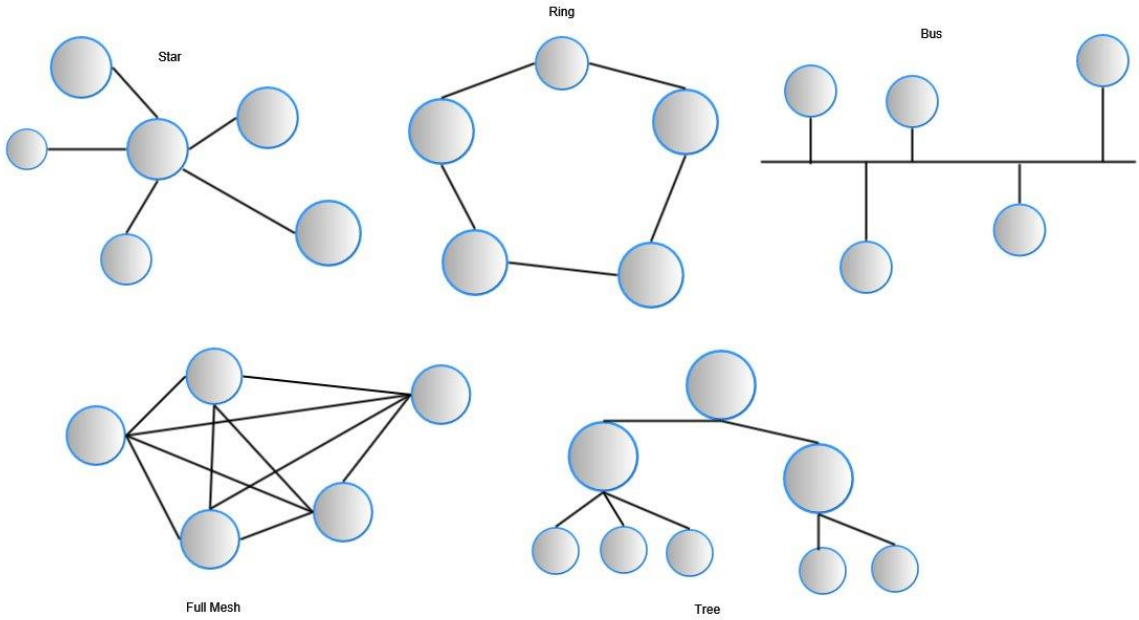

Figure 1 Network Topology Examples

3.2.1 Star

In a star configuration there is one central node that is usually the master node in the network. All other nodes are slaves and have to communicate via the master, there is no direct peer-to-peer communication among slaves. The master can therefore cause congestion, as well as being a single point of failure.

In a Bluetooth network this star topology is called a Piconet. A Bluetooth slave can belong to more than one Piconet, thus forming a Scatternet. However in all cases, the central master is a single point of failure and for this reason a wireless BASN based primarily on Bluetooth is not considered reliable for the prototype.

3.2.2 Ring

In a ring topology, each node connects to exactly two other nodes so that all nodes form a circle. Data has to travel around the ring through all intermediary nodes in-between the sender and the receiver. The entire ring is disrupted by the failure of a single link if data flow is unidirectional. Although data could flow clockwise and anticlockwise in a wired ring, a wireless ring cannot be full-duplex due to self-interference (wireless network nodes are generally half-duplex). A ring topology is not considered sufficiently reliable or efficient for a BASN as nodes are resource-deprived and could potentially waste both time and resources having to perform routing functions.

3.2.3 Bus

In a bus topology, all the nodes are connected to a common transmission medium which has exactly two endpoints. Schemes to avoid or handle collisions are needed in bus topologies as multiple senders may need to transmit data at the same time. The bus topology is commonly used in wired Ethernet networks. Wireless networks could be viewed as bus networks with radio waves being the shared medium. A wireless bus topology is effectively equivalent to a full mesh (operating on one channel).

3.2.4 Full Mesh

Every node in a full mesh network has a direct point-to-point link to every other node in the network. It is a peer-to-peer network and there is no central master. If one node fails then the remaining nodes can continue to communicate regardless. A full mesh allows messages to be communicated efficiently and it supports reliable communication as there are no single points of failure and there are redundant communication paths between all nodes. ZigBee supports mesh topologies and this was the main reason for choosing ZigBee for communications within the BASN.

3.2.5 Tree

There is a root node at the top of a tree topology with a branch and leaf structure below. Messages travel up and down the tree between senders and receivers. Leaf nodes are not directly connected so message propagation is not as efficient as a full mesh. Also, if a parent node fails then its children cannot communicate with any other nodes in the network. ZigBee also supports the tree topology but a ZigBee mesh was chosen for the BASN as it is more reliable.

3.3 Communications Configuration

The approach taken in this project to improve communications reliability is to adopt a traditional strategy of employing redundant communication channels. Although it may be costly in monetary terms to have backup links that are underutilised, this is the usual price of ensuring reliability.

The proposed communications configuration between the BASN and the Remote Monitoring System is illustrated in **Figure 2.** Note that there are two independent communication links between the BASN and the RMS:

- The primary link represents the preferred route of traffic between the BASN and the RMS;
- The secondary link is redundant during normal operation and is used for backup purposes

The proposed configuration contains key components called gateways which are described in greater detail in the following sections.

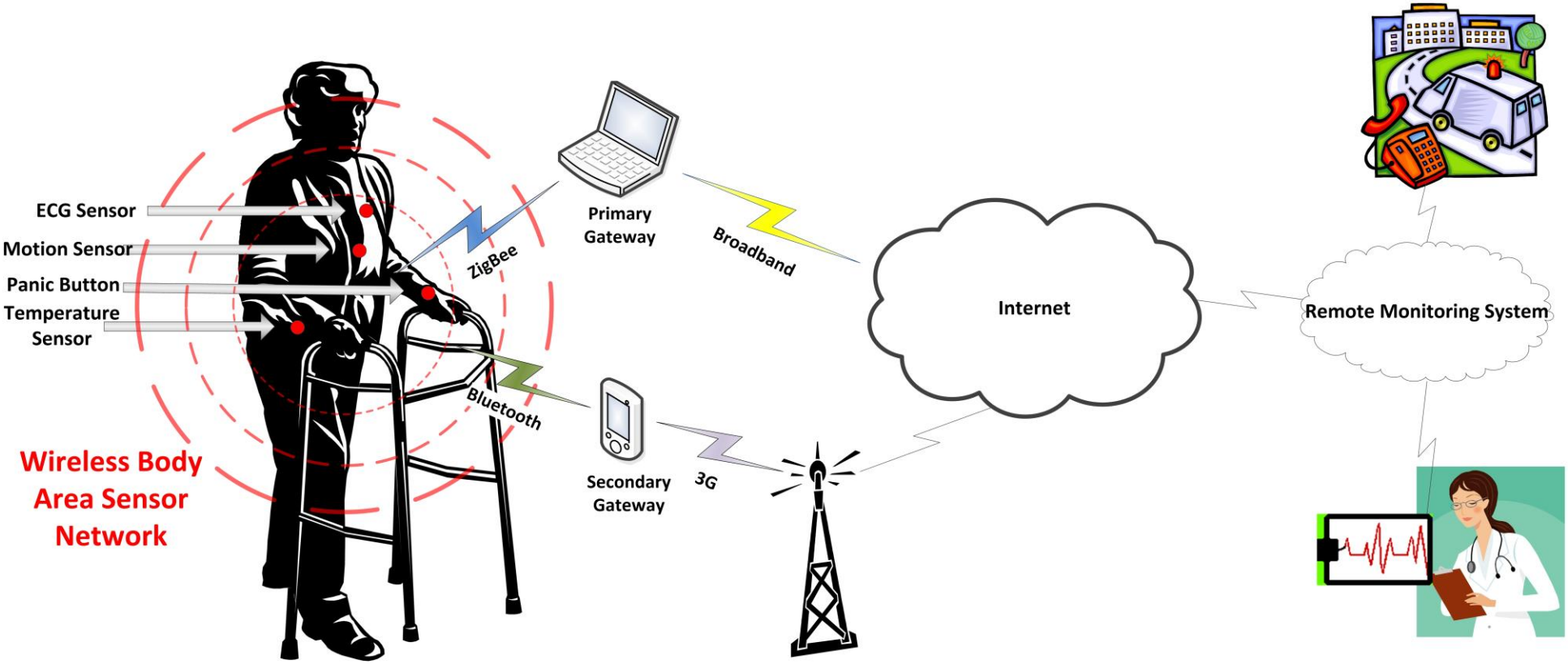

Figure 2 Proposed Communications Configuration for eHealth

3.4 Proposed Protocol

The following subsections propose a protocol that is designed to work with the configuration described in **Section 3.3** and is intended to improve reliability of communications between a BASN and a RMS. The protocol supports the provision of an External Alerting (EA) service to nodes in the BASN so that any source node can send an alert to a RMS via one or more local gateway nodes.

3.4.1 Gateway Nodes

A gateway is needed to act as an intermediary between the Body Area Sensor Network and the Remote Monitoring System. A gateway node has two important functions:

- Provides a communications "hub" supporting links to both the BASN and the RMS.
  For example, sensor nodes in a BASN are generally resource constrained and would not have broadband communication capabilities. However if a BASN node connects (wirelessly) to a gateway, then it can use the gateway's additional communications capabilities.

- Acts as a translator between BASN and RMS data formats.
  For example, in the prototype, event details from a node in the BASN are encapsulated in a ZigBee frame and need to be reformatted for interpretation by the RMS. Likewise, responses from the RMS need to be reformatted and embedded inside a ZigBee frame before it can be returned to the source node.

In order to avoid single points of failure, more than one gateway is required. A second gateway acts as a backup for a primary gateway and can also reduce the risk of a complete communications outage by using different external communication links that are completely independent of the primary's means of communication. For example, a primary gateway might use broadband from one company and the secondary gateway might use a 3G mobile service provided by a completely different operator. The BASN would then have two independent means of connecting with the Remote Monitoring System. Note that precise physical details of the communications links between the gateway nodes and a RMS are not covered in this dissertation, the emphasis is more on the characteristics of these links i.e. they should be independent of each other and provide secure and reliable (fully acknowledged) transport options, such as support for TCP connections for example.

The proposed communications configuration requires a minimum of two gateway nodes, at least one of which should be mains powered and would require the installation of a smart box/basic computer in the user's home. There could be more than two gateways if additional communication options were available for communicating with the Remote Monitoring System. The more independent gateways there are, the lower the chance of a complete communications outage and the higher both the system's reliability and the cost of redundancy.

3.4.2 Gateway Ranking System

Gateways are categorised in this dissertation according to whether they are primary, secondary or tertiary and so on. This categorisation is referred to as the gateway's ranking. The following table explains the proposed ranking system.

Ranking	Category	Meaning	Health Status
0	Unavailable	Gateway process is running but is currently not operational e.g. its external comms links may be down. The gateway is unable to provide an External Alerting (EA) service and should not be used by any source node that is generating an alert.	Non-zero = Error code
1 Highest	Primary	This gateway is offering the best level of service. It should be mains powered (with a battery backup) and have fully operational external communications links.	0 =>No errors Non-zero = Error code
2	Secondary	This gateway is offering the second best level of service. It may only be battery powered but it should have fully operational external communications links.	
3	Tertiary	This gateway is offering the third best level of service. It may only be battery powered but its external communications links are less desirable to use, perhaps they are more expensive or costly in some other way.	

Ranking	Category	Meaning	Health Status
3<n<255	Low	It is not anticipated that there would usually be gateways with rankings >=4 but these values are possible, perhaps for future extensions of the protocol. May be useful if a gateway's power supply is running low and it is trying to avoid non-critical processing for example.	
255 Lowest	Last – included for completeness	This is the lowest possible ranking. It should only be used as a last resort when all higher ranked gateways are not working. This gateway may have no external communications links itself but perhaps it has logging/storage facilities and could store alerts until a higher ranking gateway became available.	

Table 5 Gateway Ranking System

There are a few important points to note about the proposed ranking system:

1) It is proposed that each gateway determines its own ranking. A common ranking algorithm could easily be implemented in software across all gateway nodes. It would take into account how the gateway is powered (mains or battery supply) and have weights associated with each external communications option e.g. broadband might be weighted more favourably than 3G. The algorithm could be tuned as well to take into account the quality of various mobile operator's services as coverage can vary from area to area and would therefore vary according to where the user lived.

2) Each gateway should recalculate its ranking at runtime according to its current operational environment. A battery powered gateway may lower its ranking if its battery is running low for example. If a gateway loses all its external communications capabilities then it should set its ranking to zero with an appropriate non-zero error code in its associated health status field.

3) There should be no restrictions on how many gateways have the same ranking i.e. unique rankings should not be enforced. It could be very useful to have two secondary gateways for example and this would allow source nodes the opportunity to implement load balancing algorithms across multiple gateways which could be desirable for performance reasons.

4) Each gateway should regularly broadcast its current ranking and health status, in a heartbeat message, to all nodes in the BASN. The broadcast rate depends on the gateway's power supply. The primary gateway should be mains powered and could broadcast every 30 seconds for example. Battery powered gateways may broadcast less frequently, perhaps every 60 seconds or even less often for lower ranking gateways. In all cases the broadcast message should be kept as small as possible in order to keep resource usage to a minimum.

5) Each source node is empowered to select the best gateway (which is the gateway with a non-zero ranking equal to or closest to 1) whenever an alert is being generated for the Remote Monitoring System. This means that each node should maintain its own record of the current status of all gateways. This information should be updated in memory at runtime according to:

   - The last broadcast received from a gateway
   - The outcome of the node's last alert transmission

   For example, the last broadcast from a gateway could have indicated that its ranking and status were good and a node could have sent an alert to that gateway which timed out unexpectedly. The node could then adjust its own local view of that gateway's status accordingly and attempt to use the next best ranked gateway for the retransmission.

6) There are a couple of quirks that could potentially happen in the unlikely, but possible, situation that a source node detects an urgent event on start-up. For example, if a node needs to send an alert to the RMS *before* it has received its first EA broadcast from a gateway, then this first alert will be delayed. The maximum length of the delay equals the broadcast frequency of the most frequent broadcaster, which would ordinarily be the primary gateway's broadcast frequency as it is mains powered and can afford to broadcast more frequently than battery-powered gateways.

7) Also worth noting is the fact that broadcasts from gateways are not synchronised so there is no guarantee regarding the order in which a node will receive EA broadcasts. In the specific scenario described in the previous point, on start-up a node could receive its first broadcast from any of the gateways and could therefore send its first alert to a gateway which did not have the best ranking. The end result is that the alert should reach the RMS but it may not have been routed there using the preferred communication's links.

3.4.3 Gateway Heartbeat Message

As mentioned in **Section 3.4.2,** each gateway regularly broadcasts its current status to all nodes in the Body Area Sensor network. This heartbeat message is 5 bytes in size. The format of the message is as follows:

Byte Offset	Field Name	Field Description	Example Value
0	Service Code	2-character code advertising the service being offered by the gateway i.e. "EA" for External Alerting service.	EA
1			
2	Ranking	The gateway's current self-ranked value as described in **Section 3.4.2** e.g. primary gateway would have a ranking of 1, secondary gateway would have a ranking of 2 etc.	1
3	Health Status	Code to indicate the gateway's status. Zero indicates no errors and should be used whenever the gateway's ranking is non-zero. If the gateway is not fully operational and its ranking is set to zero then the Health Status should be non-zero to indicate the reason for the gateway's non-operational status.	0
4	Node Id	This field is not absolutely essential but is a useful flag in development to confirm which node has sent the broadcast. XBee Node Identifiers (NI) can be 20-bytes long, this field is a 1-bye abbreviation and could be removed.	P

Table 6 Gateway Heartbeat Message Structure

This message can clearly be optimised (to be smaller in size) but its simple structure was adequate for development of the prototype.

The heartbeat is effectively the mechanism used by each gateway to advertise its EA (External Altering) service provision status. Otherwise nodes in the BASN would not know any details regarding which gateway nodes were available, or what the gateways' addresses were etc.

On receipt of an EA advertisement/heartbeat message from a gateway node, a source node can extract the sender's source address details from the encapsulating frame and can use these details as destination addresses when unicasting an alert to the RMS via the chosen gateway.

Future versions of the protocol could use the Service Code field to advertise additional services. For example, the primary gateway might also provide data encryption or compression services etc.

3.4.4 Generic Gateway Functionality

This section specifies the high-level functionality that each gateway implements, regardless of whether it is a primary, secondary or tertiary gateway etc. The following pseudo code outlines a generic algorithm for each gateway offering an External Alerting (EA) service.

Start-up:

```
initEASP ();                    // Store details about all other EA Service Providers (SP)
freq = calculateFreq ();        // Determine appropriate broadcast rate
calculateRanking ();            // Determine own Ranking & Health Status values
broadcastHB ();                 // Send HB e.g. "EA20S" to nodes currently in the BASN
lastBroadCastTime = now ();     // Note time of last broadcast
```

Loop:

```
if ( ( lastBroadCastTime + freq <= now () ) OR
    ( request received for an up-to-date heartbeat ) ) {
        calculateRanking ();        // This should include checking own external comms links
        broadcastHB ();
        lastBroadCastTime = now ();
}

if ( EA broadcast received from another gateway ) {
        updateEASP ();              // Maintain up-to-date details re other EA SPs
        if ( another gateway's ranking has just been downgraded )
                sendAlert ("problem with gateway <X>");
}

if ( alert received from BASN node ) {
        sendAlertToRMS();
        if ( there are operational higher ranking gateways ) {
                requestHeartBeats();        // Check why alert wasn't sent to "better" gateway
                if (requested heart beats not received)
                        sendAlert ("problem with  gateway <X>");
```

```
            }
        }

        checkSensors();             // Some gateways may have their own sensors
        if ( urgent event detected )
            sendAlert();

        for (each active alert that has been sent to another gateway) {
            if ( Acknowledgement received from other gateway )
                startTimerForRMSAck();
            else if ( Acknowledgment has timed out or Negative Acknowledgement received ) {
                updateEASP ();
                sendAlert(set duplicate flag );
            }
        }
        for (each active alert that is awaiting RMS ACK) {
            if  ( RMS ACK received ) {
                if (RMS ACK is for another node)
                    sendRMSAckToSourceNode();
                markEventAsProcessed();
            }
            else if ( RMS ACK not received within RMS Timeout ) {
                updateEASP ();
                sendAlert(set duplicate flag );
            }
        }

sendAlert() {
        if ( there are operational higher ranking gateways )
            sendAlertToBestGateWay();
        else if ( own status is OK ) // This gateway is the best so send the alert directly to RMS
            sendAlertToRMS();
        else if ( there are any operational gateways of equal or lower ranking )
            sendAlertToBestGateWay();
        else
            storeAlertUntilGatewayAvailable();
}
```

Figure 3 Generic Gateway Algorithm

3.4.5 Primary Gateway Data Flow Example

Figure 4 illustrates the data flow for a system with operational primary and secondary gateways and a source node that has detected an urgent event meriting external alerting. In this case the source node is attempting to send the alert to the Remote Monitoring System via the Primary Gateway.

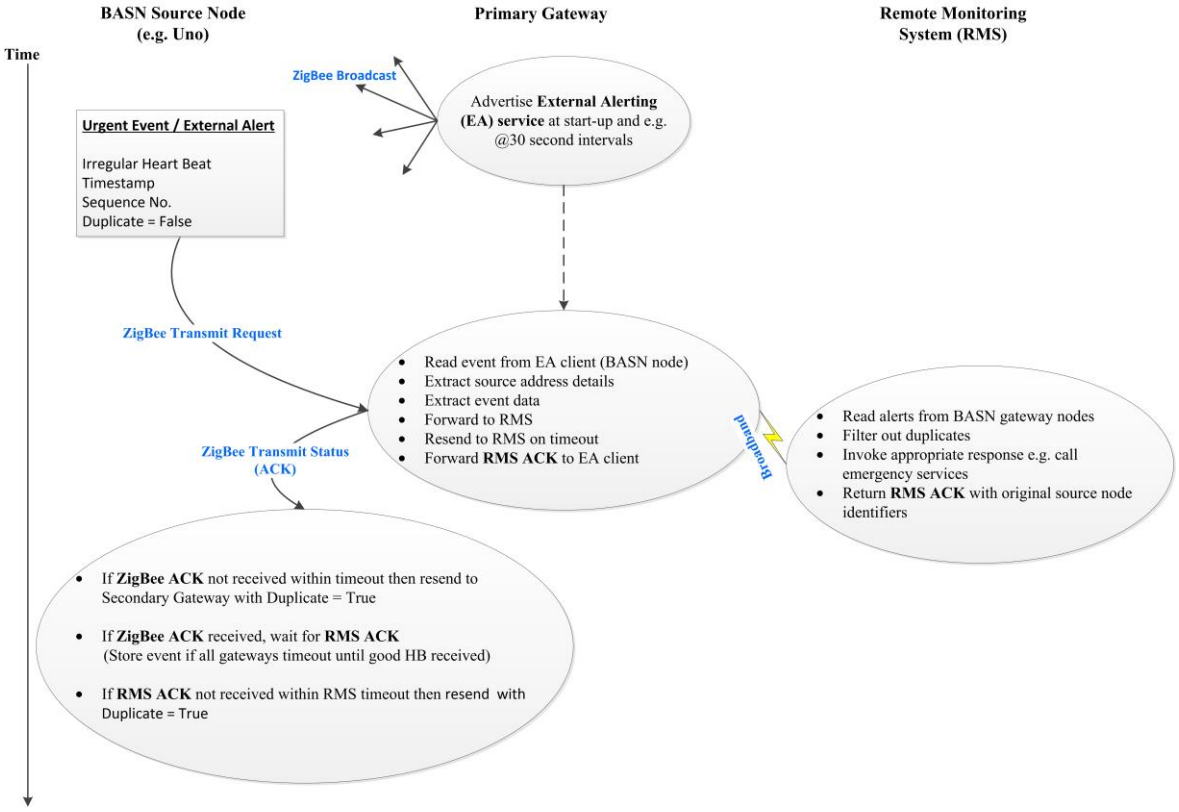

Figure 4 Message flow from BASN node to RMS via Primary Gateway

3.4.6 Protocol Features

The proposed communications protocol has a number of features:

1) **It is event-oriented**

   Sensors can generate huge volumes of data. Most of the time, each sensor reading has a normal value and is not particularly relevant. For example, in the case of an eHealth system it would not normally be worthwhile for the RMS to receive a real-time copy of the user's current temperature if the temperature readings are within a normal healthy range. In other words, it is not necessary to transmit every single sensor reading to the RMS.

Sending continuous data streams to the RMS would unnecessarily consume battery power on the BASN nodes, communications resources on the links between the BASN and the RMS and processing and database resources on the RMS itself. It would also make identifying significant data more difficult, as relevant sensor readings could be lost in among all the less relevant readings. Also, urgent messages with significant data values could be delayed by congestion caused by the proportionally higher volumes of lower-priority messages.

For these reasons, the proposed protocol sends event messages, not continuous streams of data messages. When a node in the BASN detects a significant situation, such as the user's temperature falling outside a normal range, an event message is generated and this is forwarded to the RMS straightaway. For some events, such as an EKG rated event, it may also be useful to include some older sensor readings that were taken immediately prior to the abnormal reading, while for over events, such as the panic button being pressed, there may be no other relevant historic information to include.

2) It uses acknowledgements and timeouts to improve reliability

Each time a message is sent, an acknowledgement is expected to confirm that the message has reached the intended recipient. If the sender receives an acknowledgement then it knows that the receiver definitely got the message. However if an acknowledgement is not received then the sender assumes that the receiver did not get the message and the sender retransmits the message again, with a duplicate flag set. The receiver may end up getting the same message more than once as a result but this mechanism ensures that messages are definitely received by the intended recipients. Moreover, the sender cannot wait too long for each acknowledgement (otherwise alerts could be significantly delayed). There are two timeouts for each source node to consider when sending messages:

- **The timeout for an acknowledgment from a gateway node**

  This timeout should be very short as nodes within the BASN should usually be able to deliver messages to each other well under a second, under normal operating conditions. Note that the acknowledgements at this level are returned by the underlying network which is ZigBee in the case of the prototype. Negative acknowledgments may also be returned. If the sender receives a negative acknowledgment or times out waiting for an acknowledgment then it retransmits the message with a duplicate flag to another gateway.

- **The timeout for an acknowledgment from the RMS**

  Once a source node receives a positive acknowledgment from a gateway node, it can then start a timer for when a further acknowledgement should be received from the RMS. This timeout period will be longer than the BASN timeout, as communications with the RMS

may be over a considerable distance and the RMS will probably have lots of clients. However for an eHealth system to be safe, it is anticipated that a RMS should be able to return an acknowledgment within, say, 20 seconds, the exact value would have to be carefully calculated in order to ensure that the source/gateway nodes do not generate too many duplicates while at the same time an adequate response is instigated quickly enough to deal with the BASN's alert.

3) **It empowers nodes in the BASN to select the best gateway to improve reliability**

Risk is distributed across the BASN as each node maintains its own view of the status of each gateway based on the broadcasts it has received and the last positive/negative acknowledgements it has received.

4) **It allows each gateway to rank its own service provision capabilities**

Each gateway should know the status of its own external communication links to the RMS. This status is dynamic and changes at runtime. A gateway updates its heartbeat message to reflect its operational status and this is broadcast regularly to all of the BASN nodes. If a gateway fails and so is unable to broadcast, then a lower ranking gateway will notify the RMS.

5) **It facilitates load-balancing algorithms**

As unique rankings are not enforced, this means that source nodes could have more than one gateway with the same ranking to select. In this case, nodes could implement a simple round-robin algorithm to distribute the load across gateways of equal ranking. This could be beneficial for performance and resource usage reasons.

6) **It is scalable**

The protocol will work with one or more BASN nodes and two or more gateways. One or more of the gateways can be sensor nodes in the BASN but the primary gateway should be mains powered as it sends more frequent broadcasts and will probably have external broadband links.

7) **It is flexible**

BASN nodes can start and stop, gateways can start and stop, a gateway's status can change, there can be different numbers of gateways etc. The protocol covers all of these scenarios.

8) **It is extensible**

Although the proposed protocol deals with the provision of an External Alerting service to nodes in the BASN, it could easily be extended to cover other services such as data compression or encryption etc.

9) It is a first-step towards a full-subscription based protocol

When a gateway advertises its EA service to BASN nodes, it is effectively issuing a subscription request although nodes may choose to send their alerts to a different gateway, if a higher ranking gateway is available. The underlying messaging functionality that has been implemented could be used as the basis for a full subscription protocol to improve reliability even further. This is discussed in **Section 6.2.2** as a potential area for future work.

3.4.7 Event Message Format

As previously discussed, only messages regarding significant events need to be communicated to the RMS. A very basic 10-byte structure to hold event data is given in the following table.

Byte Offset	Field Name	Field Description	Example Value
0	eventType	Code to specify what type of event has occurred. Some initial suggestions for codes are: ePanicButton 1 eLowTemp 2 eHighTemp 3 eEkg 4 eNotMoving 5	1
1	currSeq	Current Sequence Number – each node has its own local sequence no. to track its own local events. Duplicate events share the same sequence no.	100
2	dup	Boolean indicating whether or not this event is a possible duplicate. If a node has to resend an event then this flag should be set to True.	False (0)
3	timestamp	Time that the alert was originally detected by the source node	819136
4			
5			
6			
7	value1	Data Value 1 e.g. current temperature reading	38
8	value2	Data Value 2 e.g. last temperature	37
9	value3	Data Value 3 e.g. second last temperature	37

Table 7 Simple 10-byte Event Message Structure

There are a few points to note about this event data structure:

- Although it is very basic, it is sufficient to allow the first version of a prototype to be implemented.
- The timestamp field is catering for nodes in the BASN that do not have a real-time clock. The value in this field relates to the number of milliseconds that the node has been running, not an absolute time that can be correlated against another node's clock or the RMS's clock etc.
- There are a maximum of 3 bytes of data values. These may be empty if not required e.g. if the panic button was pressed then there are no associated data values.
- There is no priority field as all alerts are considered equally urgent.
- The structure can easily be modified to add new fields and change field sizes etc. The structure should be kept as small as possible for performance and resource usage reasons. It also needs to fit inside a ZigBee frame. The maximum payload size is 84 bytes approximately[6].

3.4.8 Event State Transitions

The following state transition diagram illustrates the states that an event can move through on any BASN node. An event is created when an urgent situation is detected and the RMS needs to be notified. An event can pass through a number of states waiting for ZigBee and RMS acknowledgments or resending event messages if negative acknowledgments are received or timeouts occur. It is not fully processed until an acknowledgment has been received from the RMS.

The arrows between states indicate how an event moves from one state to another.

[6] The XBee **AT NP** [26] command gives the maximum number of RF payload bytes that can be sent in a unicast transmission. This value was 0x54 for the prototype but could be lower for example if encryption was used.

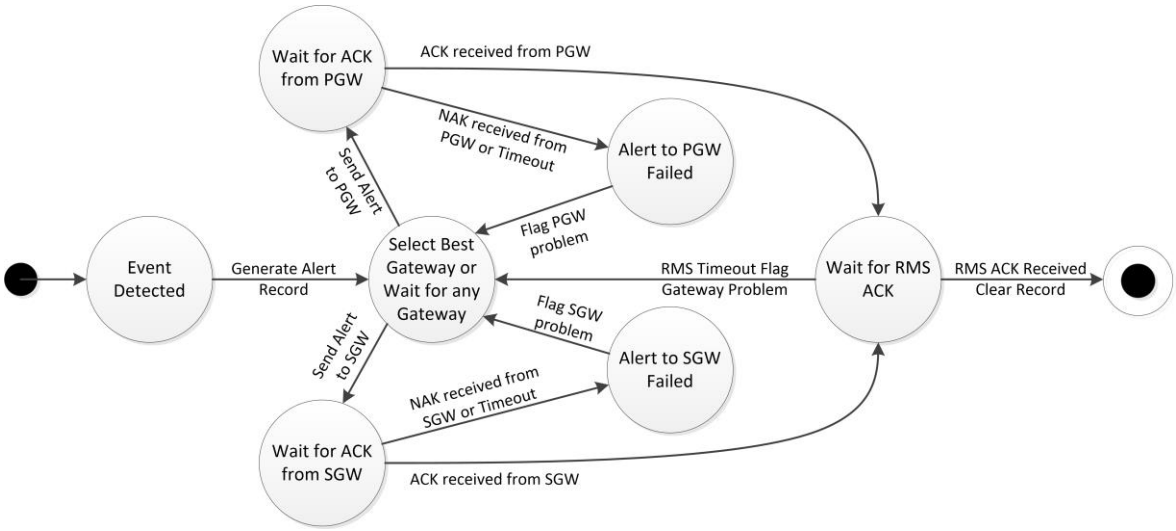

Figure 5 Event State Transition Diagram

3.5 Risk Analysis

Failure Modes, Effects and Criticality Analysis or Failure Mode and Effects Analysis (FMEA) is a tool to access the vulnerability of a design to real-world faults [19]. It allows systems to be designed for reliability from the outset through the identification of anticipated potential failures. Appropriate contingencies can then be planned to cover any identified failure modes. This can be done at many different levels, anywhere from the lowest electronic circuit designs, to high-level application functionality, to business processes. In all cases, failures should be prioritised according to how critical the impact is. FMEA is also used within healthcare to access healthcare processes, although its use is not always recommended as the tool's validity is not proven [21]. Nonetheless, it is a useful tool to help design and plan for reliability.

The following table shows how FMEA can be used at a high system level. By designing to improve reliability through redundancy, the impact of communication failures is minimised. It also shows that a contingency is needed (such as a logging service) so that the system does not completely fail if there is a total communications outage.

The severity of potential failures can be estimated more accurately if real probabilities are calculated for the occurrence of each individual failure. This could be done through mathematical models, statistics regarding historical failures, manufactures estimates for the mean time to failure and mean time between failures of components etc.

Failure Mode	Prob	Effect	Outage?	Severity	Priority	Contingency
(1) PGW loses Broadband (BB)	Low	Alerts cannot be delivered to RMS via BB	No	Med	2	SGW sends alerts via 3G
(2) SGW loses 3G	Med/ High	None as long as PGW working	No	Low	3	PGW sends alerts via BB
(1) and (2)	Low	Alerts cannot be delivered to RMS	Yes	High	1	Invoke a local logging service

Table 8 Failure Modes and Effects Analysis

3.6 Chapter Summary

This chapter described the proposed communications configuration and protocol to be used to improve the reliability of communications between a Body Area Sensor Network (BASN) and a Remote Monitoring System (RMS). The design uses redundancy to help improve reliability by reducing the risk of a total communications outage between the BASN and the RMS.

A key component in the design is a gateway node which acts as an intermediary between the BASN and the RMS. There must be at least two gateway nodes, each using different external communication links. The gateway nodes can be part of the BASN with or without their own sensors but at least one gateway must have a mains power supply and this is usually the primary gateway.

The following points briefly summarise how the primary and secondary gateways work:

- Gateways broadcast regular Heartbeat messages
- Secondary is a backup and takes over if Primary fails
- Clients (BASN Nodes) determine which gateway to use
- If the Secondary Gateway receives an alert then it: forwards the alert to the RMS and requests an up-to-date Heartbeat from the Primary Gateway. If this request times out then the Secondary Gateway generates an alert indicating that the Primary Gateway is not working.

Finally, the following two timing diagrams illustrate some of the functionality that has been covered in this chapter.

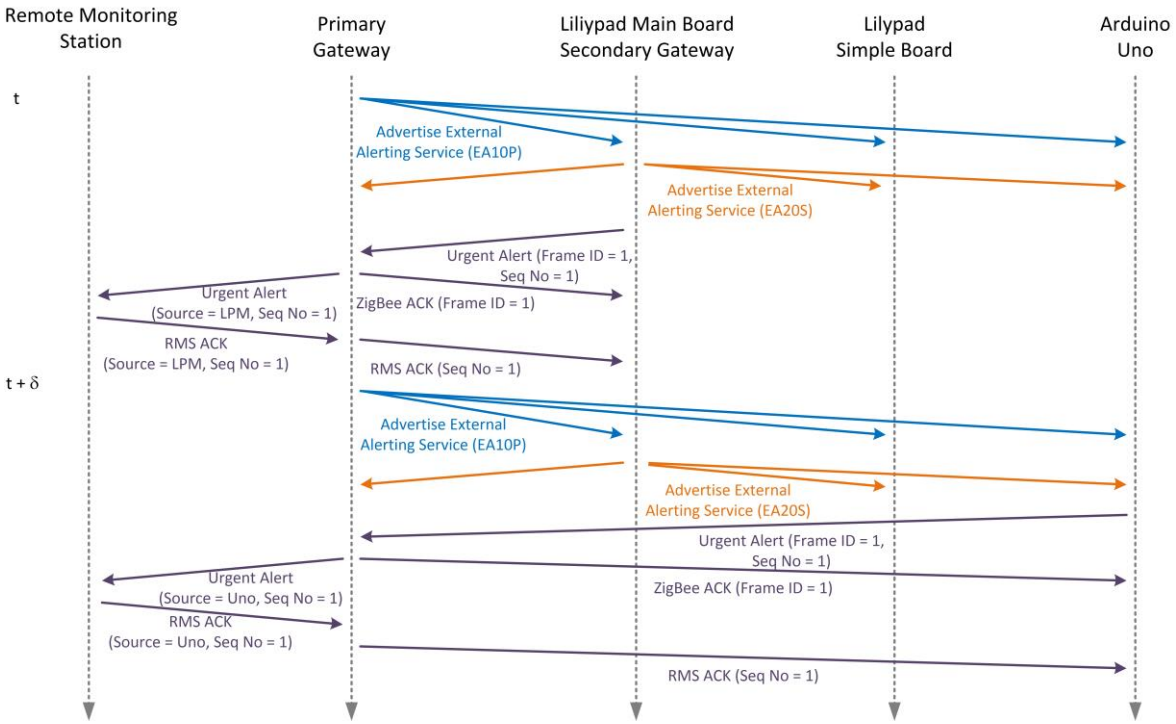

Figure 6 Timing Diagram - Normal Operation

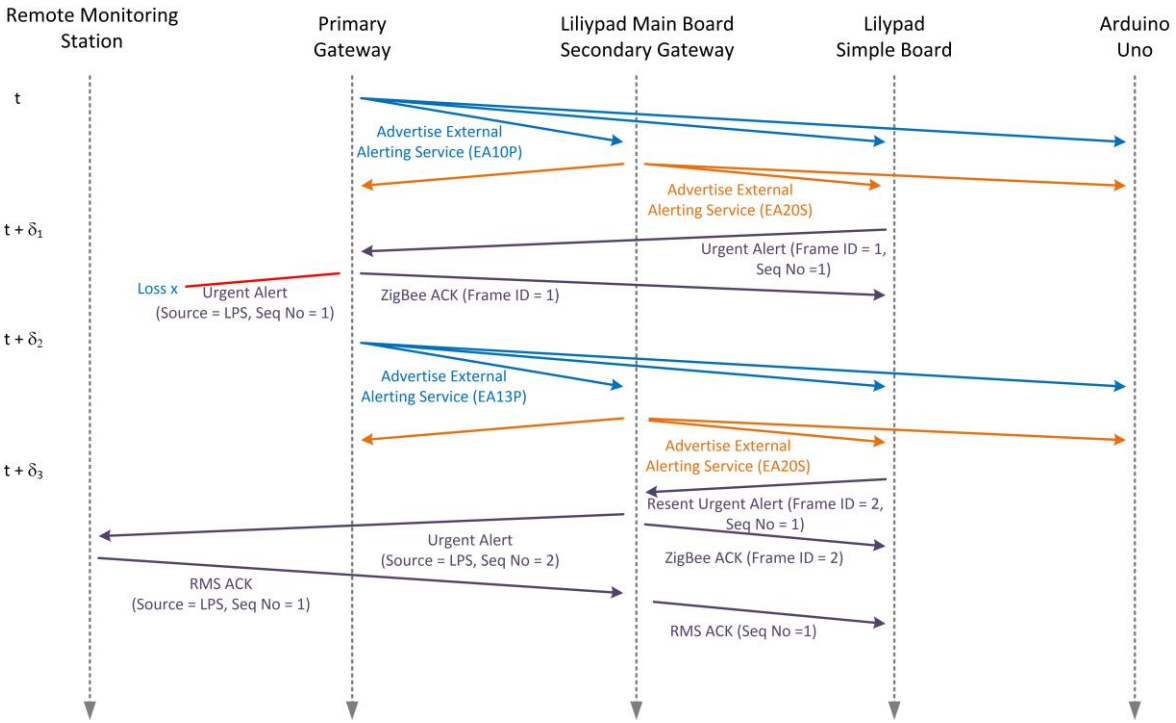

Figure 7 Timing Diagram - Example Message Loss

Chapter 4 Implementation

This chapter describes the implementation of a prototype that meets the design specifications outlined in **Chapter 3.** Note that the implementation was not completely finished as a number of technical issues arose quite late on in the development phase of the project which significantly delayed progress. These issues are described in **Section 5.3.** The sections in this chapter are organised as follows:

Section	Contents
5.1	Reasoning behind the choice of the Arduino Platform as the basis for the BASN components.
4.2	Description of the Arduino equipment used in the prototype.
4.3	Description of the sensor devices used in the prototype.
4.4	Information about the conductive thread used to make connections between the LilyPad prototype components.
4.5	Details of the configuration of the prototype.
4.6	Details of the ZigBee configuration of the BASN nodes and the libraries used in the implementation of communications using ZigBee.
4.7	Details of the components providing Bluetooth (as a redundant backup method of communication) and their configuration.
4.8	Information about serial communication between components and why hardware and software serial ports are needed.
4.9	Description of all the software components that form the prototype.
4.10	Chapter summary

Table 9 Sections in Chapter 4

4.1 Arduino Platform

Arduino was chosen as the prototyping platform for the Body Area Sensor Network as:

1) Arduino boards are readily available and the Arduino LilyPad range of electronic devices are designed for eTextile projects, as they can be stitched to clothing and connected together using conductive thread.
2) There is a good selection of compatible sensors and communications devices available from different suppliers.
3) The open-source Arduino IDE for Windows and Mac OS X allows the boards to be programmed using the Arduino programming language (similar to C/C++) and there are many libraries freely available that provide functionality for serial communications, interrupts, timers, communication with XBee radios etc.
4) There is some documentation available online and a user forum which can sometimes be useful for helping solve development issues.

4.2 Prototype Equipment

The following table gives an overview of all the equipment that was used for the prototype Body Area Sensor Network. These components are discussed in more detail in the following subsections.

Part	Brief Description
Arduino Uno – R3	5-volt board based on Atmel's ATmega328P microcontroller: 8-bit 16 MHz AVR CPU with 32KB flash. ATmega16U2 USB-to-TTL serial convertor. 14 digital and 6 analogue pins.
Olimex EKG Shield	Shield for the Uno to allow Electrocardiography signals to be captured on the Uno's analogue input pins. Heartbeat and pulse can be monitored by wearing electrodes which plug into the shield.
XBee Shield	Allows the Uno to communicate wirelessly using ZigBee as an XBee radio can be plugged in. Jumpers determine how the XBee's serial communication connects to the serial communication between the Atmega328 and the FTDI USB-to-serial chip (Atmega16U2) on the Uno.

Part	Brief Description
XBee RF Modules	Digi's XBee ZB radios where used for ZigBee communication in the Body Area Sensor Network. These operate in the 2.4 GHz band and have a variety of antenna options.
XBee Explorer	USB-to-serial base unit for an XBee RF module. Allows direct access to the serial and programming pins on the XBee so that it can be configured using tools such Digi's X-CTU or MoltoSenso's Network Manager etc.
LilyPad Main Board	Designed to be worn and used in eTextile projects. It is washable. 3.3-volt and is also based on the ATmega328P but runs at 8 MHz 6-pin FTDI header for programming and powering accessories such as a Bluetooth modem.
LilyPad LiPower	Power supply module for the LilyPad Main board. Has a JST socket for attaching a rechargeable 3.7V LiPo (lithium polymer) battery.
LilyPad Simple Board	Similar to the LilyPad Main board except that it has fewer pins and has an on-board power supply JST socket. Also, the FTDI header can only be used for programming.
XBee Breakout Boards	For the LilyPads so that they can communicate wirelessly using an XBee radio.
LilyPad Temperature Sensor	**MCP9700** temperature sensor that should output 0.5V at 0°C, 0.75V at 25°C, and 10mV per °C. Detects physical touch based on body heat and ambient conditions.
LilyPad Accelerometer	**ADXL335** three-axis accelerometer that outputs a 0V to 3V analogue signal on each of the X, Y, and Z pins. Used for basic motion sensing.
LilyPad Button Board	Simple, small button to represent a panic button.

Table 10 Prototype Hardware Components

4.2.1 Arduino Uno with XBee Shield

The Uno was used to provide EKG data wireless over ZigBee whenever an EKG-related event occurred. In practice, it was difficult to interpret the actual EKG data readings to determine whether or not significant event had occurred.

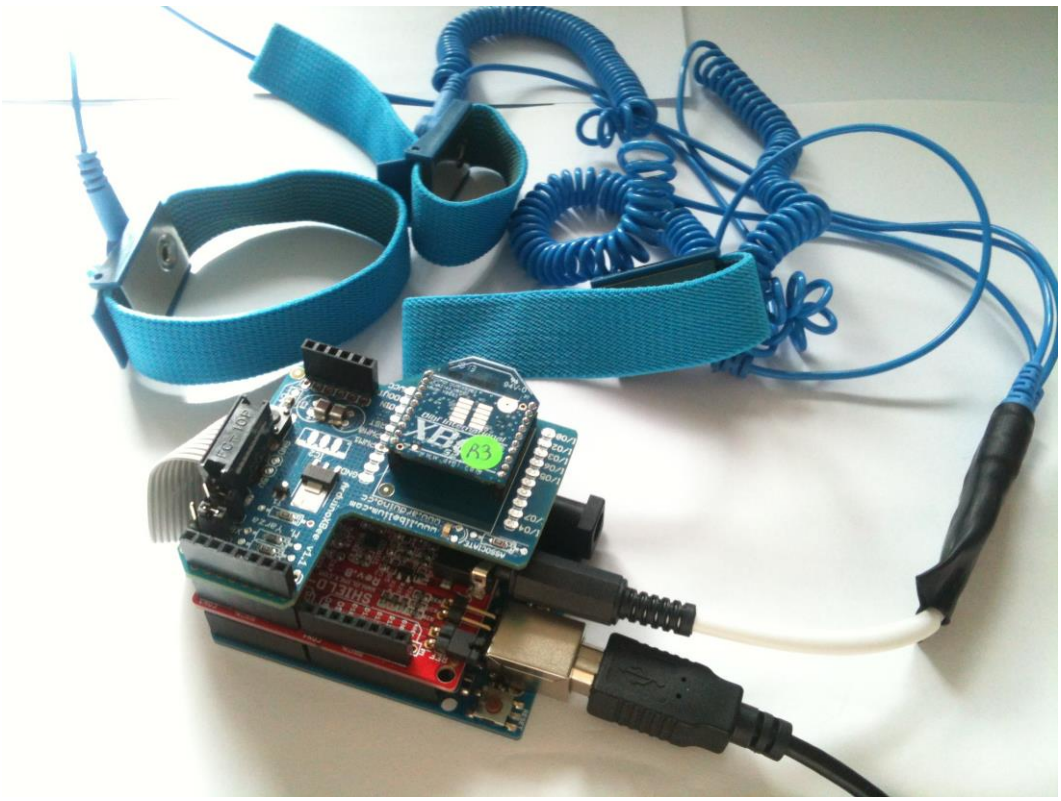

Figure 8 Arduino Uno with Olimex and XBee shields and EKG electrodes

4.2.2 Arduino LilyPad Main Board with XBee Breakout

The LilyPad main board was the most complex node in the BASN as:

- It is a sensor node in its own right – it has an accelerometer to detect movement
- It has two wireless communication mechanisms, Bluetooth and ZigBee, that have to work concurrently
- In addition to handling its own alerts, it has to handle alerts from other nodes when the Secondary Gateway is needed (i.e. whenever the primary gateway is not working)
- It was used as the Secondary Gateway so had to advertise its EA (External Alerting) service regularly and keep track of broadcasts from the Primary Gateway.

Figure 9 LilyPad Main Board with Power Supply, XBee radio, Bluetooth modem and Accelerometer

4.2.3 Arduino LilyPad Simple Board with XBee Breakout

The simple board was used to provide the processing power for a temperature sensor and panic button. This BASN node supported ZigBee wireless communication. It could not be used as a gateway node as it had no other wireless communication options.

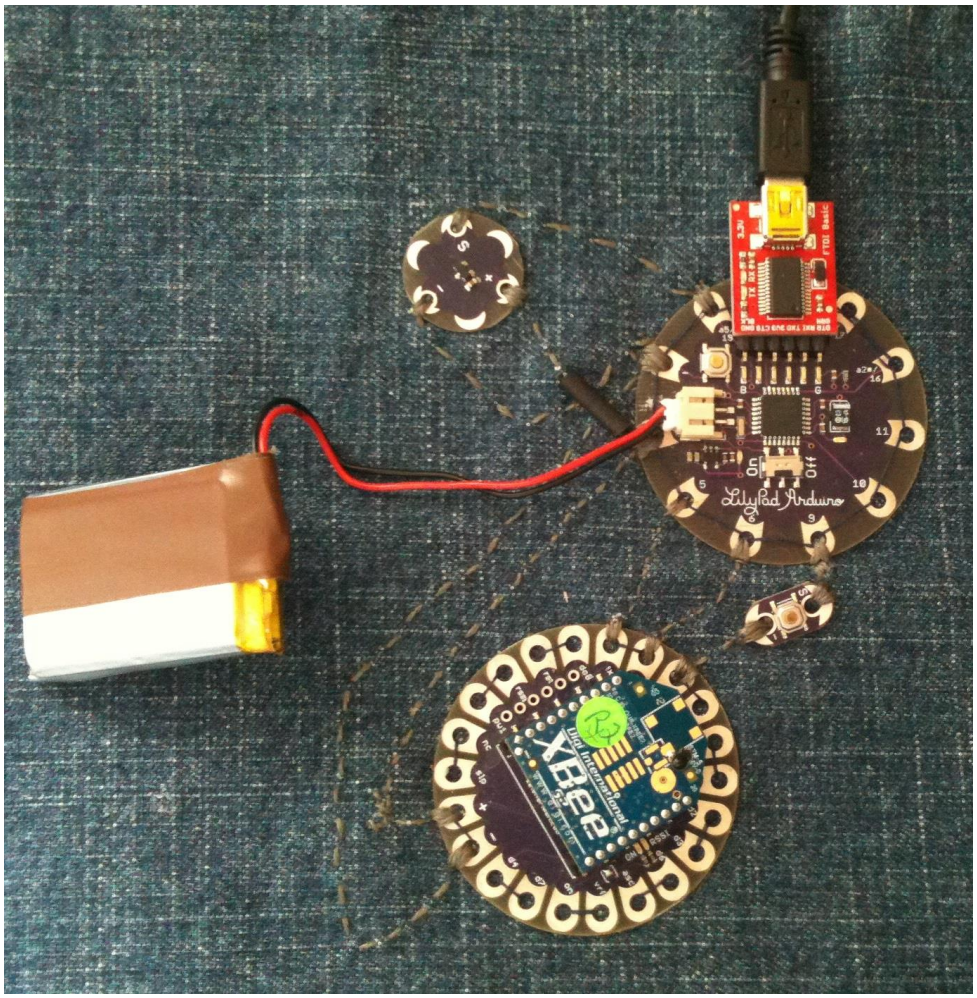

Figure 10 LilyPad Simple Board with XBee radio, Temperature sensor, Panic Button and FTDI Programmer

The LilyPad simple board has fewer pins than the main board so is more restrictive. In particular, the UART's TX and RX pins are not broken out which means that the hardware serial port has to be accessed via the FTDI header. Also, the battery charging circuitry (which did not work) restricted use of the FTDI header as it could not be used to power a Bluetooth modem for example.

4.3 Sensors

The prototype used four sensors which are detailed in the following subsections. The purpose of the sensors is to measure various physical conditions, such as ambient temperature, so that a connected microcontroller can monitor sensor readings and generate appropriate events if any readings are outside of a normal range or warrant alerting for some other reason (e.g. if the panic button is pressed). The prototype uses an event-based architecture so the specific sensors used in the project are not overly significant as the prototype supports events raised by any set of sensors.

4.3.1 Panic Button

A simple LilyPad button board[7] was used to represent a panic button. The button closes when pushed and opens when released (momentary push button). If the user presses the button then it is assumed that the user requires immediate medical attention because of some urgent event, maybe they have fallen etc.

The button was very small (8 x 16mm) and was connected to the LilyPad simple board using conductive thread. A panic button for a real system would be much bigger and more user-friendly. However the button was straightforward to program and caused panic button alerts to be generated when pressed.

4.3.2 Temperature Sensor

The Lilypad temperature sensor[8] uses a Microchip **MCP9700** [22] which is a low-power Linear Active Thermistor™ Integrated Circuit (IC) that converts temperature to analogue voltage. The sensor outputs 0.5V at 0 ºC, 0.75V at 25 ºC, and 10mV per ºC. Doing an analogue-to-digital conversion on the signal line allows the local ambient temperature to be established.

The temperature sensor's signal line was connected to the LilyPad simple board's analogue pin 5 (A5) using conductive thread. A voltmeter was used to confirm that the temperature Sensor was giving a reasonable output via its S (signal) pin e.g. 0.74 volts in a room just below 25 ºC. The voltage was measured at both the temperature sensor's S pin and the LilyPad's A5 pin and the readings were consistent.

However the temperatures that were calculated by the sketch running on the LilyPad were erratic and always very high. An accurate temperature was never reported. Although this feature was not what was expected and could not be used in a production system, it was very useful during testing to have alerts being generated automatically.

[7] http://lilypadarduino.org/?p=430 viewed on 26/08/2013

[8] http://lilypadarduino.org/?p=425 viewed on 26/08/2013

4.3.3 Accelerometer

The LilyPad 3-axis Accelerometer[9] has a 2cm diameter and is based on the **ADXL335** accelerometer from Analog Devices [23]. It outputs a 0V to 3V analogue signal on each of the X, Y, and Z pins and can detect joint movement as well as inclination and vibration. It was used in this project for movement detection purposes - the LilyPad Simple board was programmed to generate no-movements alerts if the user was not moving,

4.3.4 Olimex EKG Shield

This is an EKG/EMG shield that allows Arduino boards, like the Uno, to capture Electrocardiography (and Electromyography) signals so that a user's heartbeat and pulse can be monitored when the user wears compatible electrodes – see example in **Figure 8**.

There is limited documentation[24] available for this shield but an example sketch called **ShieldEkgEmgDemo** for the Arduino and a monitoring application called **Electric Guru** are available online[10].

The example sketch populates a 17-byte structure that contains two bytes of data per channel, for a maximum of six channels. One shield equates to one channel (using 3 electrodes). It is possible to stack shields to provide all 6 channels of data – each channel connects to one of the Arduino's analogue pins A0-A5. However data was sampled for all six channels even when only one shield was being used.

The captured data seemed very noisy (see the following Electric Guru screen shot), although it does seem to follow the general pattern of a normal sinus rhythm. Some other freeware monitors (BrainBay and FreeHC) were tried too but it was difficult to know how to use these applications' filters properly.
Because of these reasons, and not knowing how to interpret the actual raw EKG data to distinguish between normal and abnormal conditions, valid EKG-related events were not generated for the prototype.

[9] http://lilypadarduino.org/?p=384 viewed on 27/08/2013

[10] https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG/ viewed on 27/08/2013

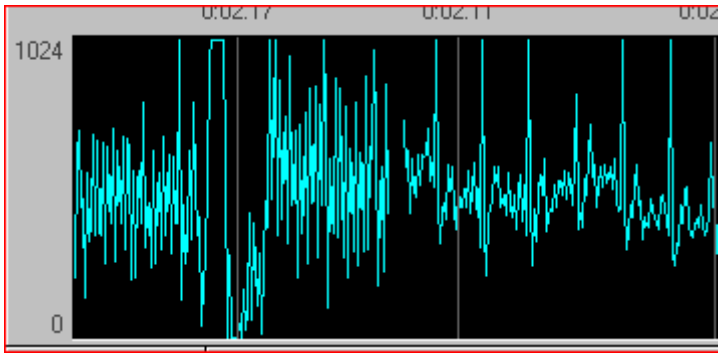

Figure 11 Poor EKG example from Electric Guru

4.4 Sewing with Conductive Thread

One practical aspect of the prototype's design concerns the sewn connections between the various LilyPad components. These connections are stitched using conductive thread which proved to be as reliable as wired connections when checked with a multimeter. However the resistance of the conductive thread was greater than wire which could cause problems on longer runs. Conductive thread can carry current for power or data signals.

Figure 12 Bobbin of Conductive Thread

There are some good online eSewing tutorials[11] and some points that were gleaned when sewing electrical connections are:

- Conductive thread is quite coarse as it is spun from stainless steel fibre. It is prone to knotting. It also tears easily if pulled too hard.
- It is essential to have excellent contact between the LilyPads' "petals" (the circular tabs) and the conductive thread. So it is best to sew through each hole several times (at least 5 or more if there is enough room).
- Doubling the thread can help increase the strength of the connection and reduces its resistance.

[11] http://lilypadarduino.org/?page_id=1256 viewed on 27/08/2013

- A large-eyed needle is very useful.
- Heavy pieces of non-stretchy fabric (e.g. denim) are easier to use than delicate material.
- For power and ground, + must be connected to + and – to –.
- RX on a LilyPad is connected to TX on an XBee breakout board and TX on a LilyPad is connected to RX on an XBee breakout board.
- Connection pathways should be designed to have as few thread crossings as possible. Although in all but the simplest configurations, it is impossible to avoid this completely. Heat shrink tubing but can be used to bridge one connection over or under another. Or if the material is thick enough, careful stitching can allow one line to cross in-between another.
- Runs should also be kept as short as possible as there is a voltage drop across the conductive material. In particular keep the power supply as close as possible to the LilyPad main board.
- Be very careful to keep separate runs sufficiently far apart so as to avoid the possibility of short circuits.

4.4.1 LilyPad Sewn Connections

The following two figures illustrate the connection runs that were originally implemented for all the LilyPad connections. Stitching for power (+) is shown in red, ground (-) in black and signal lines are shown in green.

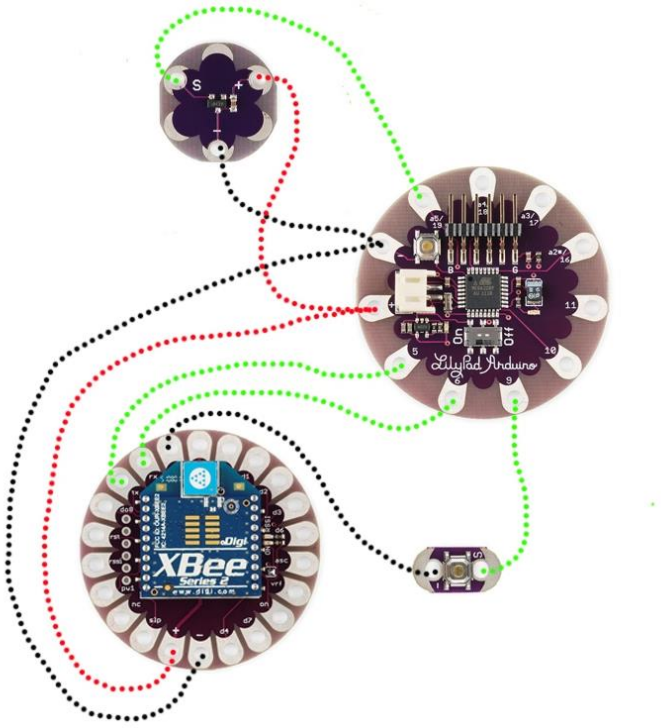

Figure 13 Stitching design for the LilyPad Simple Board

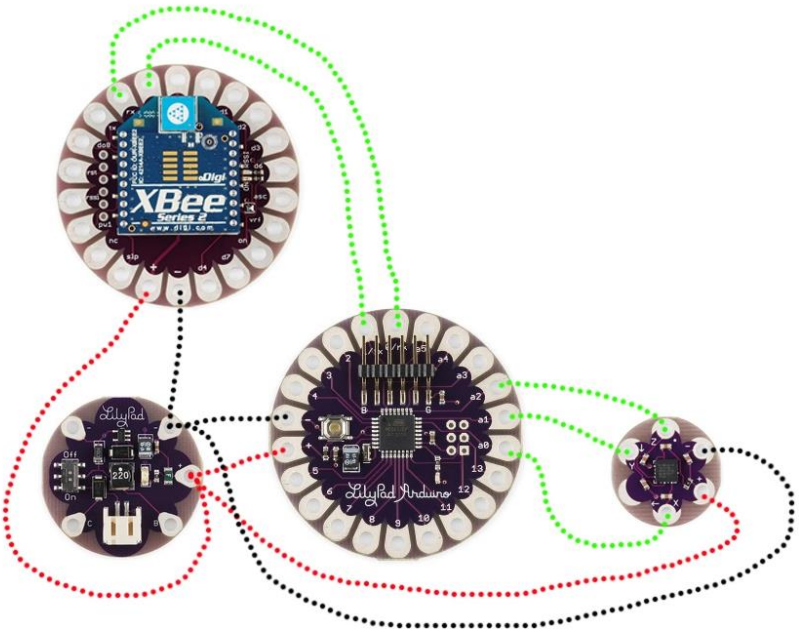

Figure 14 Stitching design for the LilyPad Main Board

4.5 Prototype Configuration

Figure 15 gives an overview of the nodes in the BASN and the gateway nodes that provide connectivity to the RMS. The LilyPad Main board was chosen as the secondary gateway as it has both ZigBee and Bluetooth wireless communication capabilities. This configuration for the prototype matches the proposed communications configuration given in **Figure 2.**

Node	Hardware	Power	Sensor	PAN Comms	External Comms	Role
1	Hub/"Smart box" (PC)	Mains	-	XBee ZB radio	Broadband Wi-Fi	XBee coordinator **Primary Gateway**
2	Arduino Lilypad Main Board	3.7V Polymer Lithium Ion Battery	Accelerometer	XBee ZB radio Bluetooth modem	-	XBee router **Secondary Gateway**
3	Arduino Uno	9V Battery	Olimex EKG	XBee ZB radio	-	XBee router
4	Arduino Lilypad Simple Board	3.7V Polymer Lithium Ion Battery	Panic Button Temperature sensor	XBee ZB radio	-	XBee router
5	Mobile device (smartphone or laptop for prototype)	Mains or battery	-	Bluetooth radio	3G Wi-Fi	Backup link between BASN & outside world

Figure 15 Prototype BASN and Gateway Configuration

4.6 ZigBee Setup

As discussed in **Chapter 3**, ZigBee was chosen as the wireless communications technology for the prototype's Body Area Sensor Network. A 4-node full-mesh topology was used with one ZigBee coordinator, three ZigBee routers and no ZigBee end nodes. There were two gateway nodes:

1) **The Primary Gateway (PGW) node**

   This node had no sensors and consisted of the ZigBee coordinator XBee radio that was plugged into the USB socket of a mains-powered computer using the XBee Explorer.

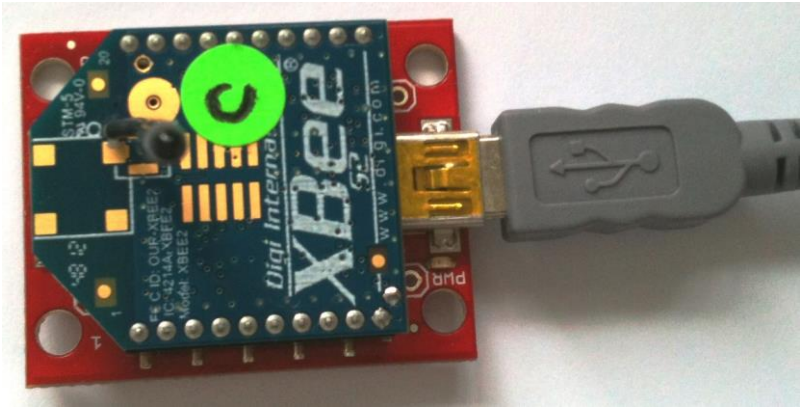

Figure 16 Primary Gateway's XBee Explorer

2) **The Secondary Gateway (SGW) node**

   This node was the LilyPad main board as it also had a Bluetooth modem to provide a backup communications link to the RMS via a local mobile device such as a smartphone or laptop.

4.6.1 AT v API mode

There are two very different ways in which an XBee radio's serial connection can be used. These modes refer to the radio's local interaction with any computer or microcontroller connected to the radio's UART and do not affect the wireless communication between XBees [25].

1) **AT**

This is the simplest way to use XBee radios. XBees that are configured to use AT commands can run in two different modes:

- **Command Mode**

  This mode is used to configure or query the XBee's settings. There are many 2-character commands to query a radio's parameters e.g. the "MY" command returns the radio's 16-bit network address. The full list of commands is given in [26].

- **Transparent Mode**

  This is the default mode for radios with AT firmware. In this mode, a radio simply passes serial data along to the destination. When data is received, there is no indication of where it was sent from and data is sent out through the serial port exactly as it was received. However any sent data is effectively sent into a black hole as the sender has no way of knowing where the data ends up.

2) API

API mode is more complicated but more powerful and reliable than AT mode as data is transmitted in a very structured frame format in a defined order. There are different frame types for sending and receiving data. Each transmission can be fully acknowledged so a sender can receive a full report regarding each of its transmissions. API mode is generally used programmatically.

The following figure illustrates the key difference between AT and API i.e. in AT mode, a receiver has no way of knowing where data has been sent from whereas in API mode, a receiver can extract all the sender's address details. This is essential if the receiver needs to return a response to the sender.

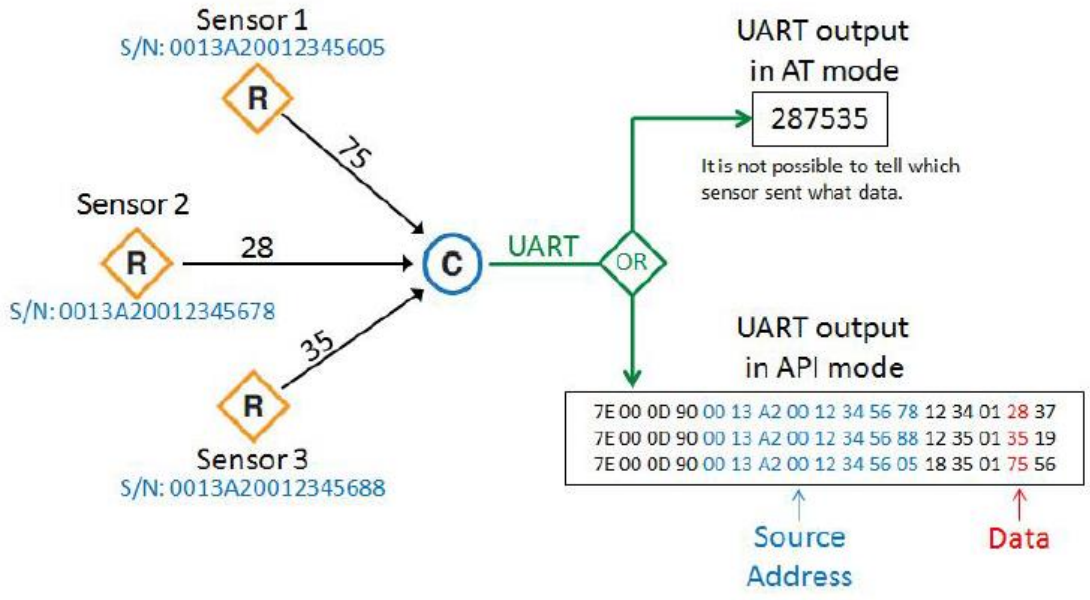

Figure 17 XBee AT and API mode [27]

Other relevant advantages of using API mode are:

- Each frame can be tracked if the sender specifies a non-zero frame identifier. This allows a sender to determine if a receiver has received each message. Like all acknowledgment systems, if the sender receives an ACK then the sender knows the receiver has definitely received the message but if a sender does not receive any ACK or NAK then it has to assume that the receiver did not receive the message even though it may have.
- The transmitting XBee radio will also attempt to resend any frames that are not acknowledged immediately.
- Frames contain a checksum for data integrity checking.
- Messages can be easily unicast, multicast or broadcast using dynamic destination addressing. It AT mode, the sender can only send to whatever destination address(es) are configured and the sender has to be reconfigured to send to different addresses.
- It is possible for any node to run commands on a remote node "over the air". This could be useful if the network needed to be reconfigured for some reason.

The prototype used XBee radios in API mode for the above reasons as AT mode would not be reliable.

4.6.2 XBee API Frames

API mode allows data to be transmitted using ZigBee in a structured and reliable manner. There is a predefined frame format [26]. All frames begin with a one-byte delimiter **0x7E** which is followed by a 2-byte length field and a one-byte frame type field. The frame type determines what data follows in the rest of the frame. All frames end in a one-byte checksum.

One very important field is the **Frame Id**. If this field is zero then no acknowledgements will be returned to the sender following a transmission. This field must be non-zero if acknowledgments are required. This is essential for reliable communications and allows a sender to track each transmitted frame at the application level. The sending application must increment the Frame Id before sending each new frame so that acknowledgements can be tracked correctly.

The following table shows the frames that were used in the prototype along with a brief explanation of how the frame can be used.

Frame Type	Name	Description
0x08	**AT command**	Allows an application to run any AT command to query or set local XBee configuration settings
0x88	**AT command response**	Response from the local XBee following a **0x08** frame
0x10	**TX Request**	Allows a sender to send data to one or more remote nodes in the network. Set Frame Id to non-zero in order to receive a full TX Status.
0x8B	**TX Status**	This frame is received by a sender following each transmission (if Frame Id was not zero). Contains a full report regarding the success or otherwise of the transmission. This is the ZigBee ACK/NAK message. If the Delivery Status field is zero then the sender knows that the intended recipients definitely received the transmission with the same Frame Id.
0x90	**RX Received**	This is the data frame received by a node as part of a unicast or broadcast. It contains all the sender's address details so a receiver knows who sent the message.

Table 11 API Frames used in the BASN

4.6.3 XBee Radio Configuration

Four Digi XBee Series 2 (**XB24-ZB**) radios were used for the prototype. ZB firmware is needed for Series 2 hardware and supports the ZigBee Pro feature set. There was one coordinator and three routers with addresses and firmware versions shown in the following table.

Radio	Antenna	Model number	High Address (32 Bits) Digi	Low Address (32 Bits)	Firmware Version
C	Whip	XB24-Z7WIT-004	0013A200	**40A26634**	**21A7**
R1	Whip	XB24-Z7WIT-004	0013A200	**408B0ECC**	**23A7**
R2	Whip	XB24-Z7WIT-004	0013A200	**40A2666D**	**23A7**
R3	PCB	XB24-Z7PIT-004	0013A200	**40AC169F**	**23A7**

Table 12 XBee addresses and firmware versions

The Series 2 XBee's microcontroller inside the radio cannot support AT and API mode and different firmware versions must be loaded onto the radio depending on which mode is used to communicate over the local serial port. There are also different firmware versions depending on

whether a radio is a coordinator, router or end node. Firmware is easily updated using **X-CTU[28]** but it means that a radio's role has to be pre-decided and cannot be changed at runtime i.e. a router cannot take on a coordinator's role if the original coordinator fails.

The four radios were configured using **X-CTU** to have the same Personal Area Network (PAN) Identifier "12345" and to use API mode with **AP=2** (escaped control bytes [26]). This setting was required in order to use xbee-arduino and xbee-api libraries. Appropriate Node Identifier were also configured for each radio using the **NI** parameter.

The following screen shot shows the four BASN nodes running. The coordinator has assigned 16-bit short addresses to the other 3 nodes. These addresses are not configurable but can be obtained programmatically at runtime in order to use fast addressing.

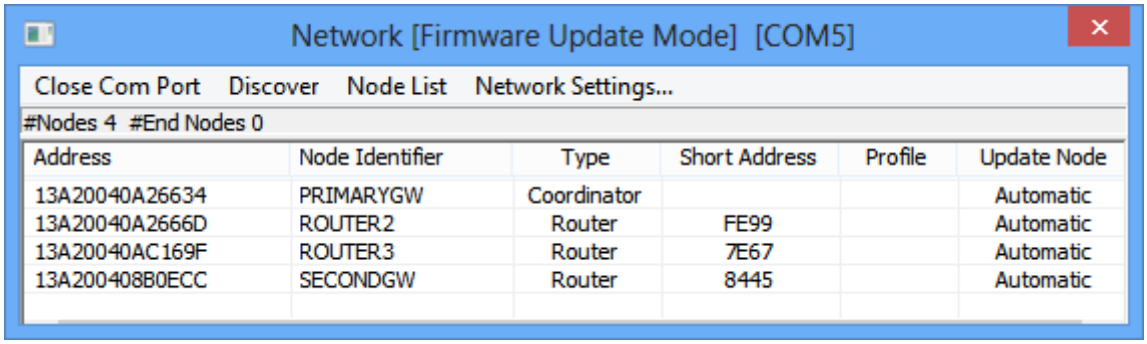

Figure 18 X-CTU screen shot showing the four BASN nodes

4.7 Bluetooth Setup

Bluetooth was used to connect the BASN's secondary gateway node to a mobile device that had both Bluetooth and external communication capabilities. Most smartphones now have a Bluetooth component. However for the purposes of the prototype, a laptop was used. The reason for the Bluetooth connection is to provide an alternative backup link from the BASN to the outside world (via the intermediate mobile/laptop). Such intermediate mobile devices generally have multiple communication choices e.g. 3G and Wi-Fi as well as Bluetooth while ZigBee is generally not available on smartphones.

4.7.1 Bluetooth Modem for LilyPad Main Board

As shown in the following figure, a **Bluetooth Mate Silver** Class 2 Bluetooth® Radio Modem was modified to incorporate a FTDI 6-pin header to attach directly the LilyPad main board.

Figure 19 Bluetooth Mate Silver with 6-pin FTDI Header

A MacBook Pro with a built-in Bluetooth module from Broadcom was used for the prototype. This is compatible with Bluetooth devices supporting the **Serial Port Profile (SPP)**. The laptop could be substituted with any other portable device that has both Bluetooth and external communication (e.g. 3G) capabilities.

4.7.2 Setting up the Bluetooth Connection

The Bluetooth connection between the LilyPad main board and the MacBook Pro laptop was set up as follows:

1) Power the LilyPad by battery and connect the Silver mate modem via FTDI[12]. The modem's red Stat LED should start blinking.
2) On the Mac, turn Bluetooth ON and tick the Discoverable option.
3) Open Bluetooth Preferences. The Mac should find the LilyPad's **RN42-DFB1** modem. Highlight this RN42 device and click the '+' button to open the Bluetooth Setup Assistant.
4) Highlight the RN42 again and click the 'Continue' button. Then click on the 'Passcode Options…' button and select the 'Use a specific passcode' option. Replace the default pass code '0000' with '**1234**' and click 'OK'.
5) In the Bluetooth Setup Assistant window, click the 'Continue' button and the Mac should pair successfully with the RN42 using the correct pass code. Messages should be displayed saying:

    - Pairing was completed successfully
    - A computer serial port was created

[12] The soldered metal connection points for the FTDI header pins on the underside of the modem was covered with insulating tape as it sits right above conductive thread stitching on the LilyPad and had potential to short circuit.

Quit out of the Bluetooth Setup Assistant. Note that these steps will *not* have to be repeated again – once the serial port is created, it will continue to exist on the Mac even if the modem is not on.

6) In the Bluetooth Preferences window, highlight the **RN42-DFB1**. It should now be listed as **Paired** and **Configured** but *not* **Connected** (circular icon is **red**). Click the cog icon (beside the + and – buttons) and select 'Edit Serial Ports…' from the dropdown list.
7) Take note of the path for the serial port e.g. **/dev/tty.RN42-DFB1-SPP** and click 'Apply'.

Figure 20 Bluetooth serial port on the Mac

Note that quite often this step fails and the user has to turn Bluetooth OFF on the Mac and start again from the beginning.

8) The connected icon should now be **green** beside the **RN42-DFB1** in the Bluetooth Preferences window. Note that steps 6 and 7 will have to be repeated manually every time a new connection is required.
9) Depending on what sketch the LilyPad is running, serial output may be seen on the Mac by running the **Terminal** utility and typing in the **screen** command followed by the path for the Bluetooth serial port that was noted earlier e.g.

   $ screen /dev/tty.RN42-DFB1-SPP
10) The modem's green Connect LED should also be on when there is a connection with the laptop (and the red Stat LED is off).

4.7.3 Configuring the Bluetooth Connection

There are a number of ways to enter "command mode" to check and/or change the Bluetooth modem's configuration settings. For example, use the same screen command as before and enter three dollar characters "**$$$**" (without the quotes). However by default the modem will go into "fast data mode" **60 seconds** after it powers up and it will no longer be possible to enter command mode then. Another complication is that local echo is off by default, to turn it on type '+' followed by carriage return. There is a full list of modem commands in [29].

In command mode, the modem's red 'Stat' LED will blink much faster (10 times per second). Some useful commands are: 'd' and 'e' to display basic and extended settings. To exit command mode, type three minus characters "---" followed by carriage return.

A straightforward way to change the modem's configuration is to use a serial port terminal application such as CoolTerm[13]. Remember to set the Baud rate to **115200** for the serial port and turn "Local Echo" on for the Terminal. Once the port is connected, it is very easy to enter commands and see both the command text and the modem's responses. The following example shows some basic "GET Commands" along with two "SET commands" (changes take effect after the module is rebooted):

- **ST,180** Sets the configuration timer to 3 minutes to give more time to enter command mode
- **SU,96** Changes the modem's baud rate to **9,600 bps**

CoolTerm (on the Mac) can also be used to display output from whatever sketch is running on the LilyPad main board. Remember to set the baud rate to whatever the modem is using i.e. 9600 bps if it was successfully changed using the SU command above.

[13] CoolTerm is a freeware application that was written by Roger Meier and was available for download from http://freeware.the-meiers.org/ on 01/08/2013.

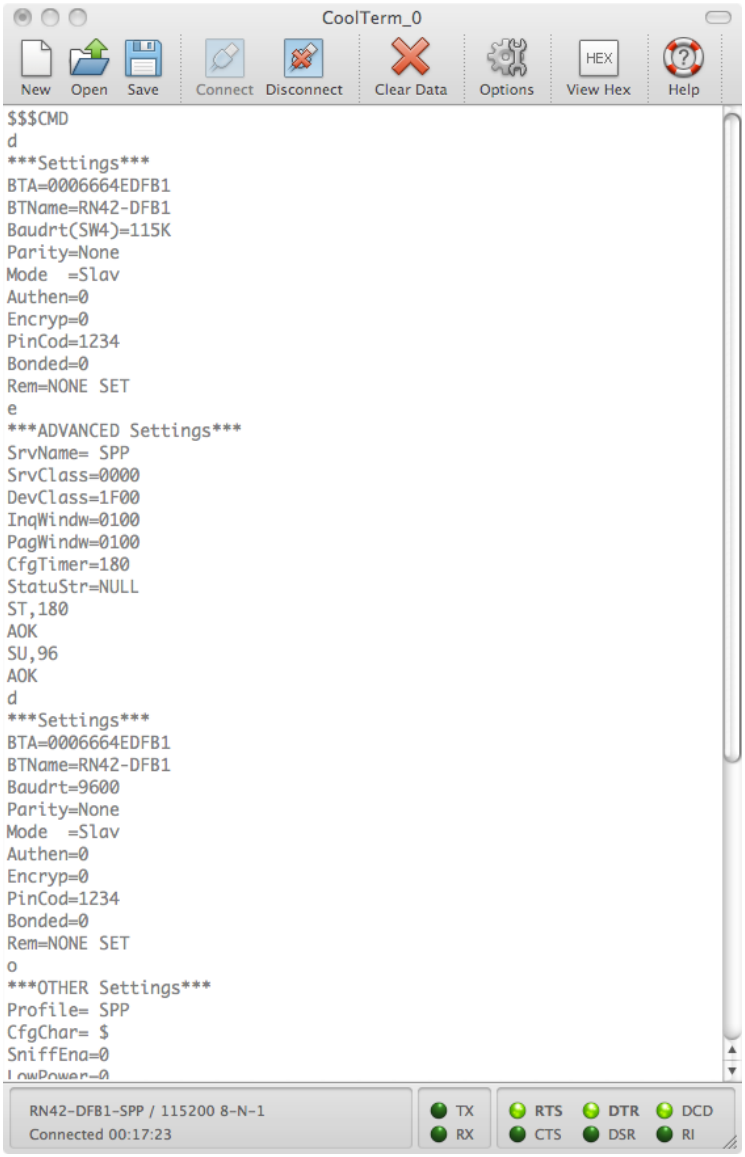

Figure 21 Bluetooth command mode example using CoolTerm

4.8 Serial Communication

4.8.1 XBee Serial Port

An XBee radio uses its antenna for ZigBee wireless communications. It also has a UART or serial port and can use its pin 3 (DIN) and pin 2 (DOUT) for TTL serial communication with another device (e.g. a microcontroller) that has a compatible UART [26]. This is how an Arduino board for example, which communicates with the XBee serially, gains the capability to communicate using radio waves to remote devices. The XBee acts as a gateway for the Arduino and extends it onto ZigBee networks [25]. A sketch running on an Arduino can then use a serial library to send and receive ZigBee messages via the Arduino's serial pins.

4.8.2 Arduino Hardware and Software Serial Ports

Arduino boards have at least one built-in serial port (also known as a UART or USART). The Uno and LilyPads have an **ATmega16U2** USB-to-TTL serial convertor connected to digital pins 0 (RX) and 1 (TX) which is referred to as the *hardware* serial port. Any computer that is connected via USB to these pins can communicate serially with the Arduino to exchange TTL serial data e.g. to upload new sketches or to view output from sketches via the Arduino IDE's serial monitor (or any other terminal interface). The UART allows the microcontroller to receive serial communication even while working on other tasks, as long as there room in the 64-byte serial buffer[14].

Some complications when implementing the prototype were:

- The LilyPad main board needs two serial connections – one to communicate via Bluetooth and the other to communicate via ZigBee.
- Although the LilyPad simple board has a UART, its digital pins 0 and 1 are not broken out to tabs. Instead serial communication using the hardware serial port is via the FTDI header. However the XBee breakout board only has sewn connections to the LilyPad simple board's tabs.

The solution in these cases is to use a *software* serial port which emulates additional serial ports. A Software Serial library effectively turns an arbitrary pair of digital pins into a new serial port although performance is not as good as a real hardware serial port. Another drawback is that if multiple software serial ports are used then only one can receive data at a time.

4.9 Software Components

This section describes each of the software components that was developed for the prototype. All software is available on the enclosed CD. There is software for the following:

- Arduino software was needed for the 3 Arduino BASN nodes – a total of 3 individual sketches for the LilyPads and the Uno. These components share a Body Area Network (BAN) library that was developed for the prototype. The sketch for the LilyPad main board (the secondary gateway) is not yet complete.

[14] http://arduino.cc/en/Serial/Available viewed on 26/08/2013

- Processing sketch for the Primary gateway on the laptop (can also run on a Windows PC) which includes an RMS client to interface with the dummy RMS
- Processing sketch for a Bluetooth application (for the laptop) to act as a bridge between the Secondary gateway and the dummy RMS. This sketch is not yet complete.
- Dummy RMS – java code to accept alert information via HTTP and display details on a local web page.

4.9.1 BAN Library

The three Arduino sketches, share a Body Area Network (BAN) library. There are two source files, **BAN.h** and **BAN.cpp**. These contain definitions for the event message and heartbeat structures etc. There are a number of common functions that the Arduino BASN sketches share as well e.g. a function to find the best gateway to use when generating an alert.

4.9.2 xbee-arduino library

This is the Arduino library for communicating with **Series 2** XBees in **API mode**[15]. A beta **version 0.4** was used for the prototype as support for using the XBees on a Software Serial port was required. There is an online Wiki and a short developer's guide. Most online examples concern the XBee's data sampling I/O functionality, which is not relevant to this prototype. The XBee class library is reasonably straightforward to follow and all the required API frames (listed in **Section 4.6.2**) are supported. The library proved to be very reliable. Two key points to note about using this library are:

- The XBee radios must be configured to have the **AP** parameter set to 2
- The XBee object's software serial port must be initialised in a slightly unusual way as follows:

```
XBee xbee = XBee();
altSerial.begin(9600);
xbee.setSerial(altSerial);
```

The **AltSoftSerial** library[16] had to be used for the LilyPad main board's sketch as it requires simultaneous ZigBee and Bluetooth serial communications. This is not supported by the standard Arduino Software Serial library.

[15] http://code.google.com/p/xbee-arduino/ viewed on 29/08/2013

[16] http://www.pjrc.com/teensy/td_libs_AltSoftSerial.html viewed on 29/08/2013

4.9.3 xbee-api Library

This is the Java equivalent of the xbee-arduino library for communicating with **Series 2** XBees in **API mode**. **Version 0.9**[17] of the xbee-api library was used in the Processing Java environment for the Primary Gateway. There is an online Wiki and developer's guide for the library.

Note that the XBee class structure for this Java library is completely different to the XBee class for the Arduino library. This library also proved to be very reliable although the class structure is quite complex.

4.9.4 LilyPad Simple Sketch

The Arduino sketch for the LilyPad simple board is called **LilyPadSimple.ino**. This code is a suitable basis for a BASN node that is not a gateway i.e. ZigBee is the only required wireless communication.

A software serial library is also required for the LilyPad Simple board as its TX and RX UART pins are not broken out. The standard Arduino **SoftwareSerial** library[18] was used.

This sketch also contains support for the panic button and temperature sensor.

4.9.5 LilyPad Main Sketch

The Arduino sketch for the LilyPad main board is called **LilyPadMain.ino**. This code is a suitable basis for a BASN node that is a gateway i.e. both Bluetooth and ZigBee wireless communications are supported. As mentioned in **Section 4.9.2**, the **AltSoftSerial** library had to be used in this sketch for ZigBee communications.
As discussed in **Section 4.2.2** this sketch is complex as it runs on a gateway node and must handle alerts from other nodes. This sketch also contains support for the Accelerometer.

This sketch was not completely finished as the LilyPad main board broke in the final stages of development. See **Section 5.3** for further details.

[17] http://code.google.com/p/xbee-api/ viewed on 29/08/2013

[18] http://arduino.cc/en/Reference/SoftwareSerial viewed on 29/08/2013

4.9.6 Uno Sketch

The Arduino sketch for the Uno is called **Uno.ino**. Support for the EKG sensor is based on the example **ShieldEkgEmgDemo** which is discussed in **Section 4.3.4.** This sketch requires support for ZigBee wireless communications that is similar to the LilyPad simple board's sketch except that different TX and RX pins are used.

4.9.7 BANPrimaryGateway Sketch

The software for the primary gateway node was developed in Java in the Processing[19] environment. The sketch is called **BANPrimaryGateway.pde** and it can run in both Windows and OS X environments. The **xbee-api** library was used for ZigBee communications and another freeware library called **G4P** (GUI for Processing)[20] was used to display a simple window with scrollbars containing alert information.

This sketch contains support for broadcasting "EA" (External Alerting) services, receiving alerts via ZigBee for other BASN nodes, receiving broadcast from other gateway nodes and interfacing with the dummy RMS application via HTTP.

4.9.8 Bluetooth Sketch

This sketch is called **BlueToothApp.pde** and is also implemented using Processing. It acts like a pipe – passing messages received via Bluetooth from the BASN's secondary gateway to the dummy RMS application and vice versa. This sketch was almost completely implemented.

This sketch runs on the Mac and requires a Bluetooth serial port to be set up (see **Section 4.7**) and connected to the secondary gateway node. The port is easily opened as follows:

```
String portName = "/dev/tty.RN42-DFB1-SPP";
btPort = new Serial(this, portName, 9600);
```

Messages are then exchanged over Bluetooth with the LilyPad Main board using **btPort.read()** and **btPort.write()**. Messages are also exchanged with the dummy RMS over HTTP in the same way as the **BANPrimaryGateway**.

[19] http://www.processing.org/ last viewed on 29/08/2013

[20] http://www.lagers.org.uk/g4p/ last viewed on 29/08/2013

4.9.9 Remote Monitoring System (RMS)

A very basic dummy RMS was implemented to run on the same computer (a MacBook Pro laptop) as the **BANPrimaryGateway** and **BlueToothApp** sketches. The Java code and configuration files for the RMS resides in a directory called **RemoteMonitoringStation** on the Mac.

A simple web page http://localhost:8080/RemoteMonitoringStation/ was created to display alerts that were received from BASN nodes via gateways using HTTP. The first version of the dummy RMS always returned an acknowledgement immediately.

Apache Maven v3.1.0[21] was used to build the RMS web page on the Mac as follows:

$ **echo $PATH**
/Applications/apache-maven-3.1.0/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/X11/bin
$ **mvn --version**
Apache Maven 3.1.0 (893ca28a1da9d5f51ac03827af98bb730128f9f2; 2013-06-28 03:15:32+0100)
Maven home: /Applications/apache-maven-3.1.0
Java version: 1.6.0_51, vendor: Apple Inc.
Java home: /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home
Default locale: en_US, platform encoding: MacRoman
OS name: "mac os x", version: "10.6.8", arch: "x86_64", family: "mac"
$ **cd RemoteMonitoringStation**
$ **mvn jetty:run**
[INFO] Scanning for projects...
[INFO]
[INFO] --
[INFO] Building RemoteMonitoringStation 1.0-SNAPSHOT
[INFO] --

An example screen shot of the dummy RMS page http://localhost:8080/RemoteMonitoringStation/ using the Safari browser is shown in the following figure. The sequence numbers used in the original source node's alert are displayed.

[21] http://maven.apache.org/ last viewed on 26/08/2013

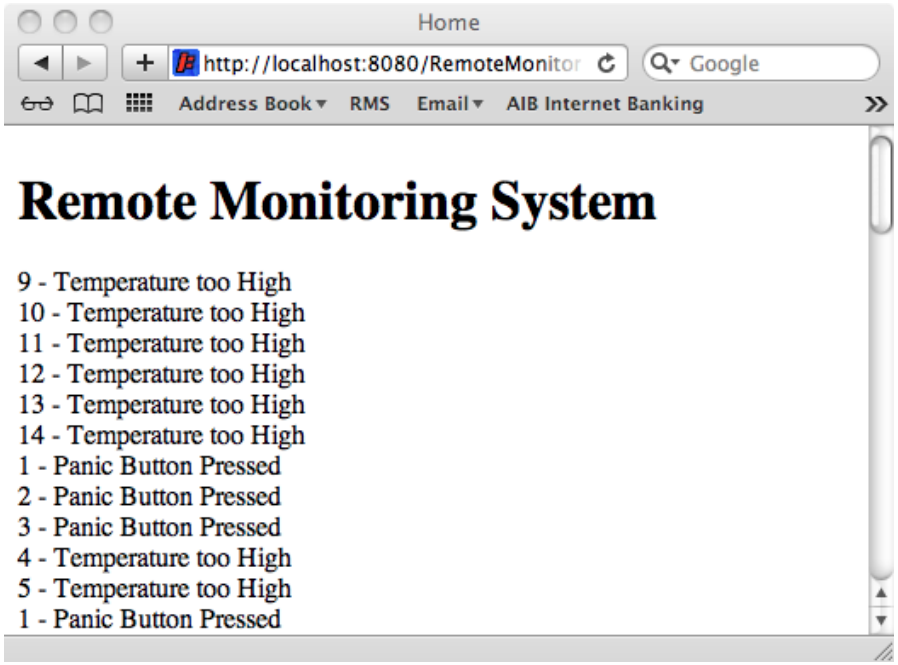

Figure 22 Example screen shot of dummy RMS alert page

4.10 Chapter Summary

This chapter described the physical implementation and configuration of the prototype in addition to giving an overview of the software components that had to be developed.

The prototype hardware components forming the BASN, are based on the Arduino platform and consist of an Arduino Uno, LilyPad Main Board and LilyPad Simple Board that are connected to various sensors and communicate wirelessly via ZigBee. The prototype uses EKG and temperature sensors, a panic button and an accelerometer. The LilyPad components are connected using conductive thread and are stitched to fabric. The primary gateway is provided by a mains powered computer with an XBee radio connected via USB. The LilyPad mainboard acts as a secondary gateway through a Bluetooth modem.
The ZigBee mesh network is configured to run in API mode, which provides reliable facilities for frame tracking, acknowledgements, data integrity checking and automatic resending of unacknowledged or negatively acknowledged frames.

The software developed for the prototype makes use of a number of external libraries and includes elements to run on the BASN nodes and gateways as well as a simulated Remote Monitoring System which displays alerts and sends acknowledgements back to the originator.

Chapter 5 Evaluation

This chapter describes some tests and experiments to help verify that the prototype was implemented as intended i.e. that the protocol that was designed in **Chapter 3** helps to improve reliable communication of alerts between the Body Area Network and the Remote Monitoring Station. The fundamental overriding principle behind all tests is that alerts must not be lost, regardless of what communications failures occur or what conditions the tests are run under. However a distinction should be drawn between what can be reasonably expected from a prototype (that has no real-time clock or permanent logging/storage facilities etc.) and what would be expected from a fully functional production system.

It is also desirable to determine what the side-effects of ensuring reliable communication are – particularly in terms of the following two unavoidable consequences:

1) The volume of duplicate alerts that can be generated under different failure scenarios, along with confirmation of how many duplicate alerts are identifiable and qualify for filtering at the Remote Monitoring System.

2) The additional time delay between an original alert being generated by a source node and the Remote Monitoring System being notified. It would be essential to discern what these overheads are under various conditions e.g. normal operation, different combinations of primary and secondary gateway failures and also under conditions of stress where multiple events occur at the same time – to simulate a faulty sensor generating incorrect or high volumes of alerts etc.

Statistics gathered from running tests to cover the scenarios outlined above would be needed in order to ascertain that a fully implemented system is safe and fit for production.

Because of the somewhat unquantifiable nature of reliability, it is expected that test results are more qualitative than quantitative. However it is nevertheless important to demonstrate that the prototype operates as intended under various stress conditions, particularly if there is interference, from household appliances for example, or if a number of events and/or failures occur contemporaneously. Reliability in a distributed system can be much more difficult to ensure as more than one failure can occur at the same time in different parts of the system.

The sections in this chapter are organised as follows:

Section	Contents
5.1	Specification of experiments in terms of tests and expected results. Tests are grouped into suites covering specific areas and scenarios.
5.2	Issues that came to light during the analysis and design phases of the project
5.3	Issues during implementation that contributed to the delayed completion of the prototype. Unfortunately the implementation of the Secondary Gateway was not completed.
5,4	Chapter summary

Table 13 Sections in Chapter 5

5.1 Experiments

This section contains the specification of all experiments in terms of tests and expected results. Tests are grouped into suites. A preamble explains some of the rationale behind the tests. Note that the following abbreviations are used in the test specifications:

ACK ZigBee Acknowledgement i.e. a Transmit Status frame (0x8B) with a successful (zero) delivery status

Env The test environment – the preconditions that must be in place before running a specific test

LPM The LilyPad Main board

LPS The LilyPad Simple board

NAK ZigBee Negative Acknowledgement i.e. a Transmit Status frame (0x8B) with an unsuccessful (non-zero) delivery status

PGW The primary gateway – refers to the laptop with the ZigBee coordinator plugged in and the **BANPrimaryGateway** sketch running.

RMS Remote Monitoring System – refers to the local web page http://localhost:8080/RemoteMonitoringStation/ running on the same laptop as the PGW. Unless otherwise stated, the RMS should return its ACKs immediately.

RMS ACK Acknowledgement from the RMS – indicates that an alert was received from a source node via a gateway.

SGW The secondary gateway – refers to the LilyPad main board with its ZigBee radio and Bluetooth modem both working

ZB ZigBee

5.1.1 Test Suite A – Normal Operation

The tests in this suite prove that the system works under normal conditions. There are no deliberate failures and no unusually high data rates. These are the fundamental control tests to verify that the system functions as expected when all alerts are routed via the Primary Gateway to the Remote Monitoring System.

It would be useful to run each test in this suite a number of times to get more accurate values for timings regarding how quickly acknowledgements are received at the source node. It would be interesting to plot the time intervals between a source node raising an alert and receiving an RMS ACK. These timings would then represent the best case results and would be useful for comparison purposes.

Test ID: [A-1] LPS Alert **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately**		
Step	**Test Description**	**Expected Results**
1.	Cause the LPS to generate a low temperature alert	The alert is routed via the PGW to the RMS. The Uno receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs[22]. The relevant alert is displayed on the RMS web page.
2.	Cause the LPS to generate a high temperature alert	As in step 1.

[22] Note that the source node should actually receive the RMS ACK almost instantaneously as the RMS is running locally

3.	Cause the LPS to generate a panic button alert	As in step 1.

Test ID: [A-2] Uno Alert **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately**		
Step	**Test Description**	**Expected Results**
1.	Cause the Uno to generate an EKG alert	The alert is routed via the PGW to the RMS. The Uno receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page.

Test ID: [A-3] LPM Alert **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately**		
Step	**Test Description**	**Expected Results**
1.	Cause the LPM to generate a no movement alert	The alert is routed via the PGW to the RMS. The LPM receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page.

5.1.2 Suite B – PGW Failure/Connectivity Issues

This suite simulates comms failures to the primary gateway from the BASN and from the primary gateway to the RMS. The aim is to ensure that messages are routed and acknowledged via the secondary gateway. Switching back to the primary gateway, following its recovery, is also exercised to ensure that messages are no longer routed via the secondary gateway.

Test ID: [B-1] PGW XBee comms failure – alerts from a single node Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately		
Step	**Test Description**	**Expected Results**
1.	Cause the LPS to generate a panic button alert	The alert is routed via the PGW to the RMS. The LPS receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page.
2.	Unplug the XBee Explorer from the laptop so that the PGW is no longer part of the ZigBee network	
3.	Cause the LPS to generate a high or low temperature alert	The LPS receives a ZB NAK as the PGW is unreachable. All unacknowledged alerts (including this one) are routed to the RMS via the SGW (with the same sequence number if already sent via the PGW). The LPS receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page. SGW generates alert indicating that the PGW is not reachable. This alert is displayed on the RMS web page.
4.	Plug the XBee Explorer with the coordinator back into the laptop and restart BANPrimaryGateway	Nodes should all receive broadcasts from the PGW indicating that it is operational again i.e. “EA10P” messages
5.	Cause the LPS to generate a panic button alert	The alert is routed via the PGW to the RMS. The LPS receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page.

Test ID: [B-2] PGW XBee comms failure – resending of unacknowledged alerts **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to delay responding to acknowledgements for 45 seconds**		
Step	**Test Description**	**Expected Results**
1.	Cause the LPS to generate a high or low temperature alert	The alert is routed via the PGW to the RMS. The LPS receives a ZB ACK within 1 second but RMS ACK is not received immediately. The relevant alert is displayed on the RMS web page.
2.	Before the RMS ACK is received, unplug the XBee Explorer from the laptop so that the PGW is no longer part of the ZigBee network	
3.	Cause the LPS to generate a panic button alert.	The LPS receives a ZB NAK as the PGW is unreachable. All unacknowledged alerts (including this one and the alert sent in step 1) are routed to the RMS via the SGW (with the same sequence number if already sent via the PGW). The LPS receives a ZB ACK within 1 second and the RMS ACK in 45-60 secs. The relevant alert is displayed on the RMS web page. SGW generates alert indicating that the PGW is not reachable. This alert is displayed on the RMS web page.
4.	Plug the XBee Explorer with the coordinator back into the laptop and restart BANPrimaryGateway	Nodes should all receive broadcasts from the PGW indicating that it is operational again i.e. “EA10P” messages
5.	Cause the LPS to generate another panic button alert	The alert is routed via the PGW to the RMS. The LPS receives a ZB ACK within 1 second and the RMS ACK in 45-60 secs. The relevant alert is displayed on the RMS web page.

Test ID: [B-3] PGW XBee comms failure – alerts from multiple nodes **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately**		
Step	**Test Description**	**Expected Results**
1.	Unplug the XBee Explorer from the laptop so that the PGW is no longer part of the ZigBee network	
2.	Cause the Uno to generate an EKG alert	The Uno receives a ZB NAK as the PGW is unreachable. The alert is routed to the RMS via the SGW. The Uno receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page. SGW generates alert indicating that the PGW is not reachable. This alert is displayed on the RMS web page.
3.	Cause the LPM to generate a no movement alert	The LPM receives a ZB NAK as the PGW is unreachable. The alert is routed to the RMS via the SGW. The LPM receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page.
4.	Plug the XBee Explorer with the coordinator back into the laptop and restart BANPrimaryGateway	Nodes should all receive broadcasts from the PGW indicating that it is operational again i.e. “EA10P” messages
5.	Cause the LPM and UNO to generate more alerts	The alerts are routed via the PGW to the RMS. The LPM/Uno receive a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alerts are displayed on the RMS web page.

Test ID: [B-4] PGW to RMS comms failure – message sent by PGW to RMS but no ACK received back from RMS. Env: Fully operational system – both gateways working, all batteries charged, RMS acknowledgements disabled initially		
Step	**Test Description**	**Expected Results**
1.	Cause the Uno to generate an EKG alert	The alert is routed via the PGW to the RMS. The Uno receives a ZB ACK within 1 second. The relevant alert is displayed on the RMS web page but there is no RMS ACK.
2.	Before 60 seconds have passed re-enable RMS ACKs	After 60 secs, the alert is routed (using the same sequence no) to the RMS via the SGW. The Uno receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page. SGW generates alert indicating that there is an issue with PGW to RMS connectivity.
3.	Cause the LPS to generate another panic button alert	The alert is routed via the PGW to the RMS. The LPS receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page.

5.1.3 Suite C – SGW Failure/Connectivity Issues

This suite simulates comms failures to the secondary gateway from the BASN to ensure that messages are routed and acknowledged via the primary gateway.

Test ID: [C-1] SGW unavailable- LPS Alert Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately		
Step	**Test Description**	**Expected Results**
1.	Switch off the LPM so that the SGW is no longer part of the ZigBee network	

2.	Cause the LPS to generate a panic button alert	The alert is routed via the PGW to the RMS. The LPS receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page.

Test ID: [C-2] SGW XBee comms failure - Uno Alert **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately**		
Step	**Test Description**	**Expected Results**
1.	Unplug the XBee module from the LPM so that the SGW is no longer part of the ZigBee network	
2.	Cause the Uno to generate an EKG alert	The alert is routed via the PGW to the RMS. The Uno receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page.

5.1.4 Suite D – Dual Failure of PGW and SGW Connectivity

This suite simulates comms failures to both the primary and the secondary gateway from the BASN nodes ensuring that messages are resent via the first available path on restoration of the primary gateway or secondary gateway.

Test ID: [D-1] PGW XBee comms failure, SGW unavailable – recovery of PGW **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately**		
Step	**Test Description**	**Expected Results**
1.	Unplug the XBee Explorer from the laptop so that the PGW is no longer part of the ZigBee network	
2.	Switch off the LPM so that the SGW is no longer part of the ZigBee network	

3.	Cause the LPS to generate a high or low temperature alert	The LPS receives a ZB NAK as the PGW is unreachable. The LPS receives a ZB NAK as the SGW is unreachable.
4.	Plug the XBee Explorer with the coordinator back into the laptop and restart BANPrimaryGateway	Nodes should all receive broadcasts from the PGW indicating that it is operational again i.e. "EA10P" messages Immediately after handling of the PGW broadcast the alert is resent via the PGW. The LPS receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page.

Test ID: [D-2] PGW XBee comms failure, SGW unavailable – recovery of SGW then PGW **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately**		
Step	**Test Description**	**Expected Results**
1.	Unplug the XBee Explorer from the laptop so that the PGW is no longer part of the ZigBee network	
2.	Switch off the LPM so that the SGW is no longer part of the ZigBee network	
3.	Cause the LPS to generate a panic button alert	The LPS receives a ZB NAK as the PGW is unreachable. The LPS receives a ZB NAK as the SGW is unreachable.

4.	Switch on the LPM	Nodes should all receive broadcasts from the SGW indicating that it is operational again i.e. "EA10S" messages Immediately after handling of the SGW broadcast the alert is resent via the SGW. The LPS receives a ZB ACK within 1 second and the RMS ACK in less than 30 secs. The relevant alert is displayed on the RMS web page. SGW generates one alert indicating that PGW is not reachable.
5.	Cause the LPS to generate a high or low temperature alert.	The alert is routed to the RMS via the SGW. The LPS receives a ZB ACK within 1 second and the RMS ACK within 30 secs. The relevant alert is displayed on the RMS web page.
6.	Plug the XBee Explorer with the coordinator back into the laptop and restart BANPrimaryGateway	Nodes should all receive broadcasts from the PGW indicating that it is operational again i.e. "EA10P" messages
7.	Cause the LPS to generate a panic button alert.	The alert is routed to the RMS via the PGW. The LPS receives a ZB ACK within 1 second and the RMS ACK within 30 secs. The relevant alert is displayed on the RMS web page.

5.1.5 Suite E – Failure of Node to connect to PGW or SGW

This suite simulates temporary comms failures on one sensor node so that it is unable to reach either the primary or the secondary gateway. It tests that messages are resent via the first available path on restoration of connectivity.

Test ID: [E-1] LPS Comms failure **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately**		
Step	**Test Description**	**Expected Results**
1.	Take the LPS out of range of the rest of the XBee network. XBee operating range can be up to 90m.	

2.	Cause the LPS to generate a panic button alert	The LPS receives a ZB NAK as the PGW is unreachable. The LPS receives a ZB NAK as the SGW is unreachable.
3.	Cause the LPS to generate a high or low temperature alert	The LPS receives a ZB NAK as the PGW is unreachable. The LPS receives a ZB NAK as the SGW is unreachable.
4.	Bring the LPS back into range of the XBee network.	The LPS should receive broadcasts from PGW and the SGW within 30 secs indicating that the gateways are reachable – e.g. "EA10P" and "EA10S" messages. Both alerts are resent via the first gateway to have sent its heartbeat message. The LPS receives ZB ACK for each within 1 second and the RMS ACK for each in less than 30 secs. The relevant alert is displayed on the RMS web page.
5.	Cause the LPS to generate a panic button alert.	The alert is routed to the RMS via the PGW. The LPS receives a ZB ACK within 1 second and the RMS ACK within 30 secs. The relevant alert is displayed on the RMS web page.

Test ID: [E-2] Uno Comms failure – PGW Comms failure **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately**		
Step	**Test Description**	**Expected Results**
1.	Unplug the XBee Explorer from the laptop so that the PGW is no longer part of the ZigBee network	
2.	Take the Uno out of range of the rest of the XBee network. XBee operating range can be up to 90m.	

3.	Cause the Uno to generate an EKG alert	The Uno receives a ZB NAK as the PGW is unreachable. The Uno receives a ZB NAK as the SGW is unreachable.
4.	Bring the Uno back into range of the XBee network.	The Uno should receive a broadcast from the SGW within 60 secs indicating that it is reachable – e.g. "EA10S" message. The Uno resends the alert via the SGW and receives a ZB ACK within 1 second and the RMS ACK for each in less than 30 secs. The relevant alert is displayed on the RMS web page.
5.	Plug the XBee Explorer with the coordinator back into the laptop and restart BANPrimaryGateway	Nodes should all receive broadcasts from the PGW indicating that it is operational again i.e. "EA10P" messages
6.	Cause the LPS to generate a panic button alert.	The alert is routed to the RMS via the PGW. The LPS receives a ZB ACK within 1 second and the RMS ACK within 30 secs. The relevant alert is displayed on the RMS web page.

Test ID: [E-3] Uno Comms failure – SGW Comms failure **Env: Fully operational system – both gateways working, all batteries charged, RMS configured to respond with acknowledgements immediately**		
Step	**Test Description**	**Expected Results**
1.	Switch off the LPM so that the SGW is no longer part of the ZigBee network	
2.	Take the Uno out of range of the rest of the XBee network. XBee operating range can be up to 90m.	
3.	Cause the Uno to generate an EKG alert	The Uno receives a ZB NAK as the PGW is unreachable. The Uno receives a ZB NAK as the SGW is unreachable.

4.	Bring the Uno back into range of the XBee network.	The Uno should receive a broadcast from the PGW within 30 secs indicating that it is reachable – e.g. "EA10P" message. The Uno resends the alert via the PGW and receives a ZB ACK within 1 second and the RMS ACK for each in less than 30 secs. The relevant alert is displayed on the RMS web page.
5.	Switch on the LPM	Nodes should all receive broadcasts from the SGW indicating that it is operational again i.e. "EA10PS" message.
6.	Cause the LPS to generate a panic button alert.	The alert is routed to the RMS via the PGW. The LPS receives a ZB ACK within 1 second and the RMS ACK within 30 secs. The relevant alert is displayed on the RMS web page.

5.1.6 Suite F – Stress Tests

This suite exercises the system under a heavy load by using a test sketch on BASN nodes[23] that generates multiple test alerts at a high frequency. The purpose of this test is to quantify the load at which failure (i.e. loss of messages or unacceptable delays) occurs and the time to failure under these loads. The test should be repeated with different message rates and durations as outlined in the following table.

Test ID	Alert Rate (msg/second)	Duration (seconds)
a.	1	60
b.	1	300
c.	5	20
d.	5	120
e.	10	10
f.	10	60
g.	10	300

Table 14 Stress conditions for testing

[23] The Uno might be more resilient in running stress tests as its processor runs at 16MHz – the LilyPads' processor is slower at 8MHz – but in all cases the source nodes have the same amount of RAM (32KB).

Test ID: [F-1] Stress Tests Env: Fully operational system – both gateways working, all batteries charged. Test sketch running on the Uno and the LPM that is generating alerts at an abnormal rate.		
Step	**Test Description**	**Checks**
1.	Cause the test sketch on the Uno to generate one unique alert (i.e. no duplicates) at the rate specified in the table above for the specified time.	
2.	While the above sketch is running Cause the LPS to generate a panic button alert.	
3.	On completion examine the logs to identify any failure.	Check for message loss and identify any alerts that were not displayed on the RMS web page. Check for the routing of messages to see that all went via the PGW. Check for missed acknowledgements resulting in alerts being resent. Identify the point at which any buffers became full.
4.	Repeat steps 1, 2 and, 3 this time generating alerts from the LPM rather than the Uno.	
5.	Repeat steps 1, 2 and, 3 this time generating alerts from both the Uno and the LPM simultaneously.	

5.2 Known Issues

This section describes some issues that were uncovered during the analysis and design phases of the project.

5.2.1 ZigBee Coordinator Failure

As discussed in **Section 2.5**, the ZigBee coordinator can be a single point of failure.

5.2.2 Potential Alert Loss if Source and Gateway Nodes Fail

The protocol designed and outlined in **Chapter 3** is deficient in the following scenario.

Time	Source Node	Gateway Chosen by Source Node - GW
t	Generates alert	
$t + \delta_1$		If GW is working then alert is received; Otherwise, alert is not delivered to GW
$t + \delta_2$	Source node fails (in a state where it does not know whether the alert has been received by the GW or not).	
$t + \delta_3$		If GW did receive the alert but it fails *before* it could send the alert on to the Remote Monitoring System (RMS), then the alert is lost.

Table 15 Potential Alert Loss Scenario

Therefore, if a source node fails after it has generated an alert and either (a) the chosen gateway is not available or (b) the chosen gateway fails before sending the alert on to the RMS, then the alert is lost. A more sophisticated protocol would keep track of each source node and continually check whether source nodes are operational; the current protocol really only keeps track of gateway nodes.

Note that if the failed source node recovers, it has no record of the previous alerts it had raised. A logging or permanent storage facility would be needed in order for a recovered node to find out if it had any outstanding partially processed alerts prior to its failure. The prototype could potentially be extended to use the Arduino EEPROM library to write details of ongoing alerts to non-volatile memory. There would then be an additional performance overhead associated with each alert as each EEPROM write takes 3.3 milliseconds to complete[24].

One suggestion for future work in **Section 6.2.1** describes a subscription protocol where any node (not just gateways) may subscribe to another node's services. This would facilitate a "buddy system" where one node could monitor another node (or nodes). This mechanism would be useful for quickly detecting node failures and faults e.g. if a monitored node's battery power was running low then its buddy may be able to generate alerts on its behalf. A node could also send its alerts to

[24] http://arduino.cc/en/Reference/EEPROMWrite retrieved on 21/08/2013

its buddy so that a buddy node could process any outstanding alerts on behalf of the source node, should the source node fail.

5.2.3 Interference

The prototype uses XBee ZB RF modules which operate within the ZigBee protocol. ZigBee defines a network layer above the 802.15.4 MAC/PHY layers. The PHY (physical) layer defines physical operational parameters including the operating frequency which is the 2.4 GHz ISM (Industrial, Scientific and Medical) band [10]. The XBee radios will be operating on one of 16 channels in this 2.4 GHz band – the coordinator selects a channel when it starts the network.

The prototype also uses a Bluetooth Mate modem with an RN-42 class 2 Bluetooth radio that operates according to the 802.15.1 standard. Bluetooth also operates at 2.4 GHz [30][25]. Although Bluetooth can hop over 79 channels in the ISM band to adapt to interference, it is not ideal to have the primary and secondary wireless channels both operating at the same frequency.

A further complication is that other devices, including 802.11 and common household appliances such as microwave ovens, can also use the 2.4 GHz band. Therefore there is potential for both the prototype's primary and secondary wireless channels to be blocked at the same time when the user is in close proximity to a source of interference. If an alert was generated while neither wireless channel was operational, then the alert would be delayed until the cause of the interference ceased.

For a production system, the primary and secondary wireless communication options would have to be carefully selected. In order to avoid single points of failure, it would be better to not have both wireless links operating within the same frequency band or at the very least to have the capability to configure both wireless links to operate on different channels with frequencies that were as far away from each other as possible. One option to consider would be to use a ZigBee radio with a different chipset that used 866/902 MHz instead of 2.4 GHz.

Details of properties of potential frequency bands for Body Area Networks is discussed in [14].

[25] Frequency range: 2,402 ~ 2,480 MHz

5.3 Implementation Issues

This section details some technical problems that occurred when the prototype was being implemented. Investigation of each issue was very time-consuming and was a key contributing factor to the prototype's secondary gateway not being fully implemented and none of the experiments specified in section 5.1 being executed. The Arduino development became extremely tedious as the project progressed and reached the stage where each new release had to be carried out in very small increments as the boards became more and more likely to hang/crash as more memory was being used, even though the compiled sketches were less than half the size of RAM. An extra "Serial.print()" statement or an extra element in an array could cause these types of problem and without basic debugging tools it is very difficult to isolate the offending statement.

The following list is not exhaustive but gives a flavour of the implementation issues that were encountered.

1) **Uploading compiled code from the Arduino IDE to the Lilypads or Uno**

   Arduino development was done mostly in a Windows 8 environment. There were frequent problems when trying to upload code e.g. sometimes the COM port would not be listed or the sketch would refuse to upload with "out of sync" errors. Sometimes plugging the USB cable in and out or resetting the board or rebooting the PC etc. would help. There never seemed to be a pattern and it often took more than 30 minutes to just upload a new version of a sketch.

2) **The Olimex shield and electrodes do not generate clean EKG data**

   Early on in the project, considerable time was wasted trying to extract clean EKG data from the Olimex using monitoring applications such as ElecGuru, BrainBay and FreeHC. In all cases, the resulting EKG graphs looked very noisy and would need the correct filters to be applied to see good EKG data.

3) **Incorrect temperature readings**

   As mentioned in **Section 4.3.2** the temperature readings were always too high and not consistent. The issue is with setting or obtaining the correct analogue reference voltage being used on the LilyPad and therefore converting the result from analogRead (A5)[26] to the correct voltage. The analogue reference voltage can be set using analogReference()[27] which should set

[26] http://arduino.cc/en/Reference/AnalogRead last viewed on 26/08/2013

[27] http://arduino.cc/en/Reference/AnalogReference last viewed on 26/08/2013

the conversion factor which needs to be applied when obtaining the voltage from analogRead as follows:

```
int         sensorValue = analogRead(A5);
float       voltage= sensorValue * (FACTOR / 1024.0);
```

Where FACTOR is 5.0 or 3.3 if an analogReference of DEFAULT was used, or is 1.1 if analogReference of 1.1 was used. Using FACTOR of 5.0, 3.3 or 1.1 as above does not give the correct value for voltage shown by the voltmeter. Furthermore switching between battery and USB power causes different results from the analogRead so obviously affects the analogReference voltage used by the board.

A number of days were spent investigating this issue and trying out alternative code that uses Band Gap Reference on the board to identify the reference voltage. This code was found in several variants on various fora with users claiming it worked successfully for LilyPads but it never produced good temperature readings for the prototype.

4) **Accelerometer incorrectly reporting changes in X, Y and Z when not moving**

A very simple algorithm was implemented to check whether the user was moving or not. The total of the absolute differences in each of the X, Y and Z values is calculated at regular intervals. A total of zero was expected if the user was not moving. However totals in the range 5 - 35 approximately were seen when the accelerometer was lying flat.

5) **BlueSMiRF Silver Bluetooth modem[28] not compatible with LilyPads**

This modem was ordered for the prototype originally. Time was wasted trying to get it to work when it has the wrong pin layout for the LilyPad main board. The Bluetooth Mate Silver modem[29] is designed specifically to be used with LilyPads but was received quite late on in the project and delayed the discovery of subsequent issues.

6) **Bluetooth and ZigBee incorrectly sharing the same serial Tx/Rx pins**

The Bluetooth Mate worked first time with the LilyPad main board but caused a number of other issues e.g. ZigBee messages from the LilyPad main board appeared on a Bluetooth serial monitor on the laptop. This happened because the XBee breakout board was stitched to the

[28] https://www.sparkfun.com/products/10269 last viewed on 26/08/2013

[29] https://www.sparkfun.com/products/10393 last viewed on 26/08/2013

LilyPad main board's TX and RX pins which were being incorrectly shared with the Bluetooth modem as it was also connecting to the same TX and RX pins via the FTDI header.

After that happened, although not ideal for performance reasons, I decided to use a SoftwareSerial port for the XBee and keep Bluetooth on the hardware serial port (as I couldn't easily change the FTDI header). So I re-stitched the XBee to LilyPad TX and RX connections and changed the sketch to use the standard Arduino SoftwareSerial library[30].

However this change resulted in one-way ZigBee communication.

7) SoftwareSerial library prevents simultaneous ZB and BT comms

In the week before the demonstration, a different software serial library called AltSoftSerial[31] was used as it supports simultaneous data flows. Unfortunately it only works with LilyPad pins 8, 9 and 10. A test using temporary wiring worked so the XBee and LilyPad RX and TX connections had to be re-stitched again.

8) Short circuit between Tx and Rx on the LilyPad main board

When I re-stitched the RX and TX connections I did not notice that there was potential for a short circuit between the TX and RX lines at the XBee breakout board caused by a loose thread. This caused incoming ZigBee frames to be corrupted and transmit frame headers were received intermittently. It took a good few hours to figure out what was causing that problem. Once the offending piece of conductive thread was fixed, I finally had simultaneous 2-way ZigBee and Bluetooth communications on the LilyPad main board.

9) **LilyPad main board hanging or crashing**

When development of the Secondary Gateway on the LilyPad main board was finally back on track, very late on in the project, the LilyPad started to hang and/or crash frequently when new versions of the sketch were uploaded. Minor changes (e.g. increasing an array by one element) could cause these problems. This completely slowed down my progress as I had to start making very small incremental software releases.

10) Unable to upload any code to the Lilypad main board – avrdude errors

Then the LilyPad main board finally stopped working and I could no longer upload even the simplest "blinking led" sketch to the board. Different avrdude errors were returned from the Arduino IDE from OS X and Windows but the end result was the same.

[30] http://arduino.cc/en/Reference/SoftwareSerial last viewed on 26/08/2013

[31] http://www.pjrc.com/teensy/td_libs_AltSoftSerial.html last viewed on 26/08/2013

Binary sketch size: 1,108 bytes (of a 30,720 byte maximum)
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync(): not in sync: resp=0x00

Much time was spent trying to upload code to the LilyPad without success and it is likely that the board is broken.

5.4 Chapter summary

This chapter describes detailed experiments aimed at verifying the correct behaviour of the prototype system with regard to its requirements and design, as laid out in **Chapter 3**. It also outlines known issues with the design and problems encountered during implementation.

The specified suites of tests cover both normal operation and failure scenarios including failure (and subsequent recovery) of the primary gateway, failure of the secondary gateway and dual failure of both gateways. Stress testing is also included.

The execution of these experiments in full was not possible due to a number of implementation issues, which are outlined in this chapter as well. These delays prevented completion of the entire prototype. However I am confident that if implementation of the prototype was completed then the specified experiments would thorough exercise its functionality and performance.

Chapter 6 Conclusions

This chapter is the final chapter in the dissertation and contains sections to review the goals that were set out in the Introduction and to outline suggestions for future work. The sections in this chapter are organised as follows:

Section	Contents
5.1	Project achievements with respect to the original goals as set out in **Chapter 1**.
6.2	Recommendations for work that could be undertaken in the future and suggestions for interesting areas of further research regarding ZigBee.
6.3	Final Remarks

Table 16 Sections in Chapter 6

6.1 Achievements

The following table summarises the goals that were set for the project in **Section 1.3** along with an assessment of achievements.

	Objectives	Achievements
1)	To research the State of the Art in areas associated with Body Area Sensor Networks and eHealth in the context of remote monitoring to promote assisted living for the elderly	**Chapter 2** contains overviews of research papers covering Remote Monitoring Systems, Technologies and Standards for Wireless Body Area Networks, Challenges facing WBANs, Middleware and ZigBee-related reliability issues.

	Objectives	Achievements
2)	To investigate a suitable communications topology and technology for a Body Area Sensor Network (BASN)	**Section 3.2** gives an overview of some potential topologies for the BASN and **Section 2.3** covers wireless options
3)	To propose a communications configuration to support the reliable communication of alerts, that are generated by a BASN, to a Remote Monitoring System (RMS)	The configuration proposed in **Section 3.3** meets this objective by providing redundancy through secondary channels and avoiding single points of failure.
4)	To propose a communications protocol to help improve the reliable communication of alerts from a BASN to a RMS	The protocol outlined in **Section 3.4** makes use of the redundancy provided by the proposed communications configuration. Acknowledgements are used to ensure that messages not flagged as received by the RMS are reset via the next best available route. The design is scalable and provides for temporary situations where no transport mechanism for alerts is available from the BASN to the RMS, by resending the alert as soon as a link is signalled as recovered.
5)	To implement a prototype that uses the proposed protocol to communicate alerts from a BASN to a dummy RMS. This includes: • Selecting and configuring hardware components such as microcontrollers and sensors as well as communications equipment for a BASN. • Designing a system architecture and implementing software components to run on the selected equipment	While not totally complete, the prototype meets most aspects of these objectives. The Arduino boards (Lilypads and Uno) were shown to be viable for use in such a system and the ZigBee network forming the backbone of the BAN proved to be reliable. While some sensor data was not interpreted correctly, the data was consistent indicating that the sensors could be used to trigger accurate alerts. The software components that were completed were seen to be behaving reliably and gave confidence that the design was valid and that further development would have been successful.

	Objectives	Achievements
6)	To specify and execute experiments to evaluate the prototype	The experiments specified in **Chapter 5** are comprehensive and would permit a complete prototype to be evaluated well, particularly with respect to the areas of handling of comms failures and recovery under a range of different conditions.
7)	To identify any issues or areas for future work	Analysis during the dissertation led to some suggestions for another area of research. **Section 6.2** covers suggestions for Future work, including completion of the prototype.

Table 17 Achievements

In addition to the project achievements detailed above I also gained:

- Experience of working on a project almost entirely on my own – my professional software experience to date has involved working as part of a development team.
- Technical experience using a good variety of hardware components and development tools such as the Arduino IDE and Processing.
- Hands-on experience setting up and using a wireless sensor network.
- Additional communications knowledge regarding Bluetooth and ZigBee

6.2 Future Work

This section suggests areas that could be worked on in the future. In addition to completing the implementation of the prototype, the analysis and implementation phases of the project revealed an interesting area that is worthy of further research in its own right.

6.2.1 Prototype Evaluation

With a new LilyPad main board, the code for the secondary gateway could be completed. Alternatively, the Uno could be adapted to use the Bluetooth modem and could then become the secondary gateway for the prototype.

Once a secondary gateway is in place, the full suite of experiments that are specified in **Section 5.1** could be run. The tests cover a wide range of failure scenarios and so would give a qualitative view of whether the proposed protocol and configuration were reliable.

6.2.2 Middleware Subscription Mechanism

The proposed protocol can be seen as a first step towards a full subscription mechanism. For example, a subscription service could be used to allow a node to register to receive messages regarding relevant events. Additionally, historic data could be included in the event message payload to provide additional context for an event e.g. the last 30 seconds of EKG data could be very useful when a serious EKG related event occurs.

The basic ZigBee messaging is implemented using XBee radios in API mode. This functionality could be extended to provide a Middleware layer that included functionality for peer monitoring and supported cooperation between nodes to enhance reliability. For example, peer sensor nodes could collaborate with each other for the purpose of (a) reaching consensus regarding the user's health status, with the aim of reducing the occurrence of false alarms and missed events and (b) monitoring the status of peers to ensure their continued operation and, where necessary, taking over the role of a faulty peer or alerting a user or remote monitoring service to the occurrence of a fault.

6.2.3 ZigBee Coordinator Failure

A separate research area in its own right concerns ZigBee coordinator failure and recovery. Every ZigBee network must have exactly one coordinator which is by definition a single point of failure. The impact of the coordinator's failure depends on the network's topology. For example, in a star configuration the coordinator is the central node and its failure would have a catastrophic impact on the rest of the network as none of the slave nodes would be able to communicate with each other. In a mesh network, the coordinator's failure would have less of an impact if the coordinator is not needed for routing as other nodes should be able to route around the coordinator.

Section 2.5.1 includes details of an interesting paper from NASA [16] that covers ZigBee fault tolerance testing but there is an outstanding area regarding the setup of parallel coordinators. Following an email communication on 18/07/2013 with the paper's author (Richard Alena), it would be very interesting to investigate how a standby ZigBee coordinator, on its own PAN, could take over in the event that the primary coordinator in a different PAN failed. The ZigBee network watchdog or middleware that provided node-monitoring services are potential ways of ensuring nodes from one PAN migrated to a standby if their coordinator failed.

6.3 Final Remarks

This dissertation has demonstrated that with readily available, inexpensive equipment and relatively little development effort, it is possible to address many of the concerns relating to communications reliability in remote monitoring solutions for Body Area Sensor Networks.

There can be little doubt that in years to come, the area of eHealth will become pervasive, particularly with regard to the safe and reliable remote monitoring of patients at home. This inevitability will be driven by the increasing cost of providing traditional health care to an ageing population and facilitated by advances in technology at reduced cost.

Appendix A Abbreviations and Acronyms

ADL	Activities of Daily Living
AODV	Ad-hoc On-demand Distance Vector - routing algorithm used by XBee nodes. Routes are discovered only when needed and nodes on inactive paths do not maintaining routing tables.
BAN	Body Area Network
BASN	Body Area Sensor Network
BSN	Body Sensor Network
Bio-sensor	A sensor that can measure some biological/physiological characteristic e.g. a temperature sensors that measures body temperature.
Bluetooth	A wireless "cable replacement" technology for PANs operating in the unlicensed ISM band at 2.4 GHz and defined by the IEEE 802.15.1 standard [9].
EA	In this dissertation, EA refers to an External Alerting service that allows nodes in the BAN to communicate with the Remote Monitoring System. The EA service is provided by gateway nodes.
eHealth	The use of Information and Communication Technologies (ICT) for healthcare[32]
FTDI	Future Technology Devices International Ltd. Provide products, such as ICs and 6-wire cables, to allow various devices to interface with USB. The LilyPads have a 6-pin FTDI header.
ISM	Industrial, scientific and medical band – frequency bands that can be used license-free e.g. 2.4 GHz worldwide and other frequencies such as 900 MHz.

[32] Based on description from http://www.who.int/topics/ehealth/en/ on 27/08/2013

MAC	Medium Access Control – service provided by the link layer in a communications protocol stack. Controls access to a shared physical medium when there are multiple senders/receivers.
mHealth	Mobile eHealth – same as eHealth but with greater emphasis on user mobility and the use of mobile devices such as smartphones.
PAN	Personal Area Network – short range wireless network. The IEEE 802.15 standards cover Wireless PANs
PHY	The physical layer in a communications protocol stack – the lowest layer that deals with the actual physical transmission of each bit of data over a specific (wired or wireless) medium. The data link layer sits immediately above the physical layer.
RF	Radio Frequency
Sensor	Device which detects or measures a physical property and records, indicates, or otherwise responds to it[33].
Serial Communication	A method (adhering to a standard such as RS-232) of transferring data sequentially, one bit at a time, as opposed to parallel communications which allows multiple bits to be transferred simultaneously. As most computers no longer have serial ports, a USB serial adapter can be used to connect an XBee radio, for example, to a PC.
Telemonitoring	When patients use basic medical equipment to monitor themselves and results are usually sent via telephone to the healthcare provider.
TTL	Transistor-Transistor Logic. Method of serial communication using 0 volts (for 0) and 5 volts (for 1) - how signals were represented in one of the first implementations of digital logic.
UART	Universal Asynchronous Receiver/Transmitter. UARTs communicate serially i.e. transmit one bit of data at a time at a specified data rate e.g. 115,200 bps.
XBee	Digi International® Inc. produce a range of RF modules including XBee ZB modules that support wireless connectivity to devices in ZigBee mesh networks.
X-CTU	Configuration and Test Utility Windows application provided by Digi to manage the firmware and configuration of their RF products including XBees [28].
ZigBee	The ZigBee Alliance sponsors the specification of a wireless network standard [11] for low-powered devices that run over PHY and MAC layers that are defined by the IEEE 802.15.4 standard [10].

[33] Definition taken from http://oxforddictionaries.com/definition/english/sensor?q=sensor on 26/08/2013

Bibliography

1. D. Tutu, et al., “The bigger picture for e-health,” *The Bulletin*, vol. 90, no. 5, 2012, pp. 330-331.

2. J.D. Piette, et al., “Impacts of e-health on the outcomes of care in low-and middle-income countries: where do we go from here?,” *Bulletin of the World Health Organization*, vol. 90, no. 5, 2012, pp. 365-372.

3. A.S. Tanenbaum and M. Van Steen, *Distributed systems, Principles and Paradigms*, Prentice Hall, 2002, p. 686.

4. M.A. Hanson, et al., “Body Area Sensor Networks: Challenges And Opportunities,” Series 42, IEEE Computer Society, 2009, pp. 58-65.

5. S. Abbate, et al., “Monitoring of human movements for fall detection and activities recognition in elderly care using wireless sensor network: a survey” Intech, 2010, pp. 20.

6. S. Abbate, et al., “Recognition of false alarms in fall detection systems,” *Proc. Consumer Communications and Networking Conference (CCNC)*, IEEE, 2011, pp. 23-28.

7. H. Alemdar and C. Ersoy, “Wireless sensor networks for healthcare: A survey,” *Computer Networks*, vol. 54, no. 15, 2010, pp. 2688-2710.

8. S. Abbate, et al., “MIMS: A Minimally Invasive Monitoring Sensor Platform,” *Sensors Journal, IEEE*, vol. 12, no. 3, 2012, pp. 677-684.

9. IEEE Standards Association, “802.15.1-2005 - IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements. - Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs),” pp. 0_1 - 580.

10. IEEE Standards Association, “IEEE Standard for Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs),”, 2011, pp. 294.

11. ZigBee Standards Organisation, “ZigBee Specification” , 2007, pp. 576.

12. ZigBee Alliance, “ZigBee Wireless Sensor Applications for Health, Wellness and Fitness,” 03/2009; https://docs.zigbee.org/zigbee-docs/dcn/09-4962.pdf. Accessed on 29/08/2013. Accessed on 29/08/2013.

13. I.S. Association, “IEEE Standard for Local and metropolitan area networks - Part 15.6: Wireless Body Area Networks ” *Book IEEE Standard for Local and metropolitan area networks*, 2012, pp. 257.

14. M. Patel and J. Wang, “Applications, Challenges, And Prospective In Emerging Body Area Networking Technologies,” *Wireless Communications, IEEE*, vol. 17, no. 1, 2010, pp. 80-88.

15. A. Boulis, et al., “Challenges in Body Area Networks for Healthcare: the MAC,” *Communications Magazine, IEEE*, vol. 50, no. 5, 2012, pp. 100-106.

16. R. Alena, et al., “Fault tolerance in ZigBee wireless sensor networks,” *Proc. Aerospace Conference, 2011 IEEE*, IEEE, 2011, pp. 1-15.

17. W. Wang, et al., “Integrating Sensors with the Cloud Using Dynamic Proxies,” *Proc. Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, IEEE, 2012, pp. 1466-1471.

18. A.B. Waluyo, et al., “Design and evaluation of lightweight middleware for personal wireless body area network,” *Personal and Ubiquitous Computing*, vol. 13, no. 7, 2009, pp. 509-525.

19. J.K. Peckol, *Embedded Systems, A Contemporary Design Tool*, John Wiley & Sons, Inc., 2008, p. 810.

20. L. Galluccio, et al., “Challenges and Implications of Using Ultrasonic Communications in Intra-body Area Networks,” *Proc. Wireless On-demand Network Systems and Services (WONS), 2012 9th Annual Conference on*, IEEE, 2012, pp. 182-189.

21. B.D. Franklin, et al., “Failure mode and effects analysis: too little for too much?,” *BMJ Quality & Safety*, vol. 21, no. 7, 2012, pp. 607-611.

22. Microchip, “MCP9700/9700A/MCP9701/9701A Datasheet,” 2007; https://www.sparkfun.com/datasheets/DevTools/LilyPad/MCP9700.pdf. Accessed on 29/08/2013.

23. Aanalog Devices, “ADXL335 Small, Low Power, 3-Axis ±3 g Accelerometer,” 2010; http://www.analog.com/static/imported-files/data_sheets/ADXL335.pdf. Accessed on 29/08/2013.

24. Olimex, “SHIELD-EKG-EMG bio-feedback shied USER’S MANUAL,” 04/2013; https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG/resources/SHIELD-EKG-EMG.pdf. Accessed on 29/08/2013.

25. R. Faludi, *Building Wireless Sensor Networks*, O'Reilly, 2010, p. 300.

26. Digi, “XBee®/XBee-PRO® ZB RF Modules,” 08/05/2013; http://ftp1.digi.com/support/documentation/90000976_P.pdf. Accessed on 29/08/2013.

27. Digi, “XBee® ZigBee RF Module Development Kit Getting Started Guide,” 25/07/2012; http://ftp1.digi.com/support/documentation/90002168_A.pdf. Accessed on 29/08/2013.

28. Digi, “X-CTU Configuration & Test Utility Software User’s Guide,” 20/08/2008; http://ftp1.digi.com/support/documentation/90001003_A.pdf. Accessed on 29/08/2013.

29. Roving Networks, “Bluetooth Data Module Command Reference & Advanced Information User’s Guide,” 26/03/2013; http://www.rovingnetworks.com/products/RN42. Accessed on 29/08/2013.

30. Roving Networks, “RN42/RN42N Class 2 Bluetooth Module Data Sheet ” 11/04/2013; http://www.rovingnetworks.com/products/RN42. Accessed on 29/08/2013.