TotTemp: A wireless data reporting system with environmental integration for infant health monitoring.

by

Adam Taylor, B.A. ,B.A.I.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

September 2011

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.

Adam Taylor

August 28, 2011

Permission to Lend and/or Copy

Acknowledgments

I would like to thank my supervisor Ciarán Mc Goldrick for all his support and suggestions throughout this project. I would also like to extend my thanks to everyone who helped to proof read this report.

Adam Taylor

University of Dublin, Trinity College

September 2011

TotTemp: A wireless data reporting system with environmental integration for infant health monitoring.

Adam Taylor, M.Sc.
University of Dublin, Trinity College, 2011

Supervisor: Ciarán Mc Goldrick

The ubiquitous vision of computing proposed by Mark Weiser [1], was that computers would seamlessly integrate into the environment. This idea leads to environments augmented with a wide variety of sensors. The remote temperature monitoring system proposed herein leverages wireless sensing and networking functionalities to supplement its on-board capabilities for providing real time monitoring of infant temperature. The data from embedded environmental sensors is fused with that from the near contact units at a SunSPOT to produce a data stream that can be routed to an output device. In the prototype, the near contact unit is connected via Bluetooth to the SunSPOT. The Bluetooth channel allows for easy interconnection to the plethora of sensors that may be available in a likely deployment scenario. To reduce the workload given to the near contact unit or base station, data analysis is done on the output device, in this example

an Android Phone. This moves a large quantity of the processing to a more easily charged device, thereby extending the lifespan of the monitoring units. The TotTemp prototype system aims to provide a flexible, cost-effective system capable of real time monitoring, alerting of potential medical conditions and reporting of gathered sensor data.

By providing continuous temperature monitoring of a sleeping child coupled with the alerting capabilities of a phone it will be possible to provide early warning of infections as well as maladies associated with extremes in temperature such as hyperthermia or fever. It can also provide warnings about potentially dangerous environmental actuators such as direct sunlight before they have a chance to affect the incident body directly. If the environment allows, the device could take remedial action itself by interfacing with the climate control system or other such system.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The use of wireless sensors for health monitoring and reporting is becoming more and more common as the cost of sensors falls. This work proposes a system to monitor the health of a sleeping infant using a variety of techniques. There are several medical conditions that can be predicted by continuous monitoring. Temperature, for example, can be an indicator for many infections and certain types of cancer. Non-contact monitoring of temperature is useful in several other fields such as environmental monitoring.

1.2 Report Layout

The following is a brief overview of the structure of this report with an outline of the content in each chapter.

1.2.1 Background Technologies

A description of the technologies used in the project and some details that were found to be significant about them will be provided in this chapter. In this part of the report the technologies' relevant capabilities will be explained. It has one section for each of the main technologies used.

1.2.2 Literature Review

A review of related work in the fields of wireless sensing and non-contact health monitoring is conducted. The chapter opens with a description of wireless sensor networks and then progresses on to discussing approaches to scheduling the monitoring of physical parameters. The latter parts of this chapter deal with the networking implications of continuous data generation and gives a brief overview of some of the medical conditions the system would possibly detect.

1.2.3 Design

This chapter will explain the design choices made and the motivation behind them in the prototype system. It will also provide some design considerations for the actual system. This chapter is split into three sections, one dealing with the architecture of the whole system, the others each focusing on one of the different devices involved.

1.2.4 Implementation

The focus of this chapter is on the actual implementation of the project. Unlike the Design chapter it is segmented by technology, as each technology had to be configured individually before they could be used. This section of the report will also outline some of the problems encountered and the solutions that were found.

1.3 Medical Introduction

As most of the rest the report will talk about human body temperature, this section will provide a brief overview of the range of values that can be generated and the rate at which it can change. Some of the conditions that temperature can indicate will also be discussed.

There are several factors that affect body temperature; the external temperature; levels of exertion; age and time of day. There are also natural individual variations. For all these reasons, a single temperature reading is not particularly helpful in providing information on general health unless it is well out side the typical range. The best way to determine if a single reading is out of range is to have an established base line for that person. Table 1.1 illustrates the temperature ranges for several conditions according to [6] and [7]. The range for Hyperthermia or Fever is the same: only the mechanism by which the temperature is raised differs, with Hyperthermia being externally driven and fever internally so.

Range	Temperature Value
Normal	36.5 - 37.5°
Hyperpyrexia	40° $<$ Reading
Hyperthermia or Fever	37.5 - 38.3°
Hypothermia	Reading $<$ 35°

Table 1.1: Temperature Ranges of Conditions

It is also important to consider the location that the reading is taken. Any internal temperature sensors will yield a more accurate representation of core temperature than external methods. An internal reading will usually only be 0.5° less than the core temperature with the oral temperature being 1° less on average. The difference between external temperature and core temperature is much more sensitive to location than internal readings. In a healthy individual at room temperature the reading obtained via a hand can vary form 25.2° to 33.8° [8]. This is because the peripheral regions of the human body have lower rates of blood flow and less insulation than the core. The forehead is a significantly more stable source of data than the hands, as is the axillary temperature (under the arm). Both provide similar readings to an oral temperature. The sources of error, though, are typically more difficult to eliminate. With an oral thermometer hot or cold drinks influence the reading for approximately 5 minutes after drinking which is easily eliminated but with external reading movement dislodging the thermometer can not be allowed for by any method other than careful monitoring. Readings taken along the temporal artery (see figure 1.1) can have accuracy equivalent to that of internal readings. There are two temporal arteries, the anterior and posterior (see figure 1.1). They both run very close to the surface of the skin and come directly from the maxillary artery, which is one of the branches of the carotid artery, so the blood still has the same temperature as the body's core. They run parallel to one another, roughly vertical from the line of the bottom of the ear up behind the temple. The only draw back with taking readings on the temporal arteries are that it is somewhat difficult to locate and only affects a small area of skin so reading here are especially sensitive to movement.

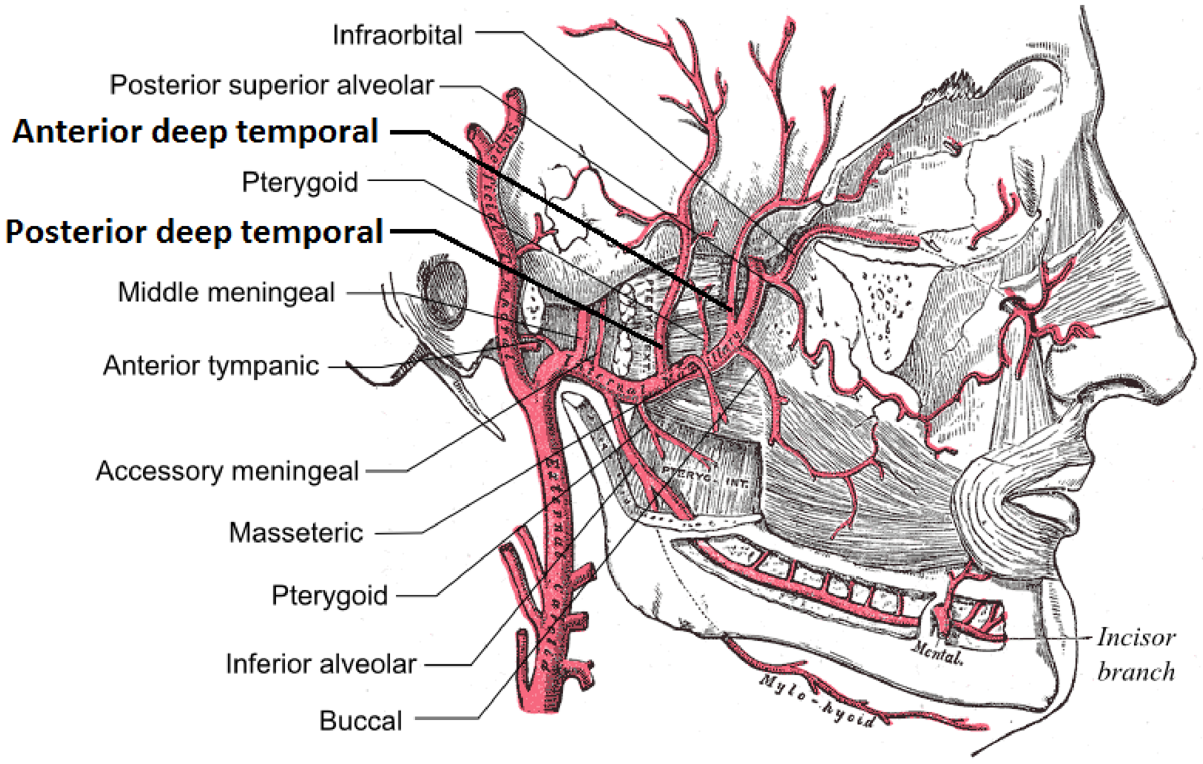


Figure 1.1: The branches of the maxillary artery [3].

As individual readings can be out of range due to inaccuracy of measure or error, it is also important to consider the rate of change. The human body changes in temperature at a relatively slow rate. If the entire day is spent at rest in a uniform temperature room the maximum temperature will occur at about 19:00 due to circadian rhythms and the lowest at 04:30; the difference between these extremes should be roughly 0.3°. This makes the rate of change negligible when compared to the error in readings over short time periods. If moderate activity is allowed the range can be as high as 0.5° in an hour. If this is set as a maximum allowable value, errors in measurement can be suppressed by filtering, as illness would not produce a rate higher than this.

Chapter 2

Background Technologies

2.1 Introduction

This chapter will outline the technologies used in this project and their capabilities with a specific focus on the aspects used for this system.

2.2 SunSPOT

SunSPOTs are a wireless sensor network device commonly called a mote. They contain a 32 bit ARM microprocessor and have 512 kilobytes of RAM and a total of 4 megabytes of flash memory. Each node has on-board light and temperature sensors as well three dimensional accelerometers. They have analog-digitals converters and five general purpose I/O pins. These give them the capability to be interfaced with other devices with both analog and digital outputs. However, the incoming signals must be brought to the correct levels of voltage and current as the SunSPOT can only handle a voltage range of 0V - 3V signals correctly. SunSPOTs come in two types, the first are 'free range' SunSPOTs, they have the sensors, I/O pins and a 3.7 volt Lithium ion battery. The other type of SunSPOT is the base station. Such devices have an 802.15.4 radio antenna. Their main use is to coordinate data from free range SunSPOTs. Both are accessible over USB and programmed in Java. There are plugins available to support this for the NetBeans IDE. SunSPOTS have their own dedicated project website sunspotworld.com [9]. It provides tutorial and sample code for the devices.

2.3 Bluegiga WT32

The Bluegiga WT32 Audio Module [10] is a Bluetooth 2.1 compatible chip. It has an integrated radio that operates at 2.4 gigahertz (GHz). The radio can transmit for up to 30 meters when operated in its standard temperature range of $-30°$ - $80°$. The chip was developed for use in high quality audio component so it has an analog digital converter and several digital signal processing functions such as decoding of compressed formats. It comes preloaded with the iWRAP firmware [11] which enables manipulation of the internal settings of the chip. It also supports many common Bluetooth profiles such as the hands free profile (HFP), a protocol frequently used for in-car connections. For this project the WT32 was used on a Sparkfun [12] breakout board. This board provides an RS-232 serial interface to the WT32 so it can be initialised and configured for the application. The serial connection needs to operate at 3.3V to match the board's voltage. The required voltage drop from a USB port's 5V was done by FTDIChip's TTL cable [13]. The board can be used to power the WT32's internal battery from an external source, which allows the WT32 to draw power from whatever it is connected to. The internal battery has an attached temperature sensor as it operates at such low power very little heat is dissipated by the battery so the sensor is able to provide quite accurate measurements of it's surroundings temperature. The experiments that support this are in section 6.2.

2.4 Bluetooth

The Bluetooth standard is a proprietary wireless communication specification. The first generation was released in 1994 and is now maintained by the Bluetooth Special Interest

Group. It operates on the range of 2.4 - 2.48 gigahertz (GHz). The 2.1 specification used in this project is backward compatible with all early versions. The main addition of this version was increased security due to secure simple pairing (SSP). SSP is based on public key cryptography and in most instances is designed to work without the need for user input. This does however compromise some of the security, as man in the middle attacks can break this approach. Man in the middle attacks are where a malicious third party inserts itself between two devices that are communicating while they believe they are communicating directly. The third party can then alter messages of inject fabricated ones. If protection against this type of attack is required the protocol's numeric comparison feature is required. It presupposes that the two devices have visual displays so a key can be displayed on one and entered on the other. Once this identification number (usually five digits) is entered the connection is considered secure.

The serial port profile (SPP) is a Bluetooth profile that has been adopted by the Bluetooth Special Interest Group. It is based on RFCOMM and European Telecommunications Standards Institute (ETSI) standards. RFCOMM is a transport protocol built on top of Logical link control and adaptation protocol (L2CAP). Together RFCOMM and L2CAP make up the data link layer in the Open Systems Interconnection (OSI) stack. With the SPP sitting on top of these technologies and providing emulation of an RS-232 connection. The emulation is most commonly used to provide wireless access to legacy devices that are only capable of wired serial output. Obviously a Bluetooth chip needs to be linked to the original serial output.

2.5 802.15.4

802.15.4 is an IEEE standard for both the physical layer and medium access control (MAC) that is intended to be used in personal area networks. It was specifically designed to function at low power. To achieve this it limits the data rate to a much lower one than what is typically possible. This not only lowers power consumption but also component cost as less complex circuits are needed to encode and decode signals. This makes it particularly attractive for embedded systems, which is why many common protocols are based on it. ZigBee and WiMi, for example, use 802.15.4 for the lower levels in their protocol stacks with higher levels designed for their respective intentions. ZigBee focuses on low power operation and secure networking while MiWi aims to provide reduced complexity and low cost operation. 802.15.4 can also be used with 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks), for the purpose of this project 6LoWPAN will not be used but it does offer an interesting possible extension to the system, in which data could be published to the Internet to allow for centralised monitoring .

In 802.15.4 there are two device types; full-function device (FFD) and reduced-function devices (RFD). FFD devices are capable of communicating with any other device in the network and if they are capable of relaying messages then they are known as a personal area network coordinator (PAN coordinator). RFDs are usually low power devices with limited capabilities. They can not communicate with all other nodes in a network, only with FFDs. They can never act as PAN coordinators.

There are two allowable network configurations that these devices can function in. The star shaped network has an FFD in the center acting as its PAN coordinator, each other device in the network communicates through it. The devices at the periphery can be either FFDs or RFDs. The other configuration is a peer to peer network, in which each node can communicate with any other. Typically RFDs will still communicate through a PAN due to their limited range. Inside either network configuration, addressing can be done as a 64 bit or 16 bit address. Networks can be chained together with one network's PAN coordinator communicating with the next network's one. In inter-network communication the 64 bit address must be used.

As both Bluetooth and 802.15.4 operate in similar frequency ranges the possibility of interference [14] needs to be considered. In most places the 2.4 GHz band is unlicensed so many consumer technologies operate in it. 802.15.4 splits this band into 16 channels and Bluetooth splits it in to 79 channels. As this system is intended for use in a home setting only likely sources of interference will be considered. As any standard that is adopted by the IEEE needs to have a Co-existence study done it is unlikely that any one technology will interfere with another. The possibility of problems arise with the cumulative effect of many technologies found in an unregulated environment. Bluetooth devices, especially class 2 (the lower power of the two device classes with a range of about 10 meters including the WT32 used in this project) operate at very low power and often employ power saving strategies so they lack the range and output to cause interference. WiFi operates at a higher signal strength and poses a greater risk; this is raised further by the fact that it is typically always on. Sikora et al [15] tested an 802.15.4 system with a 2 meter separation from an 802.11 router operating at maximum levels of utilization and found the 802.15.4's peak packet error rate across the band was 94%. While in normal usage, the level of WiFi activity and physical seperation is unlikly to occur, a significant rate of loss occurers if it does. To avoid this issue any 802.15.4 system should be situated as far from WiFi nodes as possible. There are also sources of interferance from unregulated sources. Commercially available microwave ovens should be fully shielded with a Faraday cage but this is not always the case. The possible effects of a poorly shielded microwave on 802.15.4 was also tested by Sikora et al [15]. They found that at ranges of over a meter there is no significant effect. Within this range however, packet errors were found.

2.6 Android

The operating system for the phone side of this project was not a significant decision as all the major operating systems support Bluetooth. Android was chosen for this project as development devices for it were readily available and no commercial software was required for application development or deployment.

The Android operating systems used were versions 2.1 and 2.2 (the device used was changed part way through the project for unrelated reasons), these are two of the more common deployments for smart phones. Bluetooth has been supported since version 2.0 so it or any subsequent version could be used for this project. The Android API lacks any plotting utility so a third party library was used. The Android Plot [16] library provides support for the graphing of both static and dynamic data. It has several chart types with dynamically variable scales. This made it ideal as an output method for the produced data.

The software development kit for Android has an inbuilt application programing interface (API) for Bluetooth, which takes care of much of the low level communications such as medium access control and flow control. The Bluetooth API requires all connections to have a server and at least one client. After the server finds and accepts a client there is no difference in the two device's roles. The device establishes an input stream and an output stream and reads and writes, as with any connection. The only major difference between programming for Bluetooth and other communications technologies is that the scan must be deactivated after finding a connection as it is both power and resource intensive. Any application for Android that wishes to use Bluetooth must specify the permission in its manifest (a general description for the system of the application and its requirements). This is in accordance with the Android marketplaces policy of displaying what device functionality an application can access, thereby alerting the user to possibly malicious applications.

2.7 Integrated Development Environments

2.7.1 Eclipse

The Eclipse IDE is a Java development environment that is used in this project for Android development. It integrates well with the Android SDK and it can deploy applications directly to an emulator or a device. Android emulators can also be managed through Eclipse. The main reason why Eclipse was used is the Android Development Tools (ADT) plugin, which adds Android options and tools directly into the interface which otherwise would have to be accessed separately. These tools allow for the debugging and monitoring of applications using break points.

2.7.2 NetBeans

The NetBeans IDE is another Java development environment. The reason that different IDEs are used for different parts of the project is that each of the plugins, Android and SunSPOT, were designed with a specific IDE in mind. While they can be ported to other IDEs, their functionality can be limited somewhat by this process. The plugin for SunSPOT development adds some options to the IDE such as direct deployment but it does not work particularly well, as a terminal program needs to be connected to get any output from the SunSPOT. Many functions such as updating the SunSPOTs' operating system must be done through a separate program, the SunSPOT Manager. The SunSPOT plugin does not provide the same amount of functionality as the corresponding Android plugin, but with each successive version more is added.

Chapter 3

Literature Review

3.1 Introduction

In the following section the state of the art in various, related fields will be presented. The review is split into four sections. The first, 'Sensor Architecture' (3.2.1) will compare the possible topologies and sensor configurations. It aims to provide a grounding in what is required of the sensor nodes in terms of performance and efficiency. It will present the challenges associated with deploying wireless sensors in a medical context. Some currently proposed systems will be examined and finally a short review of the medical conditions that can be detected will be provided. In effect this system will examine the sensing portion of wireless sensing systems.

The 'Network' section (3.3) will look at the transmission of data that results from the sensing phase of a system. The network topologies and power schemes will be analysed. Also in this section the issue of data privacy will be discussed.

3.2 Wireless Sensors

In this section wireless sensors will be discussed. The current thinking on sensor deployment in medical applications will be evaluated. There will be an outline of common architectures and topologies that have been deployed in the area of personal health monitoring. Some proposed sensor systems will be outlined and evaluated as to their relevance to this application. This section closes with a description of the range of values that the sensors must monitor and the accuracy that must be achieved.

3.2.1 Sensor Architecture

Wireless sensors are resource constrained devices; they operate at limited power and, in a medical context, must be capable of responding in a timely manner to both external commands and their own data. The challenges in their use are compounded by the fact that they have limited on-board processing capabilities and must maximise battery life. This section will discuss the topologies available and their motivations, as well as some novel approaches to common problems in wireless sensing.

In their paper, Chen et al. [17] outline the three main sensor network architectures. Cluster-based sensor networks consist of three node types (see figure 3.1). Cluster members are the most basic nodes that have very few capabilities. They only sense data and forward it to a cluster head. These are more complex nodes that can do some processing on the data provided to them by their cluster. The clusters report to some common data sink that does any high level work required. This structure introduces a significant work load on the cluster heads, as they must examine and route all the data produced by their section of the network. It does, however, allow for the cluster nodes to be kept very simple and power efficient. It also has advantages in applications where co-located grouping of sensors occurs and the natural partitioning can be taken advantage of. To address the power issues for the cluster heads, they can be dynamically allocated through an 'election'. In the election the node with highest battery levels becomes the cluster head. [18]

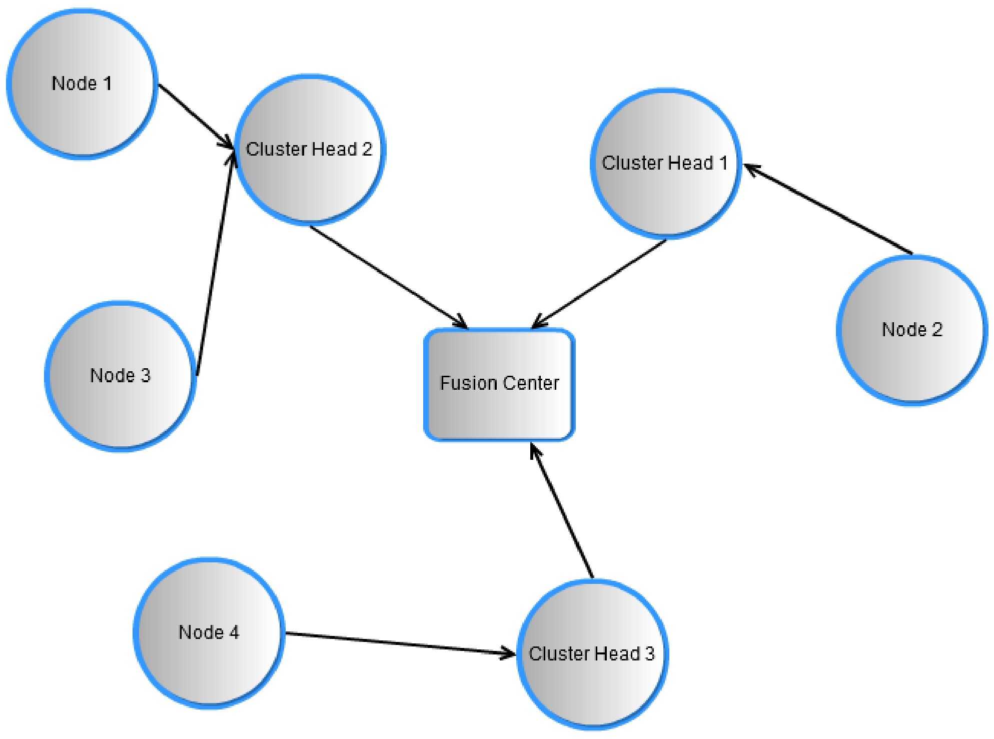


Figure 3.1: A cluster based topology.

A concatenated sensor network is one that does not have a central node (see figure 3.2). They may be organised in a purely peer to peer form, in a ring or in some such configuration. This configuration is typically power intensive as more frequent communication is required to coordinate functions of the network. However, this solution provides for greater flexibility and fault tolerance than static alternatives. It is often deployed where nodes are transient or the nodes' communications are intermittent.

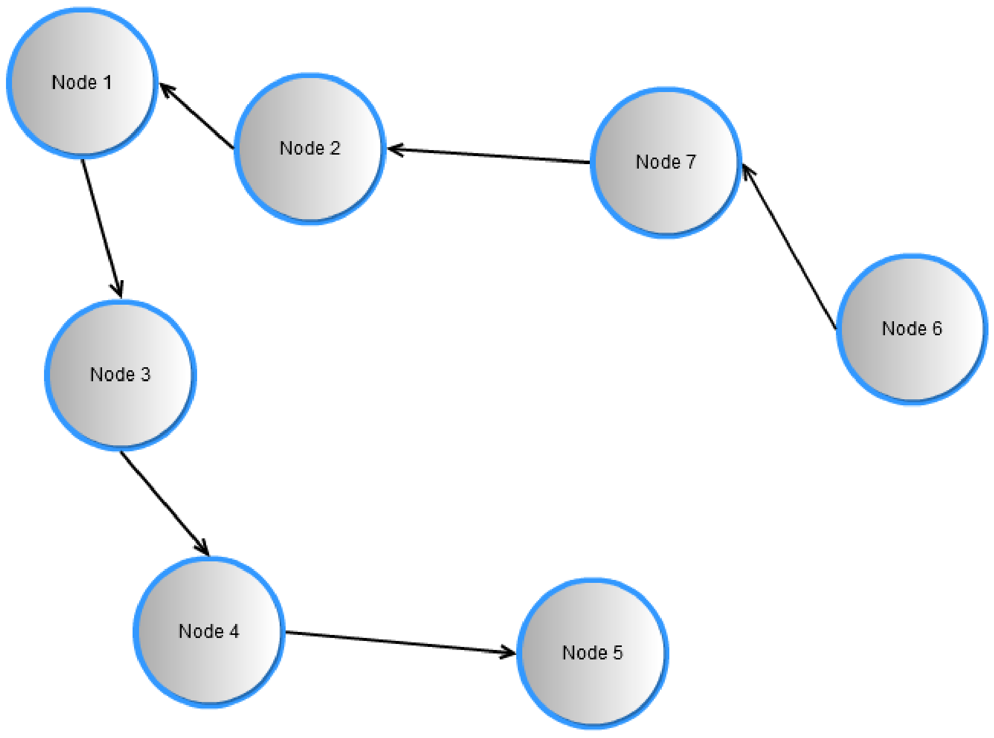


Figure 3.2: A concatenated sensor network topology.

To overcome these issues most architectures advocate moving the processing from the sensor node to some other location. This is perhaps the simplest topology, a sensor network with a fusion centre (see figure 3.3). The fusion centre is responsible for compiling all the gathered data and doing any analysis required. In their paper, Sivanadyan et al. [19] propose the use of a wireless information retriever (WIR). This device has significantly greater processing power than the sensor nodes and also has several receiving antennas so that it can calculate the phase of the signal with a reasonable degree of accuracy and thereby establish which node is transmitting data to it. The paper describes a signal hop protocol which means that the sensor nodes do not need to forward the traffic of other nodes. In general, the main advantage of such a configuration in wireless sensor networking is that it minimises the number of transmissions required from each node,

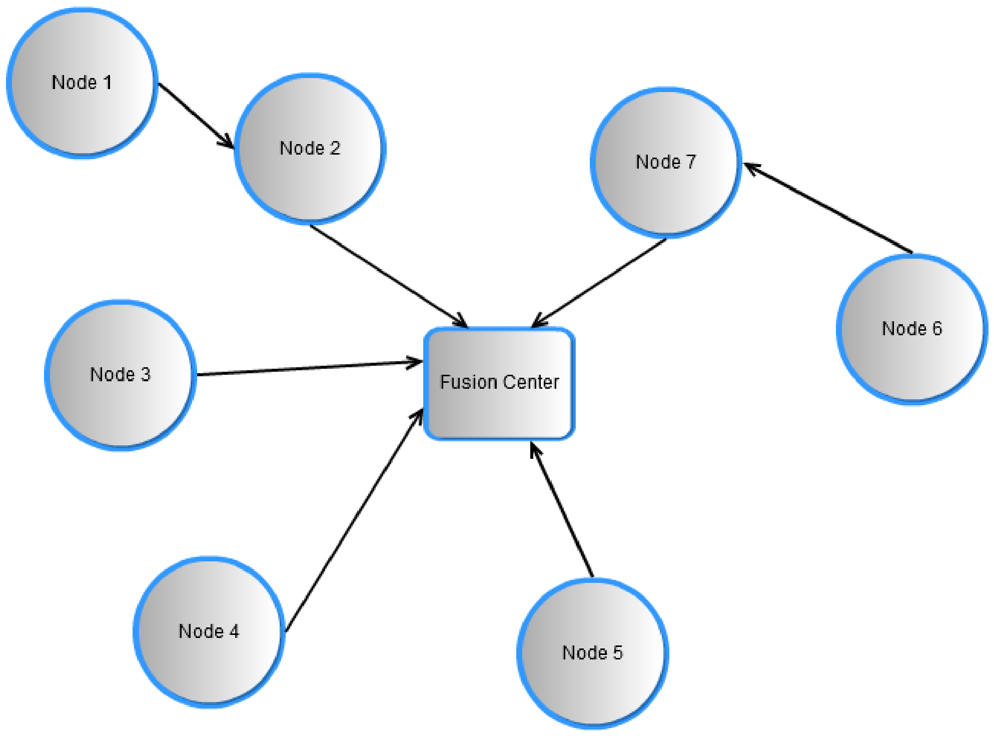


Figure 3.3: A topology showing a sensor network with a fusion center.

communications being a costly operation in terms of power. This solution requires the nodes to periodically listen to the network to allow the WIR to communicate with them and provide administrative information such as communication slot times. This is done even if they have no data to transmit. The simplicity of such a configuration makes deployment and analysis of systems easier. It also minimises the power needed to send each message, as only one communication cycle is needed from the sending and receiving nodes, whereas in a multi-hop architecture communications power is required from each intermediate node. More formally, $2N + 2$ cycles are needed where N is the number of hops.

The periodic wake up used by the WIR is an expensive operation. Zhang et al. [20] propose the use of a second communications channel that has the function of waking the device when the second channel is read from. They categorise sensors as having two possible operational functions, the first is a sensing node which is responsible for monitoring and reporting some physical value. To do this it must periodically wake up and take a reading and, if necessary, transmit this to the central node. The other device type is a stimulating node. This is a functional device that can take some action to correct a malady, e.g. an insulin pump. Due to the star topology (the above mentioned one-hop topology with a central node), if a condition requiring action is detected then the sensing node must report this to the central node, which must then wait for the stimulating node to wake up according to its own clock and then give it the instruction to take action. Depending on the stimulating node's wake up frequency this could lead to a delay in treatment or, if the wake up frequency is high, a large battery drain. The second communication channel removes the drawbacks with periodic wake up, as the central node can wake up the stimulating node when appropriate. This second channel also has the potential to improve the effectiveness of the system, as, given one parameter out of its normal range, other nodes can be queried to see if the sensor is malfunctioning or if it is a genuine medical problem that can be remedied. This secondary channel does not need to have a high data rate or provide much functionality. This makes radio-frequency identification (RFID) a interesting technology for this application, as it is very cost effective and widely available. RFID is capable of operating at a sufficiently high frequency to allow the transfer of power through the human body [21]. This then gives the potential to use devices that are permanently embedded in the body for long term monitoring without the need for replacement of batteries. Using the RFID channel for power management and charging would help in keeping such devices small, as a single antenna and accompanying circuitry can provide two necessary functions.

The power supply of a wireless sensor node is the main limiting factor in their lifespan, as many sensors monitor conditions in geographically inaccessible or difficult to access places. A completely passive system was proposed in the paper by Smith et al. [22]. The system roughly follows the idea of having one central fusion centre, but in this case the fusion centre provides the power to the devices when querying them. Under most conditions the WISPs are inert until a reader comes into range; the signal emitted by the reader causes the induction of current on the antenna, which in turn powers the sensor. This is the same way that RFID can be used to power nodes. The obvious draw back is that the WISP requires a reader to be in range to produce readings (they also propose scavenging energy from other mediums; attaching a solar panel to a light meter, for example). The platform does have the capability for long duration monitoring; a WISP could be embedded in newly poured concrete and only queried after an earthquake possibly years later in a structural health monitoring system. Such a system is not practical for providing a continuous data stream though, so it could not provide all the functionality needed for a medical system.

The impact of communications on battery life in wireless sensor nodes can be seen in the figures that are derived for a representative system in the paper by Agarwal et al. [23]. This system's sensing is provided by a thermistor and a light dependent resistor. It also incorporates a microcontroller and a transmitter module. The power consumption was 27% transmission and 28% reception over the whole cycle of operation, including configuration and shut down. This shows that over half the drawn power is used in communications, so reducing the time spent of the networking phase can significantly improve the lifetime of the sensor node. These results were the average of several experiments and network configurations, so they can be seen as a median value rather than as expected values for any one configuration of a system.

Energy awareness is not the only issue affecting the layout of sensor networks. In their paper, Cheng et al. [18] describe a network that has been optimised to produce the shortest possible average latency. This is particularly important in systems that are providing sensor data to an actuation and control system that produces real time output. Such applications use wireless sensors to obtain the state of the environment in near real-time and thereby take the most appropriate action. To achieve this, a series of algorithms are proposed that reconfigure a network's inter-node connections so that as much simultaneous communication as possible can happen. This allows the overall end to end transit time for the network to be reduced. This is particularly important in a medical context, as in an emergency monitoring situation, both the initial detection and instructions for remedial action must be handled by the network in a timely manner.

3.2.2 Continuous Monitoring

In the medical context, the regularity of data collection and the quantisation of data are critical, as a system must have the ability to detect rapid or transient changes and record them accurately. In this section these issues will be discussed with regard to the necessary duty cycle of the sensors and the issues associated with continuous monitoring of a sensor. Constantly recording and transmitting a parameter introduces a significant power overhead as discussed above; however, this can be unavoidable. Monitoring heart rate, for example, needs to happen constantly as any response needs to be instantaneous. There are other parameters that change much more slowly in the human body, temperature to name one. This disparity in data generation creates several issues.

Fair collection of data is a problem discussed by Jin et al. [24]. They term data that needs to be processed in real-time as inelastic, this data constantly requires bandwidth and usually a good quality of service is required to transmit it. This means it can effectively block the elastic traffic (that being traffic that does not need constant reporting or data that is not time-sensitive) if the network is not well designed. They propose a multi-layered algorithm that implements a queue at both sending and receiving nodes so that elastic traffic can be interspersed with inelastic when the network's capacity is not at maximum. The algorithm essentially calculates back-pressure and throttles data accordingly to minimise the data awaiting transmission with respect to its priority. This approach creates large demand for both bandwidth and energy, so it is preferable in a resource constrained system to do the processing locally on the reading node or at its cluster head depending on the underlying topology. The following step is reporting any out of bounds value to the higher level node, thus freeing up the bandwidth. This local filtering of data, as well as costing energy, causes a reduction in the amount of historic data that is logged, which, depending on the application, could be significant in domain specific analysis.

In their paper Cheng et al. [25] present a regularly spaced network that has the function of detecting mobile sources. A method of allowing overlapping nodes to sleep while maintaining a probabilistically determined adequate coverage is discussed. The system estimates the risk of missing a detection of the source and the probability of the source being present and arrives at a metric for whether allowing a node to sleep will reduce effectiveness of the network. While the proposed system relies on a regular deployment of nodes with geographic knowledge it could be extended to work on any arrangement that is aware of its node's locations. This is an arbitrary configuration. However, the chance of significant overlap is reduced. This paper discusses the risk involved in missing a detection of a geographically variant event, but it can equally be applied to a temporally variant one and used in a medical application to quantify the risk of missing a transient condition in the gap between wake up cycles. In its original geographic implementation, the algorithms could be applied, without modification, to quantifying if a person is in range of a medical sensor.

There are two main methods to accommodate continuous monitoring in wireless sensor networks. The first is to provide support for delivery of all generated data and the other is to quantise its measurement in some way. Both methods have strengths and weaknesses, but due to the limitations of sensor nodes the quantisation approach is more widely used. The above mentioned wireless information retriever (WIR) [19] could be used to provide for either approach. If the sensing array is sparse then it can allow for simultaneous receiving from several devices and resolving the angular delay in the signal to separate the received data. This, obviously, could be used to receive from several continuous sources at once. If, alternatively, the sensing array is dense, then the possible resolution of the angular delay is too course-grain to resolve many signals, so 'bins' must be used with each bin internally resolving for control of its channel. Any fair medium access control policy to allow for channel resolution will block continuous data at least some of the time, forcing quantisation.

A purely continuous approach is proposed by Goutham et al. [26]. The authors discuss a system that monitors temperature and the heart's electrical activity, electro-cardiogram (ECG). It converts the sinusoidal signals into digital samples and transmits them over the cellular network. Each packet is capable of holding two seconds of ECG data and a temperature measurement. The system is limited, sensing only two parameters by the permissible packet size on GSM networks. It does however allow for real time remote monitoring with only a small delay. The major benefit of such a system is the ubiquity of the cellular infrastructure. It would allow for mobility in monitoring and reduce the cost involved in deploying such a system significantly.

An intelligent approach to continuous data is discussed by Cheng et al. [27]. The system is capable of monitoring and locally logging several parameters. To save on memory, the logging can be programmatically controlled according to what the system is to monitor. If a value goes out of a predefined range then it will be logged, otherwise it will be disregarded. Similarly the period when data should be logged can be set, as can the frequency of logging. While this system logs locally, as it is based on radio frequency identification (RFID) tags, it could be adapted to publish the data directly to a network. When applied to medical data this could be seen as reporting every time heart rate deviated from normal. The normals for biometric parameters could also be dynamically established to improve accuracy and to adapt to normal circadian rhythms. This system would be better suited to monitoring the health of someone generally and reporting when they become ill, however, it would not be particularly effective at tracking the progress of a condition as so much data is discarded.

An extreme implementation of the quantisation approach to continuous monitoring is to remove the use of historical data and only provide data when queried by some higher level process. This is practical in systems that provide intelligent diagnoses. Such systems chain together samples of biometric data to detect any possible conditions. As they are intelligent, they know what data they need at any given time and can request it from the sensors. The sensor proposed by Kocer et al. [28] is particularly useful in data collection of this manner, as it scavenges power from available radio frequencies. This allows for long term monitoring without battery considerations. They do, however, have to operate at low-power (due to limitations on recoverable energy), which makes only communicating when required desirable. A similar sensor device is proposed by Reindl et al. [29]. It uses a different method to determine temperature. Both of these devices are designed to be left in situ and only operate when queried. This gives them limited usefulness in short term monitoring.

3.2.3 Proposed Alternatives

The measurement and analysis of biometric parameters by non-contact means is difficult, as many biological processes produce no external evidence. This creates a subset of parameters that can be observed and measured by a monitoring system. Several systems have been proposed and they can be split into two broad categories. Purely non-contact systems and those that use contact sensors. Non-contact systems rely on remote measurements such as chemical analysis of exhaled air, while contact systems that use devices mounted on or embedded in the body. Such systems often measure temperature or electrocardiogram (ECG).

The system proposed by Cao et al. [30] is purely non-contact. It is proposed to place an array of Carbon dioxide (CO_2) detectors on the bars of a crib. The CO_2 sensors are intended to detect irregularities in the breathing of an infant. The absence of CO_2 above the background level could indicate that the infant has stopped breathing. This system was initially intended for the detection of Sudden Infant Death Syndrome (SIDS), but could be extended to other breathing issues. The sensors are distributed evenly around the crib to allow for a greater detection area. The results from each sensor are then read, and if they fall below a threshold of between 2000 and 5000 ppm (parts per million) an alarm is activated. Much of the work on this system, as in any chemically based one, was in validating its effectiveness at different temperature and humidity ranges. When exhaled, the air contains around 4% CO_2 or 40,000 ppm. This rapidly diffuses due to the normal air's lower concentration. This is a challenge to any chemical sensor; the concentration needs to be significantly higher than the background to be reliably detected at range. In this system a CO_2 concentration raised by breathing directly on the sensor causes a 0.15V (volts) jump in output voltage while a breath at a typical distance according to their system raises voltage by about 0.08V. Both of these readings cause only a small percentage change in output voltage of the sensor, making detection difficult. Another approach to non-contact monitoring is discussed by Yan et al. [31]. They propose the use of Doppler radar to monitor infants in cribs. The radar is capable of detecting heart and respiration rate, even through non-metallic occluding objects. In their experiments, they use a simulator to generate the biometric signals associated with various medical conditions to test if the radar can detect them; for the most part, this is successful. However, it fails to detect bradypnea (abnormally slow breathing) and can occasionally confuse harmonics of a signal with the signal itself. This can usually be rectified by adding in some basic physiological knowledge. The use of non-contact monitoring allows for continuous, unobtrusive monitoring, which, due to its unreliability, should only be used as an indicator of a possible condition rather that a definitive answer system. Chemical sensors can only operate and be used to detect medical maladies if they produce a sufficiently large output. This limits the sensors range and they also suffer from interference from local sources. Radar units generate a lot of heat and noise so they are not particularly suited to home monitoring.

The alternate approach is to use contact sensing. This approach is taken by Chen et al. [32]. In the system proposed, they advocate a four level hierarchy to aid power control and division of functionality. The top level is the system layer. It is proposed that this be situated in a hospital. It is intended to send commands and receive data from the lower level. As it is located in a hospital it will provide the sensor data to experts for analysis as well as having centralised storage. Next is the application layer which is responsible for coordinating the lower level sensor groups. The sensor group layer is similar in function to the above mentioned cluster heads, in that it coordinates the operation of individual sensors and concatenates the data that is produced. The bottom layer is the sensor layer, and is composed of individual sensor nodes taking readings and reporting to the next level. The nodes at the sensor level are capable of monitoring temperature, pulse Oxygen (the oxygenation of haemoglobin), blood glucose, blood pressure and electroencephalogram (EEG) (the monitoring of the brain's electrical activity). All of these sensors require contact with a patient to obtain readings. The main focus of this system was in power reduction. To achieve this they propose an algorithm to adaptively put the device and its transceiver to sleep. Using this method, power consumption drops by around 99% for low network activity and 97% for high activity. Contact monitoring tends to be more accurate, as well as increasing the number of possible parameters that can be monitored. However, the patient must wear the relevant devices. For this reason it is usually only used for people with a specific ailment that has already been diagnosed rather than preventative or early warning monitoring. This system also relied heavily on the central servers in the hospital.

The challenge is to provide the unobtrusiveness of non-contact solutions with the reliability of contact ones. This is attempted by Corchado et al. [33] with the SYLPH platform. They describe a service oriented architecture where the individual sensor nodes have functionality embedded into them. This allows for distributed cooperation and integration into current heterogeneous wireless sensor networks. The system is designed in accordance with the ambient intelligence paradigm, while SYLPH itself is designed as a middle ware that provides a uniform way for the various processes to communicate regardless of how they are configured. The middle ware has three main components, all of which run on wireless sensor nodes. An ordinary node is one that has no specific function; they can be used to store code for a service or call a service on another node. A SYLPH service functions in the same way as a standard web service. The SYLPH platform has its own service definition language called SYLPH service definition language (SSDL). SSDL is a lightweight version of WSDL (web service definition language) which requires less power than WSDL, so it is ideal for constrained devices. Before any services can be used, they must register with a directory node. The directory nodes list available services and the locations they can be found. To give greater fault tolerance both services and directories can be replicated.

The SYLPH platform can be integrated with any form of wireless sensor network, allowing it to be as flexible or accurate as an application requires. The service oriented architecture means that a greater load balancing is possible than with conventional systems. This is especially true in an environment where not all devices need to be mobile. In the paper they discuss a home monitoring example. If devices with external power were brought into the SYLPH system then a sort of sensing cloud could be developed with work being passed off to the more capable nodes and the simple monitoring being all that the constrained devices handle.

3.3 Network

This section will discuss the networking aspects of sensor networks, both in terms of inter-node communication and in relaying the data to sinks outside of the wireless sensor network. It will examine current thinking on medium access control and traffic modelling in medical applications. Due to the limited nature of sensor devices, power efficiency will be discussed and finally privacy and security in medical networks will be dealt with.

3.3.1 Network Topologies

This section will discuss some of the possible network configurations that can be used to report data to a central sink. It is distinct from the Sensor Architecture section 3.2.1 as that is concerned with how nodes are deployed and relaying data to the base station. This section will focus on how the data from one or more base stations is routed to a location. It will also briefly discuss the higher level protocols that provide addressing and routing.

A Mesh network is a network where many nodes communicate over a variety of wireless communication technologies to provide a flexible infrastructure with a high bandwidth [34]. There is no need for a wired backbone in mesh networks so they are quite tolerant of failures, as, if one node drops out, traffic can be routed around it. As wireless communications provide the basic infrastructure there is inherently much greater support for mobile nodes. There are two types of node in these networks. Mesh clients are nodes with a wireless interface that only send and receive data from the network. They provide no forwarding or routing support to the network. In most implementations of mesh networks they are laptops and mobile phones. The mesh router is a much more complex node. It functions much as would be expected from an Internet router, providing repeating and routing services. It differs in the number of communications technologies it supports. A conventional router only handles data on one medium, a WiFi router and radio waves for example. Mesh routers typically provide several network interfaces across different technologies with WiFi and WiMax being common.

When using wireless mesh networks in personal area networking, a low power solution is required. Lee et al. [35] discuss the IEEE's standard governing mesh body area networks 802.15.5. The standard specifies an architecture that provides for two different data rates, the lower of which is intended for wireless sensor networks and is the main focus of their work. The architecture is scalable and, as it is designed for wireless sensor networks, it operates at a low power without the need for complex hardware. 802.15.5 is an extension of the 802.15.4 protocol. 802.15.4 has several different implementations and extensions, 6LoWPAN and ZigBee being the most notable. 802.15.4 only allows for star based networks and one hop routing. This is the main issue addressed by the extensions. ZigBee uses the ad-hoc on-demand distance vector (AODV) routing to allow more complex configurations as well as increased support for mesh networks. 6LoWPAN's main extension is the use of IPv6 addresses and header compression. This allow sensors to be directly interoperable from the Internet through the IPv6 protocol. 802.15.5 provides a similar set of extensions to provide mesh networking. The major advantage of 802.15.5 over other protocols is that its routing protocol does not require a network wide broadcast as AODV does. This reduces the amount of messages that need to be sent while also reducing the demands on nodes. A node does not need to store and maintain a large routing table under the logical tree based routing that is used. The extra power drain associated with cluster heads is also reduced. In 802.15.5 the routing is done by producing a tree of address spaces. The tree is similar in structure to a B tree (see figure 3.4). The diagram shows an address range of 0-10. Nodes 6 and 8 have no children as they are either physically isolated or operating at low power. The figure shows how the logical addresses are associated with an area of the network. The physical structure of the tree describes the organisation of the network. Each node is required to announce itself in the initialisation of the network. Based on the number of announcements they receive they position themselves in the hierarchy. The address space is assigned based on the tree's structure. There is a final round of local communication to advertise the range of addresses a node has and the network is configured. When attempting to rout messages there are four possible cases. The first is that a received message is for a child of the receiving node. In this case the address will have been allocated by the node so the message can be delivered. The second case is that the message is for an known neighbour. If the neighbour is one hop then the message can be sent, if it is two hops then the connectivity matrix (a matrix that takes account of signal strength as well as the number of children of the nodes so that some power balancing can be achieved) is used to decide the best route. The third case is if the message is for a node in a neighbour's address range (a child), the message is passed to the neighbour as in case two but with the expectation that it will be forwarded. The final case is that the receiving node has no information about the intended recipient. In this case a guess must be made. A logical idea is to give the message to the nodes parent. The mesh structure from 802.15.5 allows for improved scalability and lower power is required for each nodes operation. The major advantage to this configuration however is in its resistance to faults and network interference.

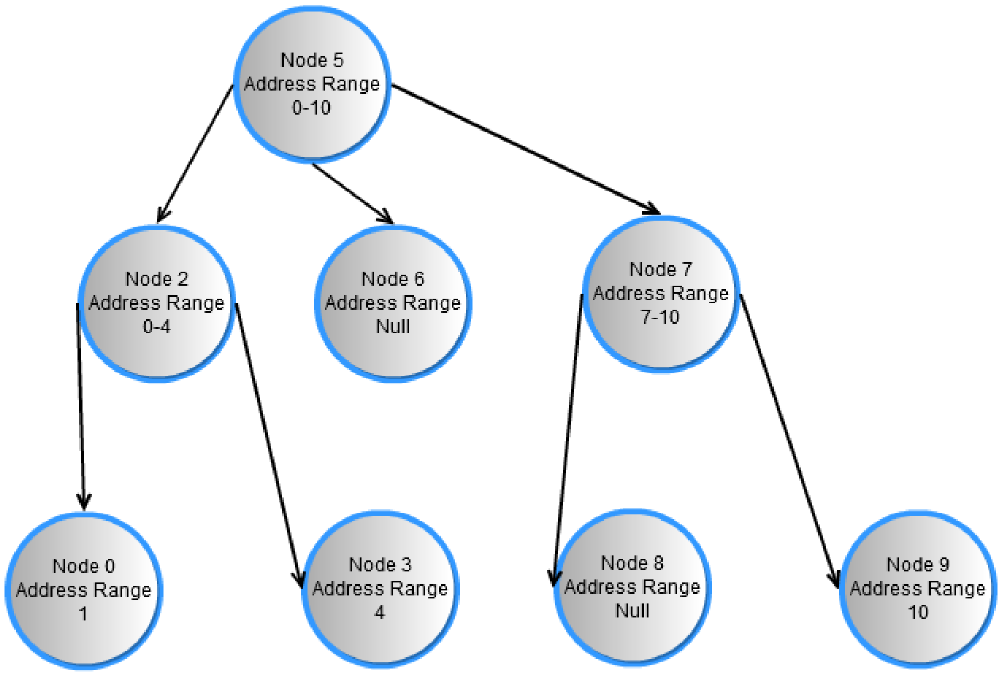


Figure 3.4: The logical structure of an 802.15.5 network.

3.3.2 Medium Access Control

Medium access control (MAC) describes the method used to allow use of a network without interfering with other communications. There are two broad categories. The first group are those that attempt to allocate the network based on some fixed aspect of the network e.g. time or frequency. The other set are the methods where nodes communicate based on network activity. In carrier sense multiple access with collision avoidance (CSMA\CA) the node wishing to transmit must check to see if the network is available; if not it waits. In reality both types are used extensively with the first type being used in cellular and satellite networks and the other in computer networks. For use in wireless sensor networks, however, these schemes are too power hungry or inflexible so Omeni et al. [36] propose a scheme based on synchronising master-slave wake up times for all the nodes in a network. This MAC protocol is designed for a simple sensor architecture with a fusion centre. Such a configuration is typical of body area networks (for which this was intended) as relatively few nodes are needed and it is desirable to only have one external transceiver due to their bulkiness. As the protocol was designed for medical applications it has the facility for 'emergency' communication in which a node can choose to communicate outside of its scheduled slot. Between wake up cycles the master polls the nodes to improve response times to out of bound values. The novel proposition is the 'Wake Up- Fall Back' timer. If communication in a scheduled slot fails then the master and node reorganise communication for the same time with out needing agreement. This new time is based on the networks sleep time. Its use allows for servicing of emergency communication and regular slots with out blocking either. This system is similar to a standard time division multiple access (TDMA) protocol, the main advantage of which is energy efficiency, as the nodes can spend the majority of time without doing network communications. This particular implementation is very simple in hardware, requiring only about twelve thousand transistors for the node which further reduces power consumption when compared to most protocols. According to the experiments conducted it uses only 17% of the power consumed by ZigBee's MAC protocol and only 3.5% of that used by 802.11. When implemented in software, these figures will rise slightly as the processor will need to active more than is strictly necessary to provide the node's function. The challenge in this approach is in maintaining clock synchronization between nodes. In the presented solution this is made easier by the central sink, as resource constrained nodes do not need to coordinate their wake up cycles. It is further simplified by the reliability of the communication links between devices and their physical proximity. There is practically no latency so the central node can just prescribe the time to the sensor. This solution will not work in lager scale applications as latency will become an issue.

3.3.3 Traffic Models

The data generated by wireless sensor networks in medical applications has several interesting characteristics. The instantaneous values for most biometric properties are not particularly significant. For example a single glucose level indicates very little but its rate of change over time is significant in conditions like diabetes. Heart rate is a notable exception to this as it is a rapidly changing parameter. This allows some data to be quantised and transmitted in batches when the network is available. There are obvious power benefits to this. The balancing of these types of data is discussed by Jin et al. [24] as mentioned above, this section will use the same terminology. Messier et al. [37] developed a traffic model for medical sensor networks. Their paper uses temperature readings to represent elastic traffic and electrocardiogram (ECG) for inelastic. The transition and compression of ECG signals is discussed with their lossless compression algorithm. A ratio of 2:1 can be achieved as it is a relatively simple algorithm the energy used in computation does not exceed the saving in transition. The ratio of 2:1 could be improved upon using lossy compression but the signal distortion is not acceptable as it can alter any potential diagnostic analysis. The temperature data was 11 bits long and even at this short length it was found compression would improve node battery life. The required rate of data presentation depends on the application. If real time monitoring is needed then constant data is required but if a systems purpose is to provide only an indication of a problem then the data transmission can be constrained to exceptional values. The way the data is presented also influences the traffic model. If the data is only occasionally viewed by a user then only small segments of it need to be transmitted the rest can be logged to local storage or discarded. However if the data is continuously monitored then a quality of service is expected, depending on the underlying network there may be a trade off between power usage and quality of service to an observer.

3.3.4 Energy Awareness and Power Saving

In the above sections many issues that affect power consumption are discussed. This section will go into detail about the effects of low power operation on communications and the accuracy with which they can report their state.

Quevedo et al. [38] discuss how the low power operation is achieved and the effects it has on communication. It is common to reduce header information for each data item so that the ratio of useful data to transmitted data rises. Also if reduction of packet size is used this leads to shorter communication cycles which cuts power usage. Both methods can lead to quantisation and insufficient precision if not carefully implemented. This raises the rates of packet loss and retransmission which effectively wastes power. The probability of introducing errors into a transmission is directly proportional to the power supplied to the sensor node's radio amplifier. This means that to achieve a real power saving the reduction in power consumed at the radio must not be completely offset by the power required for retransmission of corrupted data. A control strategy is presented that aims to do this. Packet loss as a power usage factor is not only due to low power transmission alone. It can occur because of network interference, occlusion of the receiving antenna or packet fragmentation. This means that the power usage of a node can be affected by the positioning of both it and its neighbours. The impact of topology on node lifetime is discussed by Chen et al. [17]. They find that a cluster based network becomes much more energy efficient than other organisations when there are greater that 20 nodes. Prior to that a fusion centred layout is more efficient. In terms of lifetime, if there are fewer than 45 nodes the cluster based network is best, after that a fusion centre becomes the preferred configuration.

3.3.5 Privacy and Security

This section will discuss the potential vulnerabilities in wireless sensor networks and common methods of attack that they must be capable of withstanding. A brief overview of how to protect personal health data that is being transmitted over potentially unsecured networks will be provided.

A survey of the attacks possible on wireless sensor networks are presented by Martins et al. [39]. There are three main categories of attacks, the first are attacks against a physical node. These are not necessarily malicious, they can be caused by accident or due to a sensor malfunction. A node can be attacked by preventing its sleep cycle either through constantly sending messages or by assigning it work to complete. Over time these attacks drain the node's battery, rendering it useless. The physical environment can be altered to drive the node's sensor readings to abnormal levels. Holding a flame next to a smoke detector to trigger a fire alarm is an example of this. The final type of attack in this category are attacks against the physical node, sabotage or theft for instance. Attacks in this category are difficult to defend against. The lack of sleep can be prevented by implementing a power aware scheduling algorithm and passing off work when operating at low battery. The impact of such attacks is, however, minimal as the data produced remains trusted and intact.

The second group of attacks involve inserting one or more malicious nodes into a network. If only one node is inserted there are several possible attacks. It can slow down important messages that are routed through it. This can lead to delay in detection of significant events. There is also the possibility of message alteration. This can be done on either the meta data or the payload data. A black hole attack is one in which the node stops forwarding any messages, if some are forwarded it is known as a grey hole. This type of attack becomes more effective the closer the malicious node is situated to the data sink. A grey hole is more difficult to detect than a black hole as it can respond as a trusted node would under some conditions. When a black hole attack is coupled with a Sybil attack (an attack in which a node pretends to be many nodes to influence the outcome of a cluster head election) the effectiveness dramatically increases as a node can logically move itself closer to the data sink. Multiple node insertions can be used to generate infinite loops and flood the network preventing useful message from getting to their destination. If the inserted nodes are connected by a faster link than the rest of the network they can conspire to distort the networks topology so that more traffic is routed through them from this they can execute other attacks. Defending against these attacks is possible as there are profiles to the data generated that can be analysed to detect the intruder. A detected and excluded intruder can still cause problems as they are capable of operating at the same frequency as the network. Once a criminal node is within range of the network it can never be completely isolated from the network.

The final attack type are those that attack the network connecting the nodes, these can be generated from a node or some other device that has a compatible interface to the network. The simplest of this category of attack is to eavesdrop. Jamming of the communication channel with a powerful transmitter is also possible. Message replication and injection can be accomplished through this category. Hello flooding is when many announce messages are sent to prevent useful work being done. This can only be done under routing protocols that expect announce messages, it is a type of denial of service attack. These attack are difficult to defend against as they require filtering through many malicious messages to find trusted ones. Even if this filtering is successful it is still a power intensive operation so the lifetime of the node is still reduced.

As many of the above attacks on sensor networks can not be prevented in a practical way, a system must focus on those that can be defended against. Due to the nature of medical and personal data the use of encryption to stop eavesdropping is important. Aysal et al. [40] present a one bit encryption scheme in which each side has a shared probability function that represents the likelihood of a bit being flipped given an input. They constrain the data length to one bit to allow for the resource constraints found in sensor networks. Their cryptographic scheme is analysed with respect to the bias it introduces to malicious fusion centres that are trying to achieve a state estimation based on intercepted sensor readings. By choosing design parameters for the enciphering algorithm correctly, a large bias can be generated in the intercepted data. Other conventional cryptographic techniques can be applied but many require hardware support or use considerable energy in their implementation. A further challenge is introduced as the data generated by a medical monitoring system tends to be around the same value. Given this information the time taken to crack any encryption is reduced.

3.3.6 Technologies

For use in personal area networks (PANs) many networking technologies are not practical. Lo et al. [41] discuss the requirements for the communications technologies used in personal area networking. They find that the following requirements need to be met for a technology to be use full in their vision of body area networks. The networking technology needs to have high availability. Whenever a device in the network has data to report, it can do so without delay or limiting the quantity of data to transmit. This is particularly important in PANs that aim to provide real time monitoring and alerting capabilities. Another requirement is that devices can operate at low power. This is essential as they predict monitoring devices embedded in the body that would be difficult to charge. To facilitate low power operation devices only need to have a small communications range (the technologies they discuss have ranges of about 10 meters see figure 3.5) once they can interface with devices capable of longer distance transmission so that periodically data can be logged to external storage or relayed to medical personnel. Finally, the devices must be implementable at low cost, so that they can be wildly used and if necessary made disposable. They present two technologies that meet these criteria. The first is High

Data Rate WPAN (IEEE 802.15.3 [42]). 802.15.3 is a low level communications technology (covering levels 1 and 2 of the OSI stack) that offers data rates from 11 - 55 Mbit per second. Devices using this technology can be configured into a Piconet. This paper uses the term Piconet to refer to a topology with a central node called the Piconet Controller (PNC), there can be only on PNC per network. A Piconet can have up to 255 slaves and a PNC when used with 802.15.3. The nodes can communicate in a peer to peer fashion or directly to the PNC. The other technology proposed is Bluetooth. Bluetooth was adopted as the IEEE standard 802.15.1 [43] so both names reference the same technology. They describe a Bluetooth Piconet as one PNC per 7 slaves. Bluetooth offers a lesser data rate (1 Mbit per second at Bluetooth prior to version 2.0 2 Mbit per second after) and less potential devices, however, due to its widespread use, devices are much cheaper and for many application, the extra speed is not required.

A further discussion of the usefulness of Bluetooth in PANs is presented by Rashid et al. [44]. They analyse the delays that can be expected when using Bluetooth to communicate at various ranges. To do this they send various sizes of file over the distances of 2, 4 and 6 meters, they also repeat the 2 meter experiment through a concrete wall. They find that for file sizes less that 1 Kbyte, distance has little affect on transmission time. For files up to 5 Kbyte the delay at 6 meters is approximately 25% greater than the delay at 2 meters. In all experiments transmission through a wall is comperable to the 6 meter time. They also find that for files upto 1 Kbyte the average transmission time grows portionaly with file size. For files over this size it grows nearly exponentialy. The choice of communications technology should be motivated by the amount of data that will be generated. Bluetooth is practical for data sizes of under 1 Kbyte; over this the delay is unacceptable and a diferent technology should be used.

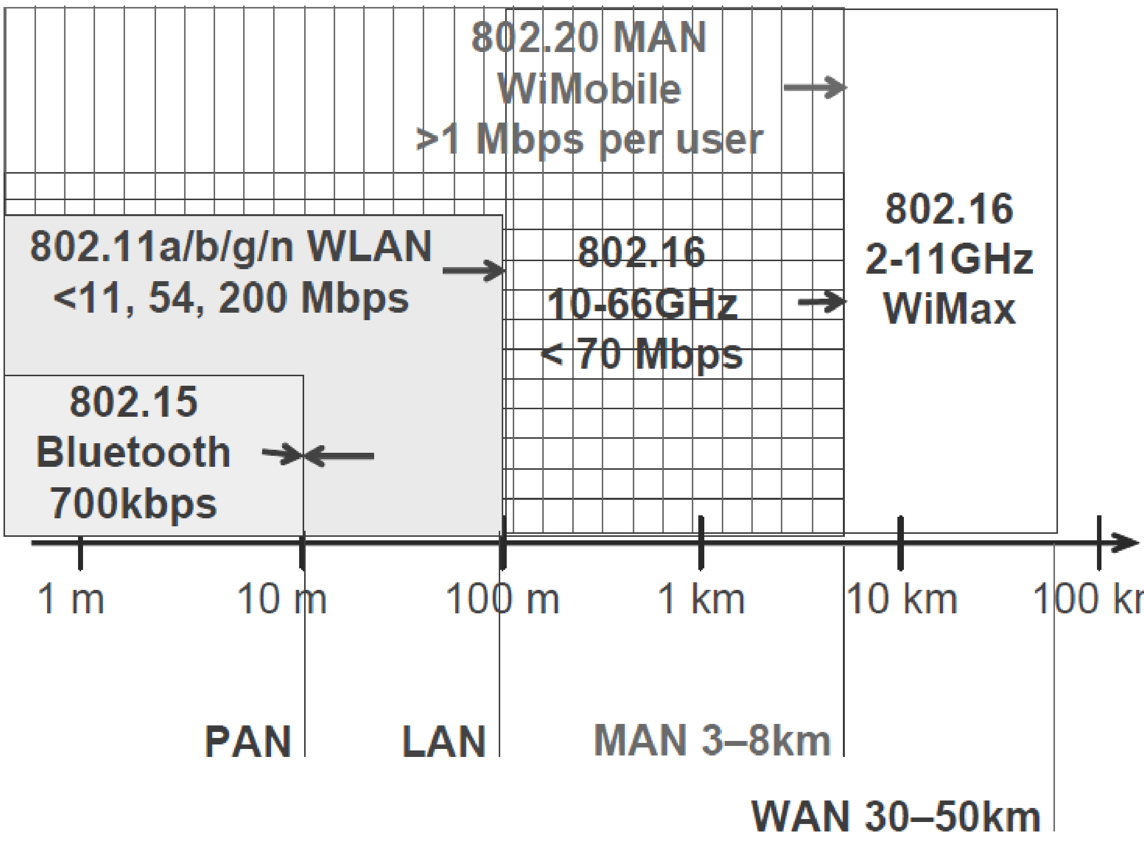


Figure 3.5: A comparison of the ranges and data rates of several wireless communications technologies [4].

Chapter 4

Design

4.1 Introduction

This section of the report will deal with the system's design. It will describe how the project's various sections are divided and the functionally that each part has. It will also explain the motivation behind the different technologies used and the reasons that the described architecture was used. Both hardware and software will be relevant to this section.

4.2 Overview

The project's design has broadly three main functions to provide; reading of data from a source; if required transmission of the data and display of the data. As the sensing needed to be independent of the display so that data could be recorded in a remote location, the system needed to have at least two independent hardware devices. In the prototype system the base station functionality and the temperature sensor are integrated into one device. The other device in the prototype is the phone that is used to display and analyse the data that has been gathered. In the actual system the sensing would be done by a small near contact device with an attached Bluetooth transceiver. In a deployment scenario the base station would manage the remote devices and any available environmental sensors. From the base station, data can be streamed to any device capable of displaying and analysing it. For most cases it would be logical to direct it to a local device, such as a phone, but if used in remote monitoring of patients, the data could be sent to a web service so that it could be managed by some centralised location for a doctor.

The difference between the intended configuration and that of the prototype was due mainly to time and available devices. In the prototype, the remote near contact devices, which were to consist of a sensor and Bluetooth chip, were replaced by the on-board temperature sensor of the base station's Bluetooth device. This removed the need for implementing a thermometer with serial data output that could interface with a commercially available Bluetooth chip. For these reasons the rest of this chapter will deal with the design of the two device system, that being the sensing base station and the phone as an output device.

4.3 Functional Architecture

This section will discus the general architecture of the project and the required functionality both at the level of the individual device and that of the system as a whole. As each device in the project had distinct functionality to provide, the logical designs were quite different. The architecture presented in the SunSPOT section 4.3.2 will be similar to the that of the intended configurations's base station. The only difference is in the implementation of the method of connecting the sensors, in the prototype's case serial wires rather than Bluetooth.

4.3.1 Phone Side

The architecture of the phone side application follows the Model-View-Controller (MVC) design pattern, which provided a basis for the development of the program. In MVC, the various aspects of the applications logic are separated out from one another so all domain logic is isolated from the user interface. This was particularly useful in developing this system, as the internal logic for processing sensor data changed often. The application was effectively designed to display the most recent values of an ever expanding data set. The model was planned as a simple data set that could provide accessor and search methods on the stored data. The view was to display the subset of data that the controller specified. The controller was to be the most complex element. It needed to handle the input of new data and analysis of it. This was accomplished by abstracting the communications into its own part of the controller and away from the rest of the controller so that there were effectively two controllers, one that handled the reception and analysis of new data, and the other that decided which data points should be displayed. The first was given priority to reduce the risk that data would be lost if the phone side buffer overflowed. As any significant analysis of data was to be done on the phone side, to save power in the other devices that could not be so readily recharged, the controller needed to be capable of doing possible lengthy calculation without causing problems in other parts of the application. For this reason it needed to operate in two threads, one for each controller. The moving of calculation to the phone side does not introduce any saving in terms of power consumption, it just shifts the power requirement to the output device. The advantages of this are that only one device needs frequent charging and if no display device is connected to the system while it is running, it reduces the power that is essentially wasted, as it does not produce any data that is observed.

As the phone side can be seen as the output device for the whole system as well for its own application, the design outlined here would work well on other possible hardware configurations. For example, if the data was being sent to a web service for more in depth monitoring, the same configuration would apply with only changes to the controller to allow for domain specific analysis. In the case where a simpler output device is to be used, such as an light emitting diode (LED) attached to a standard base station, a hybrid design of the phone side and SunSPOT side would be best. The amount of complexity in the view and controller would be determined by the devices capabilities.

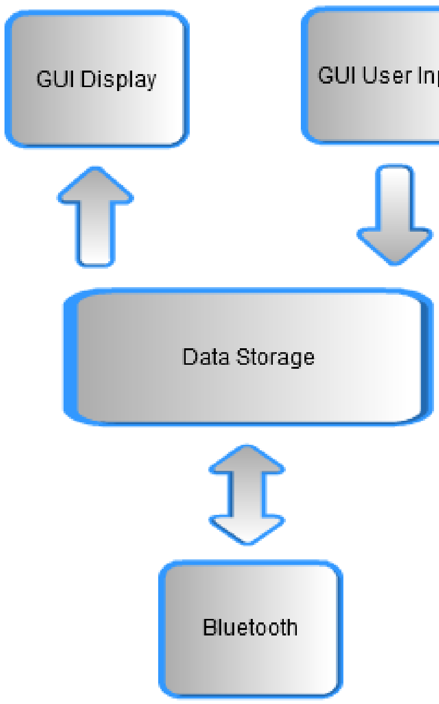


Figure 4.1: The functional architecture of the Android phone side application.

4.3.2 SunSPOT Side

The application that was run on the SunSPOT needed to provide the base station functionality. In the system the base station's required functionality was to connect environmental sensors as well as the deployed near contact units and stream this data to the output device. The above mentioned Model-View-Controller pattern could not be applied here as the base station does not have an output device in the conventional sense. While the Bluetooth communications interface can be seen as an output device, it relies on the availability of other devices that can not be guaranteed. For this reason a different architectural pattern was chosen. The whole system's architecture is broadly a pipeline with the base station providing the linkages between source and destination. The reduction of power consumption without incurring a cost in terms of accuracy was the main design consideration in the SunSPOT side architecture. To do this, a batch processing approach was taken. The batch cycle had several stages, the first of which was to gather data. This could be implemented as either polling available sensors or waiting for data to be reported, depending on the sensors. The second part of the cycle was to do some rudimentary processing on the data. This processing is limited to removing nonsensical values from the stream. This is only done at the base station, as it introduces no power overhead when compared to the power consumed in transmitting data that needs to be discarded later. The final task for the batch processor is to send the data. The main considerations in implementing this device is the amount of buffering and batch size that are necessary. The buffering is limited by the SunSPOT's memory, which is capable of holding approximately 1.7 days worth of readings at one reading per second. As this is not a limit that will be reached in any practical system, then limiting factor is timely reporting of data. As temperature is a slowly changing parameter sampling once every few seconds is sufficient to achieve an accurate measurement. A sleeping adult of average size emits 70 Watts (1 Watt equals 1 Joule per second) [2], scaled to an infant this is 15 Watts. The maximum possible rate of change therefore is 0.001633° (see appendix A for calculation). This translates to a reading needing to be taken at least once every 7 minutes. This log a period will cause the phone side display to appear unresponsive. For this reason a relatively small batch size was chosen. The smaller batch size does raise power consumption, as setting up and closing a connection is more power intensive than maintaining one for longer.

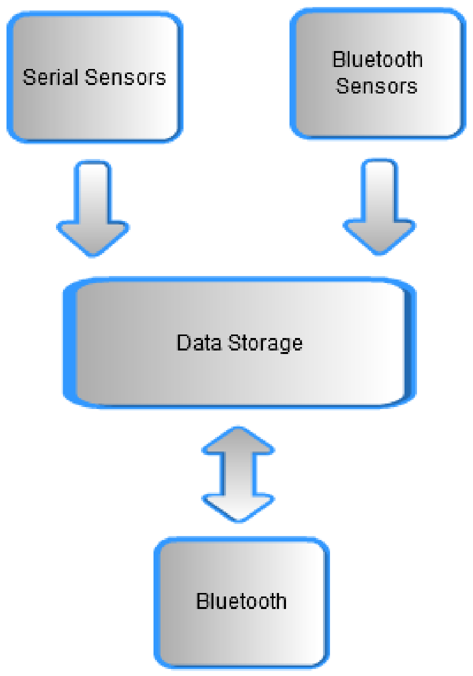


Figure 4.2: The functional architecture of the SunSPOT side (base station) application.

4.4 Technical Architecture

In this section the technical architecture will be discussed. It will focus on class structure as well as the motivations behind the chosen technologies and the design constraints they introduced. The most significant choice of technology was that used for communications, with the near contact units and local embedded sensors. It needed to provide enough bandwidth for reporting of data and needed to operate using only low power signals, as the near contact units would be close to infants' heads. Low power operation was also significant in ensuring accurate readings, as power use generates heat that must be dissipated from what is intended to be a very small device. According to the World Health Organization (WHO), the only health effect of long term exposure to electromagnetic waves is a raise in body temperature of 1° [45], this is only associated with significant field intensity found with industrial applications. However, most studies [46] have focused on adults in whom skulls are thicker offering increased protection over infants. Studies also focus on people who are exposed to radio wave through employment creating regular exposure patterns and, as few children are routinely exposed to such radiation future studies are unlikely. Rats' brains have been used as analogs in some studies. Salford et al. [47] found that when exposed to two hours of Global System for Mobile Communications (GSM) mobile phone electromagnetic fields there was a relationship between the number of damaged neurons and the output power of the transmitter normalised by the individual rat's bodyweight. There was no direct equivalent power output of a class two Bluetooth device's 2.5 milliwatt (such as the WT32). The lowest GSM power tested was a 10 milliwatt output, 4 times Bluetooth's power. Damage to neurons was rated on a scale from 0 - 2. Where 0 was none or occasional damaged neurons, 1 was moderate occurrence of damaged neurons and 2 was an abundance of damaged cells. Using linear interpolation (this is an appropriate method as the results show a linear progression) between 0 and 10 milliwatt outputs gives an average score of 0.26, this is only 0.15 higher than the no exposure score. Due to these reasons, and to increase confidence in any commercial implementation, Bluetooth was chosen for its low power signal. The other benefits of Bluetooth include small chip size reducing the physical size of devices and the fact that it is a widely implemented technology so it increases the chances of interoperability of this system with available environmental sensors.

While not implemented for reasons discussed above, the near contact unit's thermometer merits discussion. There are broadly two categories that thermal detection sensors fall into. The fist are called primary thermometers. They measure a property of matter and from this a temperature can be calculated. The property monitored, however, must be free from unknown variables. For example the speed of sound through a gas can give an accurate value for temperature only if the exact composition of the gas is known. This is done be calculating the Johnson-Nyquist noise (the noise generated by thermal disturbance of electrons in the gas). Temperature can also be derived from blackbody radiation. This is the amount of thermal radiation emitted from a surface. The other type of thermometers are secondary ones. They measure some temperature variant property such as the expansion of a metal. By themselves they can not give a value for temperature, unless they have been calibrated against a primary thermometer. A Mercury thermometer, for example, is unable to give a temperature reading for a given volume of Mercury unless a temperature dependent scale has been worked out experimentally and the quantity of a material present is known. That being, if a given volume of Mercury is due to a lesser quantity of hot Mercury or a greater quantity of cold Mercury. Mercury expands in the presence of heat.

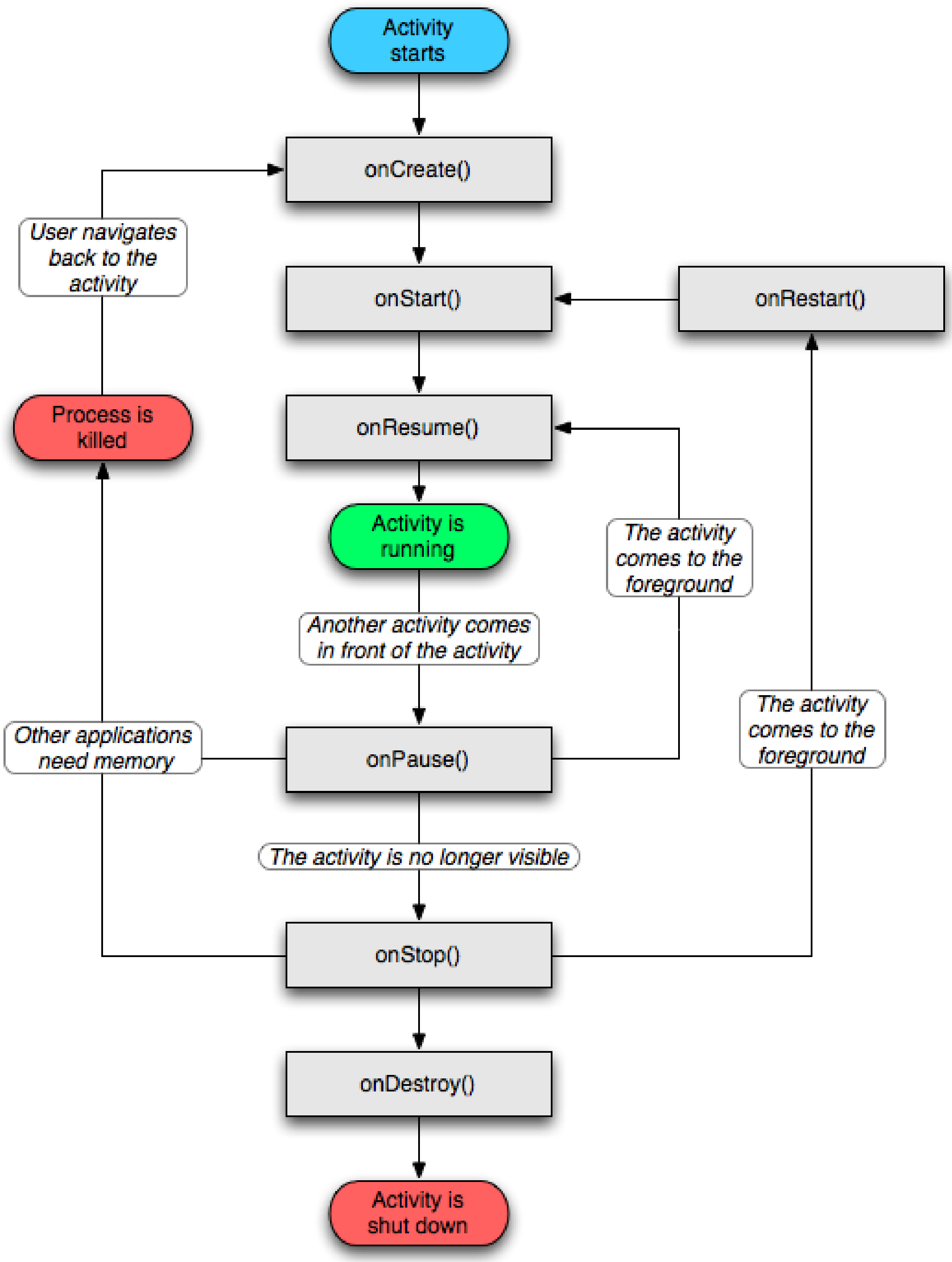
It is proposed to use an Infrared thermometer for the near contact units. These are devices that measure the Infrared light leaving a surface and from this infer the temperature. The major advantage of these is that they do not require contact to acquire temperature readings. Commercially there are three main types, those designed for use in industrial kitchens; those for metallurgy and smelting; and finally medical devices.

Kitchen thermometers typically operate in the required range for this application but lack the requisite accuracy, while metallurgical devices have the accuracy, they are very expensive and bulky. They also operate over a very large temperature range, most of which would be useless for human monitoring. Those designed for medical applications have the correct range and accuracy but have short distance-to-spot ratios (this is the ratio of distance from the source to area sampled). Short distance-to-spot ratios tend to mean that one must be close (typically 5-10 cm) to obtain an accurate reading. This is especially true of reflective surfaces. In the near contact units this range is workable if they were mounted on cots and prams, which would be a natural place to begin with. Alternately, the near contact units could utilise contact thermometers and be mounted on clothing to obtain data. In this deployment primary thermometers would be preferable as they tend not to contain potentially toxic chemicals. In the prototype system, the temperature is determined by the thermocouple in the WT32's battery. This is a thermometer that measures the voltage caused by a temperature differential across two metal conductors.

4.4.1 Phone Side

In the prototype system the output device was an Android Phone. The phone's technical architecture had several requirements; it must support Bluetooth; it must allow for display of dynamically changing data; the capability to do calculation on large data sets is also needed; and the operating system must allow applications to stay active when running in the background. Android version 2.1 or above was chosen as it met most of these requirements and the available devices had enough computational power to do the required calculations. The Android operating system does not support the plotting of data on an x-y scatter plot without overloading several standard user interface components. To save doing the significant work involved in this, a third part library was used. The AndroidPlot [16] was used as it added the graphing functionality to the standard operating system. When graphing dynamic data, the AndroidPlot library requires a class that converts data to be graphed into a form that it expects. This class constituted one part of the controller discussed earlier in section 4.3.1. This class implements a runnable, so can be passed to a thread to manage. The DataSource class provides for this. It is required by AndroidPlot to get new data from the source (in this case the base station via Bluetooth) and present this data to the DataSeries class. The DataSeries class provides a wrapper that makes dynamic data appear static to graphing utility, which is necessary as AndroidPlot was originally designed as a graphing tool for static data. The DataSeries class provides the same functionality as DataSource to a static graph. In either case, dynamic or static data, the draw method of the graphical object is called by the operating system, which in turn calls to DataSeries for its data to display.

Managing the life cycle of an application is particularly significant for this project as it must continue gathering sensor data even when the onPause method is called. This is the method used by the operating system to inform an application that it is being moved to the background (see the Android application lifecycle in table 4.3). For the purposes of the phone side application ,the onPause method must set any events that expect user interaction as suspended and it should also put the graphing utility to sleep. While this is not strictly necessary, it reduces the power that is used while the application is not visible. The onResume method, is in essence, the inverse of onPause. The operating system calls it when the application is moved to the foreground and its functionality should put the application back into its 'normal' mode of operation. The pairing of onStop and onRestart are similar but are used by the operating system when the application is moved to the background and nothing is replacing it in the foreground.



Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Figure 4.3: The lifecycle of an Android application [5]

4.4.2 SunSPOT Side

The base station's design was to provide for gathering data from several sources; collation and fusion of data; steaming of data to the output device; and for the purposes of the prototype, receive serial data. The SunSPOT was chosen for this as it has a programmable processor on-board and serial input is possible. This means it could interface with the WT32 which would be required for communication with environmental sensors. It also has the capability to use 802.15.4, which provides another potential technology to interface embedded sensors. The SunSPOT has several built in sensors that would also be useful for adding functionality. Its light sensor has the potential to reduce power consumption significantly when the system is used in a home environment. During daylight in a room with a window the light sensor reads about 700. If the window is covered by a curtain this drops to approximately 150 with lower readings if no natural light is available. Using this baseline the SunSPOT can be set to sleep if the light is too high. This is reasonable as if the room is too bright no infant will be sleeping there, so no temperature can be determined. The SunSPOT's temperature sensor gives much higher readings than that of the WT32 as it dissipates more power, causing a hotter battery. This is useful as it provides a maximum value for room temperature. If the assumption that the SunSPOT has been in the room long enough for its battery to acquire room temperature holds then the reading from the SunSPOT's battery thermocouple will be room temperature plus heat from battery usage, this must be greater than or equal to room temperature. With this fact, any environmental sensor that returns a room temperature reading greater that the SunSPOT's internal reading can be discounted as wrong.

As the SunSPOT does not have Bluetooth capabilities, a chip needed to be integrated into the base station. The WT32 was chosen as it supports RS-232 serial data. This meant that it could be used in conjunction with the SunSPOT's serial ports. The WT32's range of 30 meters was considered sufficient as the data from sensors out side of this range would most like be irrelevant to the system's monitoring. A smaller range also reduces the power requirement; some chips offer ranges of up to 100 meters but with much shorter battery life spans. The WT32 is also a very small chip which makes it ideal for the near contact units (see figure 4.4).

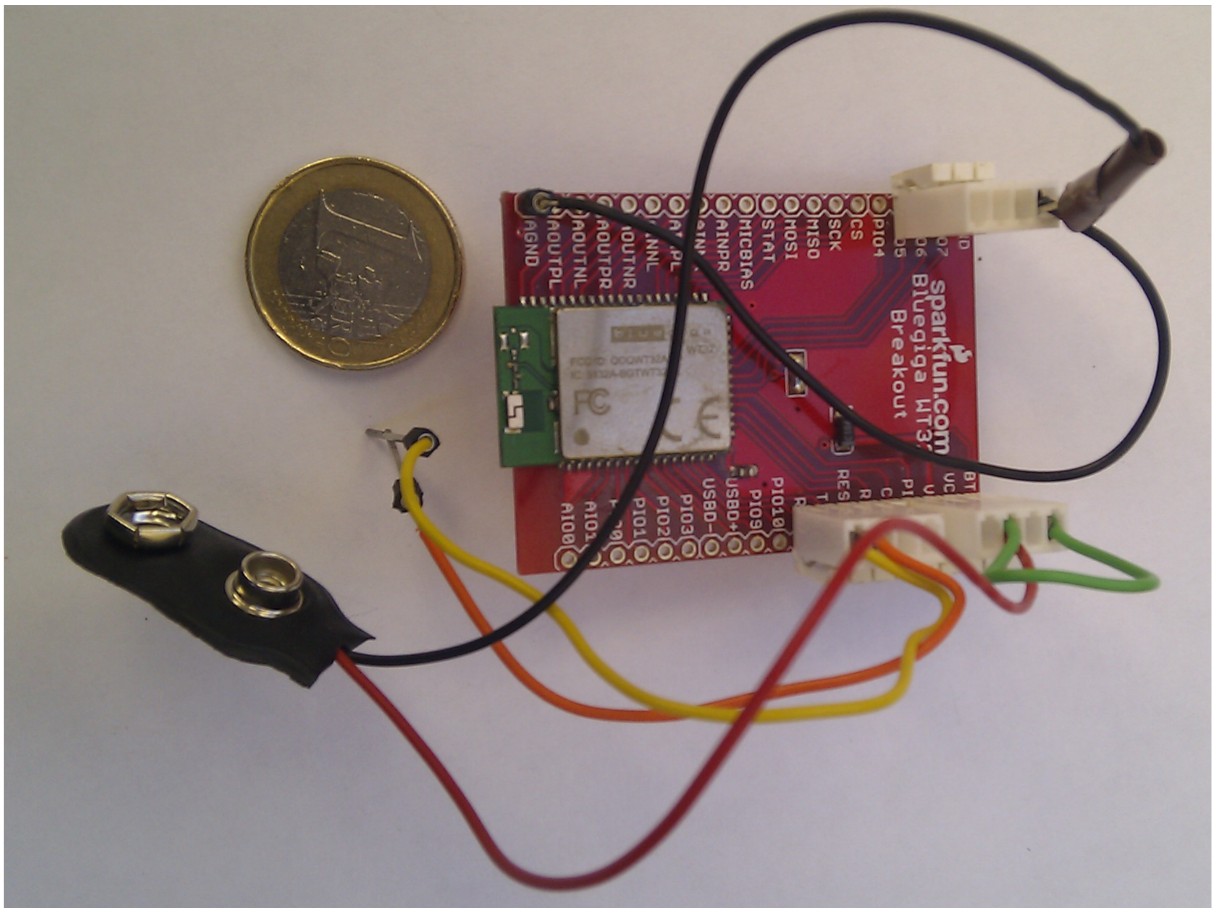


Figure 4.4: The silver chip is the WT32. It is mounted on a Sparkfun base board in this image.

The structure of the classes in the SunSPOT side application is relatively simple as most of the Bluetooth configuration and management is abstracted away from the main class (that extends MIDlet as required by SunSPOT applications).

Chapter 5

Implementation

5.1 Introduction

This section will provide a description of the implementation of the project. It will begin by providing an outline of what each device was required to do and how it was to operate. It will then discuss the way this was brought about with an outline of the configuration process for each device. Finally it will discuss problems encountered in the implementation.

5.2 WT32

The Bluetooth chip the WT32 was to allow for two way communication between the sensor via the SunSPOT and the Android phone. It was required to connect to the phone when the system was set up and forward any data that it was passed by the SunSPOT. It also needed to respond to instructions issued by the SunSPOT such as requests for reset of alterations to the configuration.

Before the WT32 could be used with the SunSPOT it needed some initial configuration, to allow it to understand commands passed to it by the SunSPOT. Once basic communication was established then the SunSPOT could alter the WT32's configuration as required. To achieve this the WT32 was connected to a USB to TTL cable so that it was accessible through a terminal program, in this case Free Serial Port Terminal [48]. This configuration was to set the baud rate (which by default is 115200 Bd) to one which the SunSPOT could handle. The SunSPOT's maximum rate is 38400 Bd. To reduce pressure on the internal buffers of the SunSPOT, this rate was set to 9600 Bd. The SunSPOT's buffer for a serial connection is only 64 bytes so data can be lost if this overflows before being moved to other storage.

```
SET CONTROL BAUD 9600,8N1
```

In the above command '9600' refers to the baud rate, '8' to the number of data bits, 'N' is for no parity bits and '1' is the number of stop bits used. The next step in configuring the WT32 was to change its operating mode. By default it runs in command mode in which it can receive and respond to ASCII commands. Once a connection is established in this mode it must change to data mode to transmit anything. In data mode it can not process commands. There is a delay in transitioning between the two modes. This delay can be as long as two seconds so this mode was not ideal for this project. It was instead used in the multiplexing mode. In this mode it can receive commands as well as manage data, however, the commands must be provided in a special hexadecimal frame (see frame structure table 5.1). The mode is changed with the following command.

```
SET CONTROL MUX 1
```

Here the '1' indicates that the mode is enabled. In multiplexing mode the WT32 can not accept ASCII strings for commands. So the command to change operational mode must be given in hexadecimal.

```
BF FF 00 11 53 45 54 20 43 4F 4E 54 52 4F 4C 20 4D 55 58 20 30 00
```

The above is the command including the frame to turn off multiplexing mode. It is the hexadecimal equivalent of the following command.

SET CONTROL MUX 0

With the exception that it can be processed in multiplexing mode.

Length	Name	Value
8 bits	Start of frame	0xBF
8 bits	Link ID	0x00 - 0x08 or 0xFF (control)
6 bits	Frame flags	0x00
10 bits	Size of data field in bytes	
0 - 800 bits	Data	
8 bits	Link ID xor 0xFF	

Table 5.1: The frame structure used by the WT32's mulitplexing mode.

The above commands complete the configuration of the WT32. It then could be connected to the SunSPOT. To connect to a device involves running a scan to detect other devices.

INQUIRY 1

With this command the number indicates the number of 1.28 second scans cycle that should be completed. It returns the number of devices as well as the MAC address for each of the discovered devices. This MAC address can then be used to connect to a specific device. The devices must first be paired for an RFCOMM channel (communications channel used by most Bluetooth applications) to be created. This is done using the pair instruction.

PAIR AB:CD:E0:12:34:56

This associates the device at the passed address with the WT32. The address must be issued in this form. It is 48 bits grouped into 6 bytes or octets. Each byte is written as two hexadecimal digits for human readability, the WT32 requires bytes to be separated by colons only. At this stage the phone side connection may request authentication via a code if the WT32 has been so configured.

```
CALL AB:CD:E0:12:34:56 1101 RFCOMM
```

This command is used to connect to any device, the device does not need to have been previously found with a scan. The MAC address should be provided in hexadecimal. The argument '1101' is a 16 bit universally unique identifier (UUID), in this case it indicates the connection should follow the serial port profile (SPP). Finally 'RFCOMM' indicates the connection type. Most applications use RFCOMM here but hands free kits for example would use 'HFP' to indicate a different connection type. With this sequence of instructions the devices will have a connection that will remain available while both are powered up and in range.

When using the WT32 in multiplexing mode to identify a frame as containing commands the control byte must be set to the hexadecimal value FF. If data is to be issued to the device any other two digit control byte can be used. Data can be written to the WT32 at any time but it will only be used if a connection is available. In this case the data will be written to the primary connection (this is only relevant if there are multiple connections, when one must be placed first in the list to indicate it is the primary). In getting to the stage where this configuration was possible, some issues were encountered. Firstly, the WT32 has a firmware called iWRAP, the boards used were previously at version 3. To remove any data from previous projects the board's firmware was reinstalled. Part way though, this reinstallation was interrupted by accidental disconnection of the USB to TTL cable. When restarted a newer version of the firmware was selected to install. This installation was constantly rejected by the device despite the documentation saying a partial installation should not block any subsequent one. When version 4 would not install a reversion back to 3 was tried which worked without issues. Disconnection of the cable does not corrupt the installation, it merely suspends it so the earlier version was able to resume the process.

Another issue encountered was that when in the multiplexing mode and using the the WT32 over the USB to TTL cable, the hexadecimal command to return to command mode was not interpreted as hexadecimal command but as a string. To work around this the SunSPOT had to be used to disengage multiplexing mode as its commands were framed correctly. The USB to TTL cable also required the connection of a ground pin for the signal to be detected. When using the serial data and the SunSPOT this was not necessary. This created additional complexity when getting familiar with the operation of the WT32 as sample code ran via the SunSPOT but not for the serial cable.

5.3 SunSPOT

The SunSPOT has two main functions in this project, the first was to provide an extension to the WT32. The WT32's processing power is comparatively limited so the SunSPOT manages the WT32, switching between commands and preparing the data for transmission. The SunSPOT's on-board sensors are also used in this project to provide additional accuracy to the readings taken. For example, if the SunSPOT's light meter's reading is high, a higher temperature can be expected from the WT32's temperature sensor. The final use of the SunSPOT was for purposes of debugging. It has superior memory to the WT32 so can log more data. The second function was as a base station at which environmental sensors could be managed.

Each program for a SunSPOT must have one class that extends MIDlet so that it complies with J2ME, which the SunSPOTs run. A project must provide start, pause and destroy methods as required by J2ME, however, the Java virtual machine used by the SunSPOTs does not call the pause method under any circumstance and the destroy method is only called if the start method throws a MIDletStateChangeException. The Start method is called upon the running of the application. It is used to set up any other required classes. For the proposes of this project it creates an instance of the

ManageBluetooth class (see figure 5.1). The ManageBluetooth class was written to handle communication with the WT32. It has two main functions, to read and to write.

Writing to the WT32 is done through an eDemoBoard (a virtualised instance of the SunSPOT's components). Its virtual UART (universal asynchronous receiver/transmitter) must first be initialised so that the baud rate is compatible with that of the WT32. Writing data is relatively simple as the data is pushed out and then forgotten about. A call to the writeUART(outputData) is all that is required.

The reading side of the connection, like writing, requires the initialisation of the eDemoBoard's UART. The eDemoBoard has several stock functions to read serial data from the SunSPOT's general purpose I/O pins. With these functions, by default the pin D0 is used for TX (the serial connection's transmit line) and D1 is used for RX (the connection's receive line). As all of the read functions are blocking they need to be surrounded by logic to prevent the code from hanging indefinitely when no data is present on the connection. Each read is protected by an if statement containing the availableUART function, this returns an estimate of the data available in the buffer. All functions that use the serial port are capable of throwing exceptions so try-catch blocks are provided. Finally this is surrounded by a while loop as the read function chosen (one that includes a timeout) only reads one byte at a time.

Periodically the SunSPOT checks for available environmental sensors using both Bluetooth and 802.15.4. If it finds one it then gets the available data and handles it. Currently only one type of sensor can be handled, that being external temperature. Upon receiving a reading for external temperature the SunSPOT compares this to the last reading transmitted to the phone, if there was one. If the reading is inside a range considered to be close to the last reading nothing is done, if the reading is outside this range then the new reading is sent to the phone so that the maximum rate of change can be calculated, the reading is sent over the normal Bluetooth connection with an additional identifier to differentiate it from the WT32's readings. This allows the alarm to be more accurate, as the levels at which it sounds can adapt to the environment. In an actual deployment this could go further and adjust the temperature in the room if required through interacting with a climate control system. Similar improvements to the system could be achieved using other possible sensors.

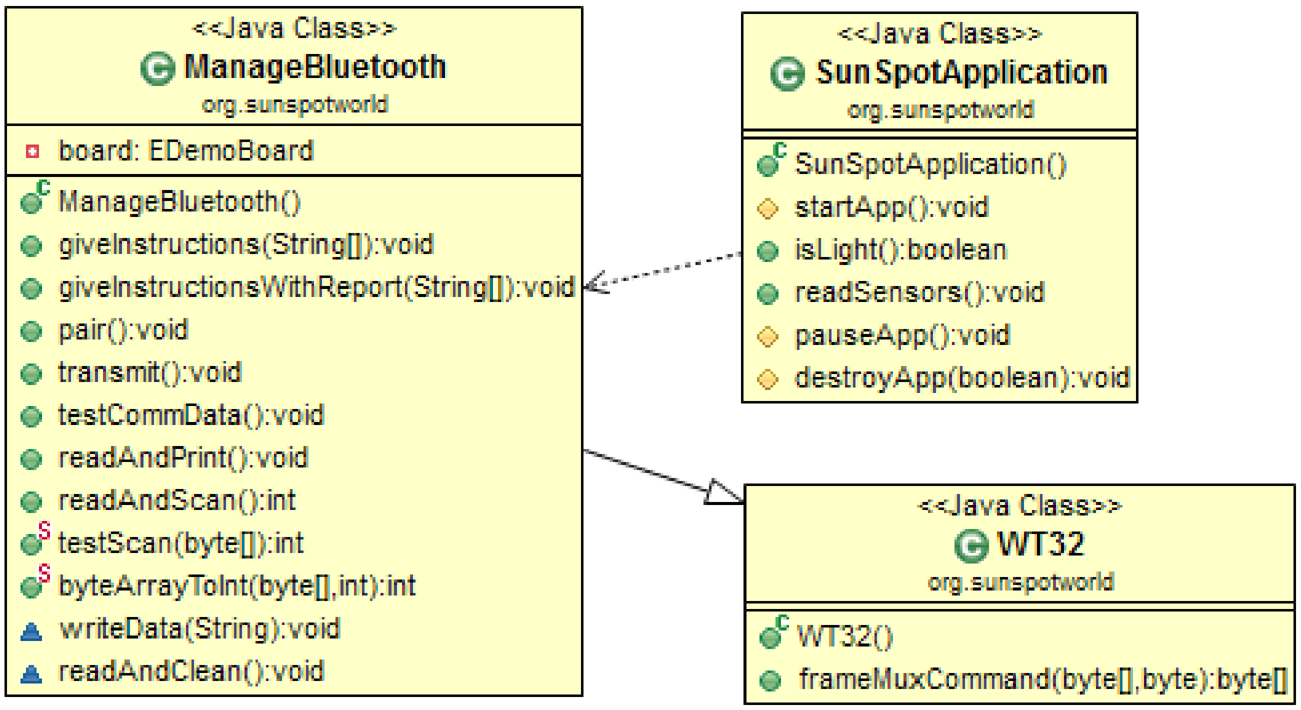


Figure 5.1: The class diagram of the SunSPOT side code.

The WT32 class is a helper class that has one method. It takes in a byte array and a control byte and returns another byte array that is correctly framed so that when given to the actual WT32, the returned array is processed as a multiplexing mode command.

When code is being sent to the SunSPOT under the Linux operating system, the hardware abstraction layer (HAL) is used to detect the connected devices. If the WT32 is connected to a SunSPOT's serial ports when the USB is plugged in the HAL will not detect the SunSPOT. This was a source of problems early in the project. It took time to understand why the error was occurring. As the Linux environment used was virtualised the problem was incorrectly assumed to be in the mapping between the real USB ports and the virtual ones.

5.4 Android Phone

The phone side application's main function was user input and output of the gathered data. As most of the low level data processing and validation was done on the SunSPOT, the phone needed only to break the bundled readings back into individual data points and display them. The phone was responsible for scanning through its data store and detecting successive values outside of a specified range. The connection management was also handled by the phone.

All applications developed for the Android operating system must provide at least one activity that is created with a new instance of the application. An activity is a section of code that executes a task and usually interacts with, the user. For example, the menu screen of an application could be one activity with each button spawning another activity. An activity must have an onCreate method which is called when the activity comes into scope much like a constructor method. There are a selection of other methods that can be overridden to manage the activity's life cycle. The initial activity in this project created a basic screen layout with a graph and some buttons, it also started a thread running.

This thread was an AndroidPlot DataSource. In the AndroidPlot library a graph is made from one or more DataSeries classes. The DataSeries (see figure 5.2) is used to get readings from the DataSource and provide information about it, such as number of readings. The DataSource class implements a Runnable. The run method has two main sections, the first handles the case of if a temperature sensor is connected, the second if no sensor is available. If there is a thermometer connected the method makes a call to the Thermometer class' getReading method. This returns a bundle of readings if any are available. The run method then breaks these up into individual values, timestamps them and adds them to the data store. The data store is implemented as a vector of class DataPoint, each DataPoint is a double representing the reading a UTC time since 1/1/70. The second part of the run method generates sample data. The sample data is a Gaussian curve around a fixed value with occasional spikes and simulated erroneous readings. This sample data is added to the data store. The DataSource class has a standard interface that it must provide to DataSeries class, getX and getY methods that take an index and return the values to display on the graph. In a static graph the index can be taken as an absolute value, that being index 1 gets value 1. In a dynamic graph the index must be treated as a relative position so that index 1 would be the first value that should be displayed. The determination of which value to display is made by the DataSource class. The search functionality is located in the DataSource class as it needs access to the full data store. It scans the data for any readings outside of a predefined range. To allow for measurement errors and corrupted values the data must be out of bounds for several consecutive readings before activating the alarm. The DataSource class can also shift the data it returns so that data can be reviewed off line. The alarm sound was played through the phone's music player which allowed for processing to continue while the sound was played without the need for a new thread. The operating system manages the music player once the correct methods are called.

The interaction with the WT32 and the SunSPOT and whatever sensors are connected thereto is handled by the Thermometer class. When the WT32 has established a Bluetooth connection and it is in the multiplexing mode as described previously, it forwards all data regardless of its content once the correct control byte is provided. When in multiplexing mode it should be able to handle both commands and data, but it interprets its escape sequence (the command that should terminate a Bluetooth connection) as data to transmit, which it duly transmits rather than closing the connection as it should. This is most likely due to the way the escape sequence is entered (as three single ASCII characters separated by at least one second). For most applications this would not be a problem as data can be streamed indefinitely once no further commands need to be issued. For this project though, this creates a problem as the WT32 can not be queried for its temperature reading while it is connected to the phone via Bluetooth. For this reason the Thermometer class' getReading method must create a new connection and read from then close it so the WT32 can be free to service other request from the SunSPOT. This is also the reason that the data is handled in batches. There is a power overhead in setting up and tearing down the connection for each reading.

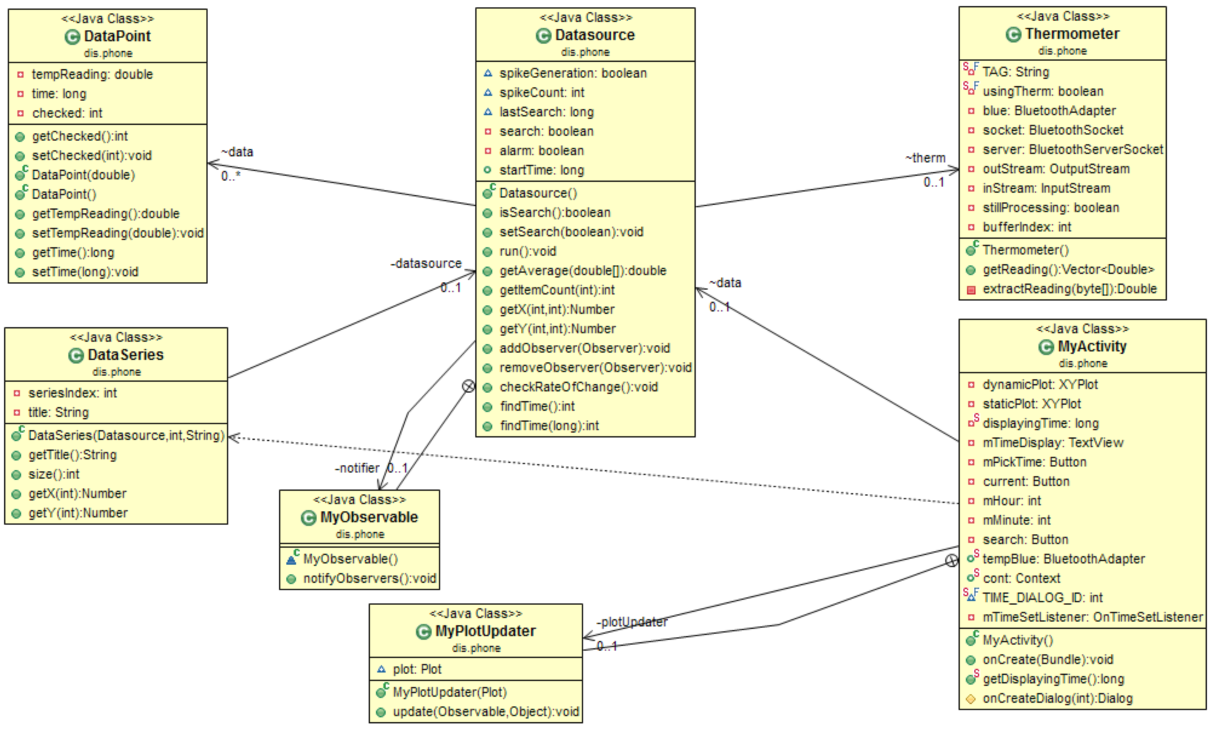


Figure 5.2: The class diagram of the phone side application.

The Eclipse plugins for Android provided significantly more information for debugging purposes that the other devices so it was easier to quickly isolate problems and find bugs. The most difficult aspect of programming the phone was learning to use the AndroidPlot library which provides very little support for dynamic graphing on the community website.

5.5 Bluetooth

The roll of Bluetooth in this project was to transmit and receive data reliably. Once both sides of the application, phone and SunSPOT, were at a sufficiently developed level, connecting the two devices and sending test strings was the first stage. The 'SET' command on the WT32 provides information about the devices's configuration, included in this is the hardware address used by the Bluetooth protocol to identify nodes. The corresponding address for the phone has to be obtained through the BluetoothAdapter class. The BluetoothAdapter must be initialised with getDefaultAdapter (other methods are available for devices with multiple Bluetooth chips). This method associates the BluetoothAdapter with the corresponding hardware. Once this is done the getAddress method can be called. The discovery of other devices can be done using scan methods, but, as the WT32 will get no user input, deciding which device to connect to is difficult. The necessary steps to establish a connection have been discussed previously. To send data from the WT32 side all that needs to be done is to pass the data to the WT32 and it automatically transmits. When receiving data the WT32 passes all data to the serial connection so that the host can do any processing. On the phone side the connection is more complicated. Once the BluetoothServerSocket has received and spun off a new connection as a BluetoothSocket, any input or output from the connection must be done through the corresponding Java.IO stream methods. Both input and output are initialised by getting their respective instances from the BluetoothSocket. When accessing the OutputStream, data is appended using the write method, which writes the data to the buffer but does not send it. If the data is to be sent and the connection kept open, the flush method is required. If the connection is no longer needed then the close method also transmits the data that has been buffered. For reading from the connection, a similar procedure to reading from the SunSPOT's serial connection is used. The available method returns an estimate of the quantity of data that has been buffered, if this is greater than zero then data can be read, otherwise the connection is closed so that the WT32 can continue with its work. The read method used differs from that used in the serial connection as it takes a byte array as an argument and writes any data to that. The method with an in-built timeout is not needed here, as the available function provides a better estimate when used with wireless communication. Due to the fact that there is less chance of noise from movement of serial pins. All the I/O methods require try and catch blocks as they can throw exceptions.

The major problem experienced in using the Bluetooth in this project was that if the WT32 is not given the 'PAIR' command prior to the 'CALL' command it will be unable to write data to the target device. With most devices the pairing and calling are done at the same time as they are used together at the application level. The WT32 does not do this, as doing so allows greater flexibility in its use. Once this was determined then the connection was bi-directional as expected.

5.6 Embedded Environmental Sensor

While the embedded environmental sensors were not part of the system at least one had to be implemented to test and demonstrate the functionality to discover and interact with available sensors. The sensor that was implemented was a room temperature sensor as its data was most relevant to the system. With the correct room temperature known, the rest of the system could calculate maximum and minimum possible rates of change.

A SunSPOT was used to provide the hardware for the sensor as it was capable of using an 802.15.4 radio to communicate, thereby enabling the testing of the base station's discovery functions and radio communications. The requirements for a SunSPOT application were similar to those discussed earlier. The sensor application only needs one class, the main class that extends MIDlet. The SunSPOT's radio has two basic modes, the first of which is the RadioStream which provides reliable, ordered data transmission analogous to the Transmission Control Protocol (TCP). It operates in much the same manner as

TCP in that it establishes a dedicated channel between two endpoints and the streams data into it. RadioGram is the other mode by which data can be sent. It is unordered and does not guarantee delivery much like User Datagram Protocol (UDP). Again the method that RadioGrams are used in is similar to that of UDP's datagrams. For this application the RadioGram was used, as the environmental sensor would have no way of knowing when a connection would be established and in a real deployment it would not use the power associated with keeping a server active.

To initialise a SunSPOT to use RadioGram based communication first an instance of the RadiogramConnection class must be created. This is done by casting the returned connection from a generic Connector's open method to the correct type. The open method takes a string representing the conection type and a port for the connection to run on.

radioGram = (RadiogramConnection) Connector.open(”radiogram://:” + Port);

Once the RadiogramConnection is set up, data can be sent. This is accomplished by creating a Datagram and writing the output data to it, then sending this Datagram. The creation of a Datagram can be done through the RadiogramConnection. There are several available methods to do this but the simplest was sufficient here, it takes one argument for the size of the DataGram. The data is then written to this through the DataGram's write method (the write method selected must represent the type of data to be written). Multiple Datagrams can be written to a RadiogramConnection between each call to the Send method, which obviously flushes the data. The environmental sensor is capable of reading packets from the base station and adjusting the generated value accordingly.

Chapter 6

Evaluation and Validation

6.1 Introduction

In this chapter the methods used to determine the accuracy and effectiveness of the system will be examined. All the experiments were based on the prototype system, that being the SunSPOT base station with the WT32 attached and functioning as a temperature sensor. As the WT32's temperature sensor will play a significant role in the experimentation, the first section will deal with validation of its effectiveness.

6.2 WT32 Accuracy and Responsiveness

The WT32's temperature sensor was designed to determine its battery temperature. It is required to prevent the unit from operating in conditions that could cause damage to either the battery or the Bluetooth circuitry. For the device as a whole [10] the operating temperature range $-30°$ - $80°$, it can survive storage to the range of $-40°$ - $80°$, however the battery should not be charged outside of a range from $0°$ - $60°$. As these ranges are significant to the reliability of the hardware it is reasonable to assume that the temperature sensor can operate in this range also, although the documentation does not specify. Even if the smallest range is taken as the temperature sensor's measurement range it should

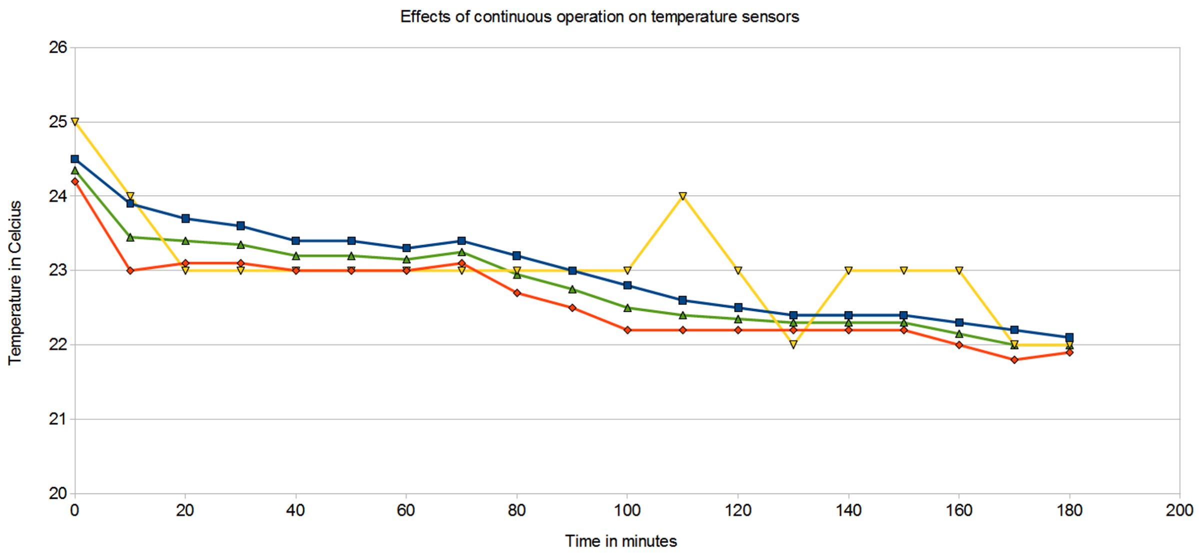


Figure 6.1: The results of the validation of the WT32's temperature sensor over time.

provide a range sufficient for measuring human temperature.

The first experiment was to determine the accuracy of the WT32's temperature sensor over time. As the sensor is connected to the WT32's battery it is possible that under constant operation the WT32 could produce enough heat to affect the reading. If this is, so the system would become less accurate as it ran. This experiment was conducted over three hours using room temperature as the data source. Three independent readings for temperature were taken every 10 minutes; one with a thermocouple type thermometer, one with a resistive thermometer and the WT32. Room temperature was used as the data source, as it was easier to obtain multiple readings of the room's temperature than that of a person's temperature on a small area of skin. The same area of skin would need to be used as temperature varies across the body due to the distance from the body's core and blood vessels. The results are as follows.

In the above graph the WT32's data is in yellow, the thermocouple is in red, the resistive thermometer is in blue and the green data is the average of the non-WT32 sources. The WT32's temperature sensor returns a rounded integer value, as opposed to the other

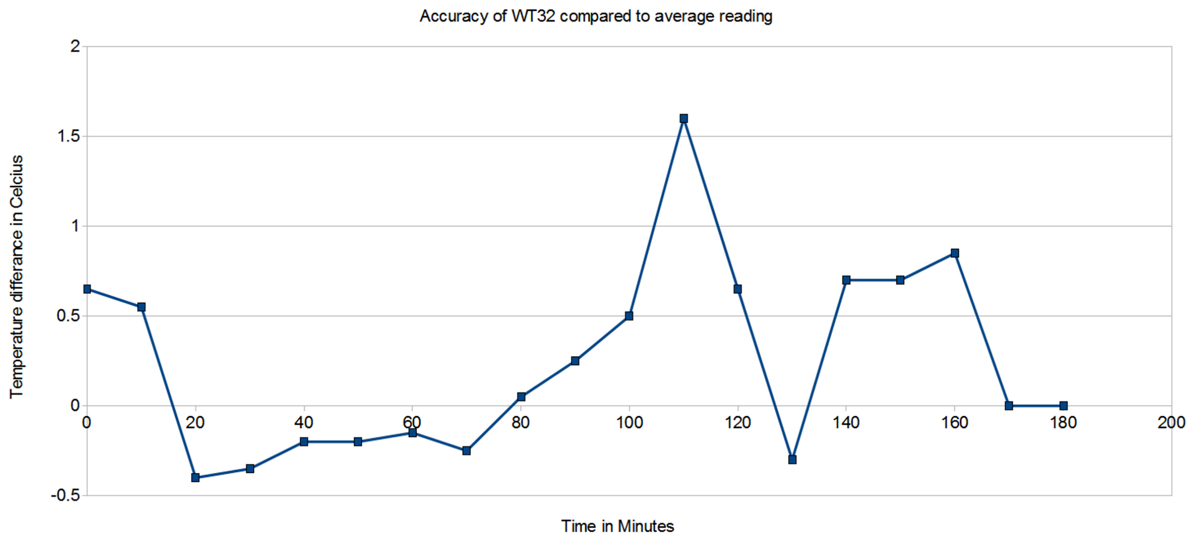


Figure 6.2: The difference between the WT32 and the average reading from the validating thermometer.

sensors used that provide accuracy to a tenth of a degree so disagreement by an average of 0.5° is acceptable as rounding error for this experiment. The legend provides the colour assigned to each data set, in green is the average reading from the validating thermometers. The WT32's yellow plot agrees with the other values with the acceptable error with the exception of the data point at 110 minutes. It differs by 1.6° from the average reading, this is most likely due to a slower response to the changing room temperature (gradually dropping since 80 minutes following the opening of a window), as the WT32's reading rises from slightly lower than the others to higher before dropping. The difference in rate of reaction to the change can be seen in the difference between the two validates also. The thermocouple which has very little covering on its temperature probe drops quickly in the interval from 80 to 100 minutes while the resistive thermometer takes a more gradual decline from 80 to 130 minutes. The WT32 takes approximately the same time to adjust as the resistive thermometer; both of these devices are mounted on Silicon boards. Silicon is a better thermal insulator than the metal around the thermocouple's probe which may explain the differential rate of change. Alternately, the spike at 110 minutes could just be quantisation error as it drops one degrees in each of the next two intervals.

The figure 6.2 shows the average reading from the validating temperature sensors subtracted from the WT32's reading. When this figure is averaged over all samples the average difference is 0.26°. This experiment shows that the WT32 stays accurate to within the expected variance for approximately 70% of the time, which should be acceptable for the prototype system.

The second validation experiment for the WT32's temperature sensing capabilities was a determination of its response time to a new thermal source. The basic set up for this experiment involved applying a heat source to both the thermocouple (the same one used above) and the WT32 and observing the response. Once both had reached a stable output, the heat source was removed to determine the speed at which the sensors cooled. The resistive thermometer was not used in this test as it was used to monitor the ambient temperature. This was required, as the end point of the experiment would be when both thermometers had returned to room temperature, so the base line temperature needed to be measured. With this set up, it was also possible to ensure that the ambient temperature remained constant throughout. Finding an adequate heat source proved problematic. Ideally a heat source for this type of experiment would be able to produce an output in the required range that is stable, uniform and programmable. This would give an accurate mid-point to the experiment, as when the thermometers read the source's programed value they could be removed and cooling could begin. As no such heat source was available a person was used instead. This method provided the a value in the correct range and as the response to the temperature was expected to be reasonably rapid (about 10 minutes); there would be little fluctuation in this value with a person at rest. Uniformity was the main issue, as one sensor was to be held in each hand but constant temperature could not be guaranteed, so the thermometers where switched every minute so that the average temperature they were exposed to over the period of the experiment would by approximately even. The readings for temperature were taken every 30 seconds and recorded. The room temperature remained constant with in $\pm 0.1°$ of 23°. This means

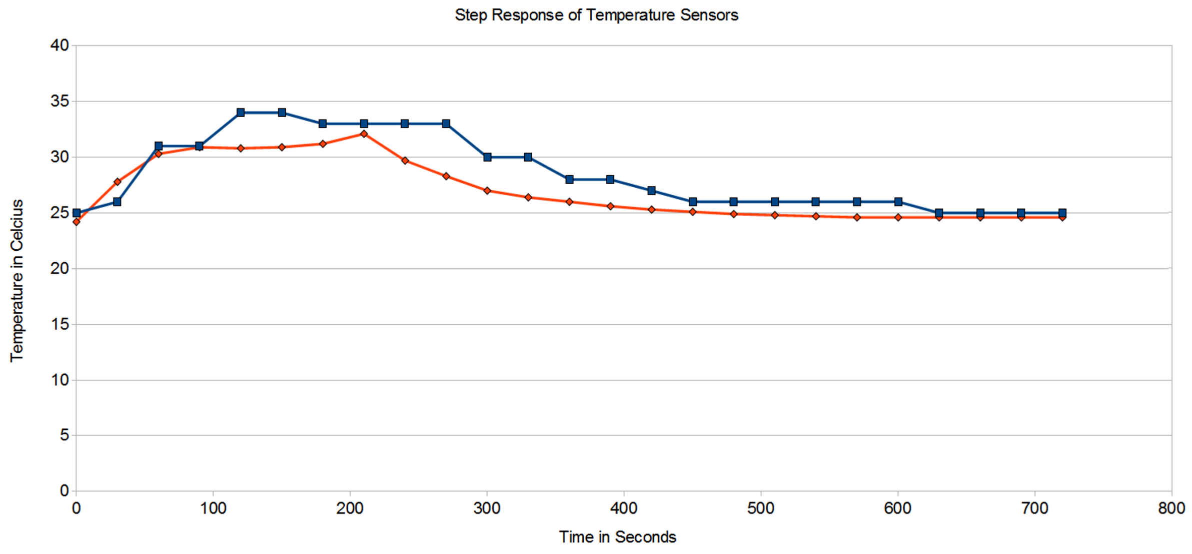


Figure 6.3: The results of the step response of the WT32's temperature sensor and the thermocouple over time.

any change in temperature read by the thermometers was due to the heat source not background changes.

Essentially this experiment aimed to determine the time for the step response of the thermometer to reach a stable output value. In the above graph the WT32's data is in blue and the thermocouple's is in red. For this experiment that temperature was found to be 33° on the WT32 and 32.1° on the thermocouple, as at this point both values agreed to the expected accuracy and were remaining stable. In the first four measurements on the graph 6.2 both thermometers average an increase in temperature of 0.066° per second. The WT32 continues this rate of change until it reaches its stable value of 33° - 34° (true value most likely between the two and quantisation causes the transition seen at the time of 180 seconds). The thermocouple's rate drops significantly as it approaches the data source's temperature, eventually reaching its final value at a time of 210 seconds. In this part of the experiment all the heat is transfered by conduction (the method of heat transfer between connected regions of matter with differing temperatures) and can be described by the law of heat conduction (Fourier's law), which states that heat flux density is equal to a product of thermal conductivity and the negative local temperature gradient. Heat flux density is the amount of energy that flows through a given area per unit time. Formally this can be written as

$$q^{\rightarrow} = -k\nabla T$$

where $q^{\rightarrow}$ is the heat flux, k is the materials conductivity and T is the temperature gradient. In the case of the thermometers this is made more complicated, as they are composed of several materials that conduct heat at different rates so it is not practical to calculate the rate of heat transfer.

As no programmable heat source was available, the only responses that could be measured with any accuracy were those of heat applied and heat not applied. It would be interesting to see the response times to smaller fluctuations in temperature but this was not possible so the cooling response was taken. After the heat stabilized at 120 seconds the sensors were removed from the data source and the second part of this experiment began. The thermocouple responds more rapidly to the change in temperature, as discussed earlier, this is due to the surrounding materials, in particular the WT32's Silicon base. Each takes roughly 300 seconds to reach a stable value once they begin responding to the change. From the point of removal both showed similar cooling profiles with a decaying rate as predicted by the underlining physics (once again allowing for noise introduced by WT32's quantisation effects). Newtons law of heat transfer can be used to explain the smoothness of the cooling curves. It is analogous to Fourier's law, except that describes convection cooling (that being the way heat is transfered through a fluid due to diffusion). The law states that the rate of temperature change for a body is proportional to the temperature difference between object and surroundings. Formally this is

$$\nabla Q = mC_p\nabla T$$

Where Q is the amount of thermal energy, m is the mass, C_p is the specific heat capacity (the amount of energy required to raise the temperature of a given amount of substance by a given temperature, usually in Joules per Gram per degree Kelvin, it varies with temperature) of the substance and ∇T represents the time variant temperature gradient. As the formula shows the smaller the temperature gradient, the smaller the loss of thermal energy. This formula can only be applied to situations where thermal conductivity is constant. This is rarely true but as the change in temperature is so small it is a good approximation.

From these experiments, it is clear that the WT32 is reasonably accurate and responds in a comparable time to other thermometers, making it sufficient for the prototype system. However, the lack of resolution causes problems in accurately monitoring a stable value. This can be seen in both experiments, where when monitoring a constant value the reading fluctuates due to quantisation. The other major design consideration that these experiments have highlighted is the housing for the sensor. On the WT32, one side of the sensor is backed in Silicon and the other is exposed metal. Not only does this cause problems in cooling but it makes the device orientation dependent. If the Silicon side is placed on the temperature source both heating and cooling times will be significantly slower as it insulates the sensor from the source when heating and only emits heat slowly when cooling (this also accounts for the slightly longer cooling time than the thermocouple seen in the second experiment). In the actual system, the temperature probe would need to be as exposed as possible to the heat source to ensure accuracy or at the very least it should be surrounded by a uniform material so that it responds evenly in all directions. Ideally the sensor would not be mounted on the board with the Bluetooth chip and power source, it would be attached instead via a cable like the probe on the thermocouple used for validation (see figure 6.4). This would provide greater flexibility and would also protect against dissipated heat causing inaccuracy if a different chip was used. The backing material issue would not occur is an infrared thermometer was used.

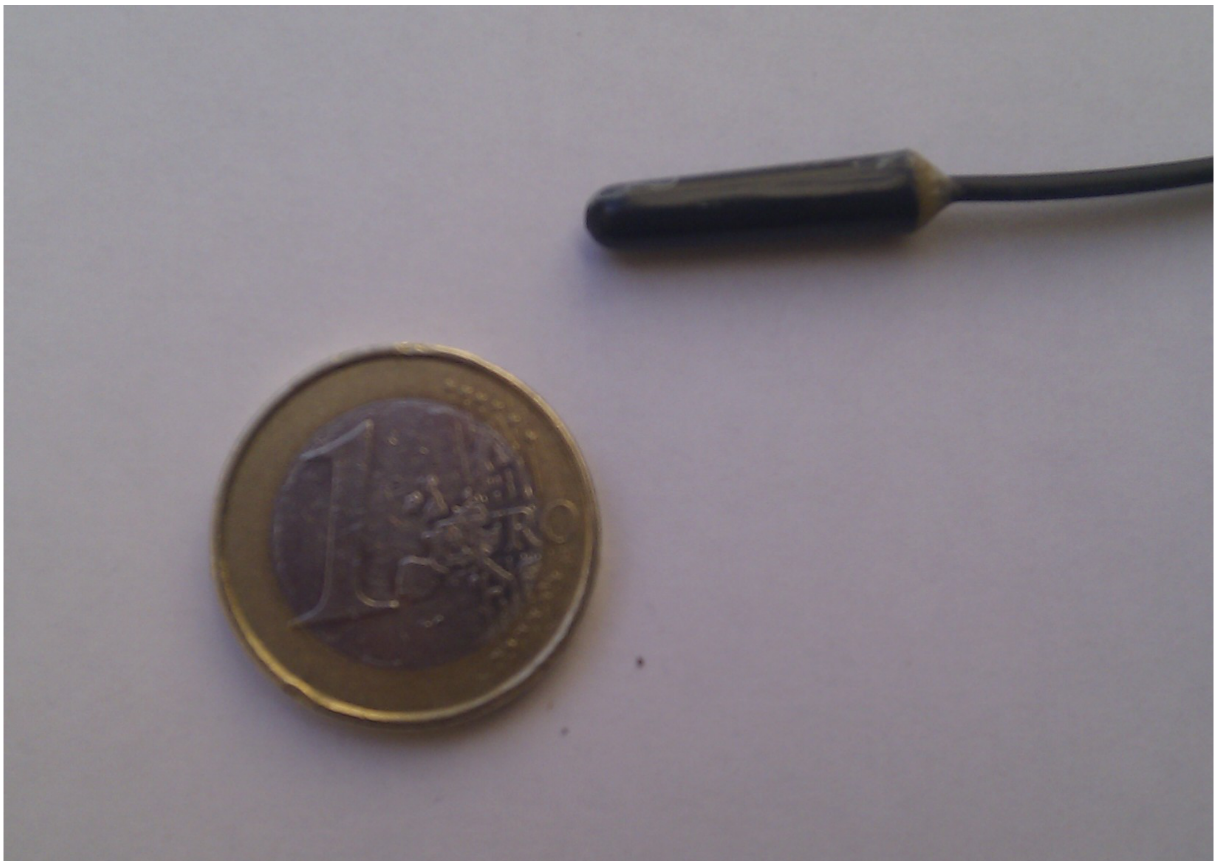


Figure 6.4: The thermocouple used in validation.

6.3 Overall System

Testing the effectiveness of the whole system was difficult as realistic temperature profiles were not available to input to the system, so representative ones had to be used. The data set used for testing had Gaussian distribution around an average value for human body temperature. Occasionally erroneous readings were generated to simulate malfunctions. The test data also had periodic spikes to simulate periods of overheating. When testing the system with this data rate of change had to be discounted, as the data was generated much faster than a human body would. The experiment had to done this way as it was not feasible to run the experiment over a period of 6 - 8 hours (the approximate amount of time that would be needed for normal human body to produce a dangerous temperate change from a normal range). The generated data was run for 4 minutes, giving a new reading every 2 seconds. This represents 7 hours of human generated data at 1 sample every 210 seconds. In this experiment there were two periods of overheating to detect. Both were correctly detected, but there were 11 false positives, where the alarm was activated due to consecutive errors. Errors were generated randomly with a probability of 0.02, all errors were considered independent. To prevent sounding the alarm for every error two consecutive reading needed to be out of range. This gave a probability of 0.0004 of any two consecutive readings causing an alarm due to error. The experiment had 25200 data points, so there should be an average of 10.08 false positives per experiment. This showed that the alarm conditions were too sensitive. The number of consecutive errors for alarm activation was changed to 4. This translated as an expected false positive rate of 0.004032. This is approximately one false alarm per week based on constant monitoring. There should be no false negatives with this value due to the slow rate of change of temperature, 4 consecutive readings represents 14 minutes at the system's normal rate.

Chapter 7

Future Work

7.1 Introduction

The future work on this project can go in several different directions depending on how the system is to be used. For this reason, this chapter is split into two main parts. The first is the Further Research section. It will discuss the possibilities of extending the system and adding functionality to it to improve performance. The other section is Commercialisation and it will focus on the issues associated with converting the system in to a viable product.

7.2 Further Research

Currently the system is based solely on temperature, and while this is can be a reasonable predictor of infections and other conditions, the accuracy of the system could be improved by the inclusion of other sensors. To remain useful in a home environment the sensors used must not require complex calibration or be invasive. There are several possible candidates. Systems that use Doppler radar to determine heart and respiration rates [31] have been shown to produce accurate results. This would significantly expand the range of conditions that could be detected. Various chemical sensors could also be integrated.

Carbon Monoxide (CO) for example is easily detectable in the atmosphere and children are particularly sensitive to its effects. As it is colourless and odorless and can be produced by common heaters it can be very dangerous. If chemical sensor were interfaced with this system it is conceivable that upon detecting high levels of CO the system could shut down the heater to prevent further exposure.

For dynamic discovery of sensors and actuators in an environment, an advertising protocol would need to be developed. The protocol would need to allow a sensor to advertise its capabilities (accuracy, range, remaining power etc.), it would also need to be light weight so that even functionally limited devices could advertise. One of the most difficult aspects of such a protocol would be in determining how to describe data sources in an environment. For example if two light meters were both in range of the proposed system how would the system determine which reading was more relevant. It may be possible to obtain a measure of how close the two sensors are based on tier signal strength but the closest sensor may not be the most relevant.

As not all sensors that could exist in a deployment scenario will provide the same quality of service to the system, some objective measure of confidence in the systems out put would be needed. For example, if the system was producing output based on its own sensors and one environmental sensor, then the embedded sensor ran out of power, the data produced by the system would be less reliable. The reliability measure developed would need to take account of how likely the system is to miss a significant event for a given configuration. This measure would also be useful in deciding weather to interface a newly discovered sensor. By comparing the current configuration's reliability with that of the new one the correct decision can be taken.

Adapting the system to other applications would also be possible. Monitoring the temperature of waste water from power plant could be a possible application. Power plants have to comply with regulation regarding the temperature differential between the water they output and the body of water they discharge into. Such monitoring is currently done by internal sensor in the plant and at a wide scale by satellites [49]. The readings from the plants sensors could be verified by this system mounted out the outflow pipe.

7.3 Commercialisation

If the system was to be commercialised, it would need to be tested using near contact units rather than the WT32's temperature sensor. With the new configuration further investigation would be necessary. Which ever thermometer was chosen for the near contact units would need to be validated in much the same way as the WT32 was. If multiple near contact units were allowed for each base station, then a medium access control system would be needed. This would not be necessary for any environmental sensors that use 802.15.4, as the SunsPOT's buffer can handle many received packets at a time. Depending on the implementation of the system, additional work may be needed on load balancing or power saving, as, if the near contact unit's battery runs out, the system looses all functionality so any remaining power at the base station is essentially wasted. The system would also need more accurate profiles for alarm activation than are currently used.

Chapter 8

Conclusion

The TotTemp system presented in this report aims to provide real time medical data on an infant. To do this, a design is proposed that uses three devices connected over Bluetooth. The system also has the capability to interact with other sensors and actuators that may be present in the environment. Using these embedded sensors it can improve the accuracy and relevancy of its results. The environmental actuator can be used to provide automated control of the infants room. For example, if the infant is over heating then an air conditioner could be activated.

To demonstrate that such a system could be effective, a prototype was constructed. Due to limits imposed by time and hardware availability the system was implemented using only two devices. With the sensor functionality and the base station being combined. The system was able to produce a data stream from the on-board temperature sensor enhanced by the environment's embedded sensors which was relayed to the phone, where it was displayed and analysed. If a dangerous condition was detected an alarm was sounded and remedial action taken.

The prototype system demonstrated that remote monitoring of temperature can be achieved while interfacing with devices already present in the environment. While the accuracy of the alarm conditions can not be tested directly, they proved accurate in simulations. Temperature alone is unlikely to provide sufficient data for an accurate diagnosis of a specific condition. The radar based system proposed by Yan et al. [31] could provide information more relevant to diagnosis but would not be able to provide forewarning of an event as temperature can. To improve the accuracy and performance of any such system the more data generated the better the results so, for this reason, a flexible and effective system will monitor several biometric parameters.

Appendix A

Calculations for Human Temperature Change

The maximum possible temperature change for a period of time can be defined as the maximum amount of heat energy emitted, multiplied by the specific heat capacity (the amount of energy required to raise the temperature of a given amount of substance by a given temperature, usually in Joules per Gram per degree Kelvin, it varies with temperature) of the human body. More formally this is as follows.

$$\Delta T = \Delta Q / C_p$$

Where T is the change in temperature, Q is the heat added to the system and C_p is the specific heat capacity. The value of Q for a sleeping child is difficult to determine as there is no practical reason to know it for most applications so it must be extrapolated from that of an adult (which is used when designing heating and cooling systems). The data in table A.1 is based on that of an average sized adult of 70 kg. The extrapolated values for infants are set at 5% and 30% of this figure to give either end of the possible range of values for heat production.

Method of Heat Generation	Amount
Adult - Sleeping	70 Watts
Adult - Sitting	100 Watts
Adult - Walking	200 Watts
Adult - Exercising	300 - 870 Watts
Infant Low - Sleeping	3 Watts
Infant Low - Sitting	5 Watts
Infant High - Sleeping	20 Watts
Infant High - Sitting	29 Watts

Table A.1: Human thermal energy output [2]

The human body's specific heat capacity is also somewhat complex. The exact figure will vary around the body as skin and flesh have different capacities. The specific heat capacity of skin is 3600 Joules per kg per degree Celsius [50]. The average for the rest of the body is 3470 Joules per kg per degree Celsius. Assuming the heat is dissipated evenly across the surface, the calculations are as follows.

Q in Watts	Heat Capacity in J \ (kg * °C)	Weight in kg	ΔT in °C \ second
3	3600	3	0.000277
3	3600	20	0.000041
3	3470	3	0.000288
3	3470	20	0.000043
5	3600	3	0.000462
5	3600	20	0.000069
5	3470	3	0.00048
5	3470	20	0.000072
20	3600	3	0.00185
20	3600	20	0.00027
20	3470	3	0.001921
20	3470	20	0.000288
29	3600	3	0.002685
29	3600	20	0.000402
29	3470	3	0.002785
29	3470	20	0.000417

Table A.2: Possible rates of change for given parameters using the formula $\Delta T = \Delta Q/C_p$

The table A.2 shows how the various parameters affect the calculation for rate of temperature change. The values for ΔT are for the temperature change in the human body and could only be observed using an ideal thermometer (that being one with perfect accuracy and instantaneous response). The lack of an ideal thermometer introduces an additional risk to the system, as the time to heat the thermometer will tend to smooth out the profile of temperature change generated by the body. For this reason the maximum permissible rate should be set below the maximum possible rate to negate this effect. The maximum possible change was found with the parameters of 29 Joules of heat generated and measured directly on flesh (removing the time for heat to move to the skin) on a 3 kg infant. In this circumstance, the temperature rose by 0.002785° per second. At this rate it would take 3 minutes to rise 0.5°, the maximum change that will only advance the WT32's reading by 1°. This calculation takes no account of the cooling effect of the air in the room.

The cooling effect can be modeled using Newtons law of heat transfer. It describes convection cooling (that being the way heat is transfered through a fluid, in this case air, due to diffusion). The law states that the rate of temperature change for a body is proportional to the temperature difference between object and surroundings. Formally this is

$$\nabla Q = mC_p \nabla T$$

Where Q is the amount of thermal energy, m is the mass, C_p is the specific heat capacity of the substance and ∇T represents the time variant temperature gradient. As the formula shows, the smaller the temperature gradient, the smaller the loss of thermal energy. This formula can only be applied to situations where thermal conductivity is constant; this is rarely true but as the change in temperature is so small it is a good approximation. This form of the law can only be used for forced cooling; that being cooling by circulating air as would typically be found in a room that is not sealed. There would be no cooling effect if the room was at the same temperature as the body or if the room is hotter, but as this is monitored by the system these cases will be discount . The calculation for the rate of Newtonian cooling for the maximum possible heat producing body mentioned above in a room at 25° assuming it started at 37°, yields a Q of 12 Watts. When this is applied to the same body it converts to 0.001152° per second of cooling, making the maximum possible heating rate achievable at room temperature 0.001633°. Converting this into a reading frequency for the WT32, it is at least once every 7 minutes to ensure that each individual degree rise in temperature is sampled. This rate of change can only be applied to resting infants once the thermometer has reached a stable reading of their body temperature.

Bibliography

[1] M. Weiser, "The computer for the twenty-first century," in *Scientific American*, pp. 94 – 100, September 1991.

[2] C. Binggeli, *Building Systems for Interior Designers.* John Wiley and Sons, 2009.

[3] H. Gray, *Henry Gray's Anatomy of the Human Body.* 1918.

[4] H. Chan, "Comparing wireless data network standards," in *AFRICON 2007*, pp. 1 – 15, September 2007.

[5] Google, "Android developers home," July 2011. Accessed on 26/8/11.

[6] Y. Axelrod and M. Diringer, "Temperature management in acute neurologic disorders," in *Neurologic Critical Care*, vol. Volume 26, Issue 2, pp. 585 – 603, June 2008.

[7] K. B. Laupland, "Fever in the critically ill medical patient," in *Critical Care Medicine*, vol. Volume 37, Issue 7, pp. S273 – S278, July 2009.

[8] K. Agarwal, A. Lange, and H. Beck, "Thermal imaging in healthy humans what is normal skin temperature?," in *InfraMation Proceedings*, May 2007.

[9] Oracle Labs, "Sun SPOT World," July 2011. Accessed on 26/8/11.

[10] Bluegiga Technologies, "WT32 Data Sheet," July 2011. Accessed on 26/8/11.

[11] Bluegiga Technologies, "iWRAP User's Guide," July 2011. Accessed on 26/8/11.

[12] Sparkfun Electronics, "Sparkfun Electronics Home Page," July 2011. Accessed on 26/8/11.

[13] Future Technology Devices International Ltd, "USB to RS-232 Cable Data Sheet," July 2011. Accessed on 26/8/11.

[14] Jennic, "Co-existence of IEEE 802.15.4 at 2.4 GHz Application Note," vol. 1.0, February 2008.

[15] A. Sikora and V. Groza, "Coexistence of IEEE 802.15.4 with other Systems in the 2.4 GHz-ISM-Band," in *Instrumentation and Measurement Technology Conference, 2005. IMTC 2005. Proceedings of the IEEE*, vol. 3, pp. 1786 – 1791, May 2005.

[16] Android Plot Community, "Android plot," July 2011. Accessed on 26/8/11.

[17] H. Chen, C. Tse, and J. Feng, "Impact of topology on performance and energy efficiency in wireless sensor networks for source extraction," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, pp. 886 – 897, June 2009.

[18] C.-T. Cheng, C. K. Tse, and F. C. M. Lau, "A delay-aware data collection network structure for wireless sensor networks," *Sensors Journal, IEEE*, vol. 11, pp. 699 – 710, March 2011.

[19] T. Sivanadyan and A. Sayeed, "Active wireless sensing: A versatile framework for information retrieval in sensor networks," *Signal Processing, IEEE Transactions on*, vol. 57, pp. 1588 – 1603, April 2009.

[20] X. Zhang, H. Jiang, L. Zhang, C. Zhang, Z. Wang, and X. Chen, "An Energy-Efficient ASIC for Wireless Body Sensor Networks in Medical Applications," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 4, pp. 11 – 18, February 2010.

[21] E. Freudenthal, D. Herrera, F. Kautz, C. Natividad, A. Ogrey, J. Sipla, A. Sosa, C. Betancourt, and L. Estevez, "Suitability of nfc for medical device communication

and power delivery," in *Engineering in Medicine and Biology Workshop, 2007 IEEE Dallas*, pp. 51 – 54, November 2007.

[22] J. Smith, A. Sample, S. Powledge, P.and Roy, and A. Mamishev, "A wirelessly-powered platform for sensing and computation," in *UBICOMP 2006, Springer*, pp. 495 – 506, 2006.

[23] R. Agarwal, R. Martinez-Catala, S. Harte, C. Segard, and B. O'Flynn, "Modeling power in multi-functionality sensor network applications," in *Sensor Technologies and Applications, 2008. SENSORCOMM '08. Second International Conference on*, pp. 507 – 512, August 2008.

[24] J. Jin, A. Sridharan, B. Krishnamachari, and M. Palaniswami, "Handling inelastic traffic in wireless sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 28, pp. 1105 – 1115, September 2010.

[25] C.-T. Cheng, C. Tse, and F. Lau, "An energy-aware scheduling scheme for wireless sensor networks," *Vehicular Technology, IEEE Transactions on*, vol. 59, pp. 3427 – 3444, September 2010.

[26] A. Goutham, B. Manjunath Iyer, and R. Rajiv, "Patient Data Management System ; Periodic Model using GSM," in *Digital Information Management (ICDIM), 2010 Fifth International Conference on*, pp. 519 – 524, July 2010.

[27] S. Cheng, K. Tom, L. Thomas, and M. Pecht, "A wireless sensor system for prognostics and health management," *Sensors Journal, IEEE*, vol. 10, pp. 856 – 862, April 2010.

[28] F. Kocer and M. Flynn, "An RF-powered, wireless CMOS temperature sensor," *Sensors Journal, IEEE*, vol. 6, pp. 557 – 564, June 2006.

[29] L. Reindl, I. Shrena, S. Kenshil, and R. Peter, "Wireless measurement of temperature using surface acoustic waves sensors," in *Frequency Control Symposium and*

PDA Exhibition Jointly with the 17th European Frequency and Time Forum, 2003. Proceedings of the 2003 IEEE International, pp. 935 – 941, May 2003.

[30] H. Cao, L.-C. Hsu, T. Ativanichayaphong, J. Sin, and J.-C. Chiao, “A non-invasive and remote infant monitoring system using CO2 sensors,” in *Sensors, 2007 IEEE*, pp. 989 – 992, October 2007.

[31] Y. Yan, C. Li, X. Yu, M. Weiss, and J. Lin, “Verification of a non-contact vital sign monitoring system using an infant simulator,” in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp. 4836 – 4839, September 2009.

[32] S.-L. Chen, H.-Y. Lee, C.-A. Chen, H.-Y. Huang, and C.-H. Luo, “Wireless body sensor network with adaptive low-power design for biometrics and healthcare applications,” *Systems Journal, IEEE*, vol. 3, pp. 398 – 409, December 2009.

[33] J. Corchado, J. Bajo, D. Tapia, and A. Abraham, “Using heterogeneous wireless sensor networks in a telemonitoring system for healthcare,” *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, pp. 234 – 240, March 2010.

[34] M. Rabbi, M. Rahman, M. Uddin, and G. Salehin, “An efficient wireless mesh network: A new architecture,” in *Communication Technology, 2006. ICCT ’06. International Conference on*, pp. 1 – 5, November 2006.

[35] M. Lee, R. Zhang, J. Zheng, G.-S. Ahn, C. Zhu, T. R. Park, S. R. Cho, C. S. Shin, and J. S. Ryu, “IEEE 802.15.5 WPAN mesh standard-low rate part: Meshing the wireless sensor networks,” *Selected Areas in Communications, IEEE Journal on*, vol. 28, pp. 973 – 983, September 2010.

[36] O. Omeni, A. Wong, A. Burdett, and C. Toumazou, “Energy efficient medium access protocol for wireless medical body area sensor networks,” *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 2, pp. 251 – 259, December 2008.

[37] G. Messier and I. Finvers, "Traffic models for medical wireless sensor networks," *Communications Letters, IEEE*, vol. 11, pp. 13 – 15, January 2007.

[38] D. Quevedo, A. Ahle andn, and J. andstergaard, "Energy efficient state estimation with wireless sensors through the use of predictive power control and coding," *Signal Processing, IEEE Transactions on*, vol. 58, pp. 4811 – 4823, September 2010.

[39] D. Martins and H. Guyennet, "Wireless sensor network attacks and security mechanisms: A short survey," in *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, pp. 313 – 320, September 2010.

[40] T. Aysal and K. Barner, "Sensor data cryptography in wireless sensor networks," *Information Forensics and Security, IEEE Transactions on*, vol. 3, pp. 273 – 289, June 2008.

[41] A. Lo, W. Lu, M. Jacobsson, V. Prasad, and I. Niemegeers, "Personal networks: An overlay network of wireless personal area networks and 3g networks," in *Mobile and Ubiquitous Systems - Workshops, 2006. 3rd Annual International Conference on*, pp. 1 – 8, July 2006.

[42] IEEE, "Draft Standard for Telecommunications and Information Exchange Between Systems – LAN/MAN Specific Requirements – Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPAN) Replaced by IEEE 802.15.3-2003," *IEEE Std P802.15.3/D17*, 2003.

[43] IEEE, "IEEE Standard for Information Technology- Telecommunications and Information Exchange," *IEEE Std 802.15.1-2002*, pp. 0 – 27, 2002.

[44] R. Rashid and R. Yusoff, "Bluetooth performance analysis in personal area network (pan)," in *RF and Microwave Conference, 2006. RFM 2006. International*, pp. 393 – 397, September 2006.

[45] World Health Organization, "Electromagnetic fields and public health," May 2006. Accessed on 26/8/11.

[46] A. ICNIRP (International Commission for Non-Ionizing Radiation Protection) Standing Committee on Epidemiology:, Ahlbom, A. Green, L. Kheifets, D. Savitz, and A. Swerdlow, "Epidemiology of health effects of radiofrequency exposure," in *Environmental Health Perspective*, vol. 112(17), p. 17411754, September 2004.

[47] L. Salford, A. Brun, J. Eberhardt, L. Malmgren, and B. Persson, "Nerve Cell Damage in Mammalian Brain after Exposure to Microwaves from GSM Mobile Phones," in *Environmental Health Perspective*, vol. 111(7), p. 881884, June 2003.

[48] Joku.it, "Free serial port terminal," July 2011. Accessed on 26/8/11.

[49] M. Q. Chen C, Shi P, "Application of remote sensing techniques for monitoring the thermal pollution of cooling-water discharge from nuclear power plant," *Environmental Science and*, vol. 38, pp. 1659 – 1668, August 2003.

[50] J. K. Moon, *Warmed Blankets are Safe.* Devicix, LLC, 2006.