Smart Soles: Posture Analysis Using Wireless Sensors

Paul Flanagan

A dissertation submitted to the University of Dublin,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science.
(Mobile and Ubiquitous Computing)

2010

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: _____

Paul Flanagan

13/09/2010

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signed: _____

Paul Flanagan

13/09/2010

Acknowledgements

I would like to thank my supervisor, Dr. Ciarán McGoldrick, for his persistent constructive feedback and encouragement throughout the year.

I would also like to thank my friends and family for their invaluable assistance and support on the Smart Soles Project. In particular I would like to thank Michael Flanagan, David McGrath, Aileen Shaw and Meabh Shannon.

Signed: _____
Paul Flanagan
13/09/2010

Contents

Abstract

Posture is a key area of analysis in many rehabilitation exercises. [1]. Patients who are being monitored during rehabilitation exercises are often under-sampled by healthcare professionals [2]. It has been shown that patients monitored in their habitual environment (such as their home) act more naturally and hence will give better results for monitoring motor movements [3]. Thus, monitoring in the natural environment was the motivation for the Smart Soles system developed for this study.

Our Smart Soles system is designed to be as unobtrusive as possible. It provides users with a system that can be used in their own home to measure and try to correct any posture abnormalities. In order to do this Sunspots from Sun Microsystems are utilized with a Bluetooth module, and a Wii Balance Board provides the external sensors. This potentially allows the system to interface with a large number of mobile devices that use Bluetooth. The mobile device is used for providing pseudo real-time corrective postural feedback to the user. It was found that the Posture Correction application developed through this study provides a state of the art cost effective, configurable, real-time, unobtrusive and intuitive system.

1. Introduction

1.1 Motivation for Smart Soles

Posture is a key area of analysis in many rehabilitation exercises. [1]. Patients who are being monitored during rehabilitation exercises are often under-sampled by healthcare professionals [2]. It has been shown that patients monitored in their habitual environment (such as their homes) act more naturally and hence will give better results for monitoring motor movements [3]. Thus, monitoring in the natural environment was the motivation for the Smart Soles system expounded in this study.

Sunspots from Sun Microsystems are utilized with a Bluetooth module interfaced with a Wii Balance Board and Android phone. The Bluetooth module potentially allows the system to interface with a large number of mobile devices that use Bluetooth. The mobile device is used for providing pseudo real-time corrective postural feedback to the user.

There are some existing products that focus on postural rehabilitation such as the "RehabSPOT" in [7]. Most of these products involve the use of a professional at all times. In [7] a professional is needed to analyse movements in real time that are graphed on a PC. The application developed through the Smart Soles system aims to minimize professionals' inputs and free up their time by letting the application suggest corrective actions to the user. The capability to connect a smart-phone to the application to provide the feedback is an important element of unobtrusive health monitoring.

The goal of the Smart Soles system is to provide users with a cost effective, configurable, real-time, unobtrusive and intuitive system to report and correct any posture abnormalities. It provides users with a system that can be used in their own home to measure and try to correct any posture abnormalities.

1.2 Report Layout

Chapter 2: Background Technologies

Background technologies provide a brief insight into the major technologies used in this project. The technologies discussed include:

- Integrated Development Environments for use with both the Android phone and the Sunspot.
- Bluetooth as the form of communication.
- The WT32 chip which provides the Bluetooth communication channels.
- The Android and Sunspot platforms
- The Wii Balance Board that's used for its four pressure sensors.

Chapter 3: Posture Correction Application Overview

This chapter provides a high level overview of the Posture Correction Application. It shows how the individual pieces are linked together. This provides a better understanding of how and why the different technologies are used.

Chapter 4: State Of The Art

The State Of The Art chapter highlights the knowledge behind the posture correction application that has been built upon. It provides us with the underpinnings of the various work and research that has been done in the field of Posture Correction. As an understanding of the general topics is amassed, chapter 4 goes into more depth and provides discussion on existing solutions in the area. From this chapter various points have been identified that make the system developed through this study a unique and more effective solution than existing products.

Chapter 5: Design

The Design Chapter covers the system architecture and how the components should be linked together in order to create an effective solution. The different parts of the system design are discussed, including the Balance Board status reports, Sunspot components and the smart-phone components.

Chapter 6: Implementation

The Implementation describes how the different classes were created. It details how the methods were implemented and how the main invocations are used to perform various functions such as setting up Bluetooth channels, calibrating data and displaying real-time feedback.

Chapter 7: Evaluation

The Evaluation chapter covers experiments that were carried out based on a set of exercises. It covers problems that were encountered during design and development of the system. It also covers future work suggestions, personal career and development record and a conclusion based on what was implemented.

Chapter 8: Appendices

This chapter provides instructions on how to start the Posture Correction Application and how to set up the development platform.

Chapter 9: Bibliography

State of the art references are listed here. Any other technologies or other external sources utilised in are also listed in this chapter.

2. Background Technologies

2.1 Bluetooth

Bluetooth was designed as a short range, low power, and inexpensive communication technology. As a result of this it has been developed in combination with many different technologies such as mobile phones and laptops. Bluetooth is on many millions of devices. It was estimated in [4] that it was integrated into approximately 32 million devices worldwide. There are different ranges for different classes of Bluetooth chips. The first version of Bluetooth chips has a range between 10-100 meters with a data rate of just less than 1 mb/s. The newer versions have data rates of between 2-3 mb/s. Bluetooth also communicates in the range of 2400–2483.5 MHz in Europe which is an unlicensed band and hence free to use.

With all these advantages of Bluetooth and the fact that it allows free communication to a mobile handset, it was the obvious choice for our Smart Soles system. It also provides enough data throughput to supply near real-time feedback to the user as discussed in section 1.7.

The Sunspot supports 802.15.4 but it does not support Bluetooth natively. As a consequence of this an external Bluetooth module (Bluegiga WT32 [27]) needs to be interfaced to the Sunspot for the purposes of communicating to an external Bluetooth device such as a mobile phone or a data logger.

2.2 WT32

The WT32 is a Bluegiga Bluetooth Audio module. It supports the latest Bluetooth 2.1 + EDR standard [27]. IT has an embedded DSP core and provides enhanced features such as "advanced audio decoding, echo cancellation, noise reduction and data manipulation" [27].

The WT32 has various specifications that were suitable for the Posture Correction System. It comes loaded with Bluegiga's iWrap firmware. The iWrap firmware makes it easy to setup or manipulate Bluetooth connections with minimum development. It's a class 2 Bluetooth device which means it has a range of up to approximately 30 meters. It has a very low power consumption (transmit power 7dBm) and a large temperature range (-35 to +85 degrees Celsius). These factors make Bluetooth communication an ideal solution to interface with the Sunspot.

2.3 Sunspot

Sunspots are wireless sensing devices that contain embedded microprocessors. They are relatively easy to program as they are based on the high level Java programming language [26]. This made it highly attractive as a base technology for this Smart Soles prototype system.

There are 2 different types of Sunspots, free range (FR) Sunspot and basestation. The free range Sunspots include onboard sensors such as light, temperature and 2G/6G 3-axis accelerometer. They provide wireless communication based on 802.15.4 protocol stack. They have 3.3v Li-Ion batteries. The FR Spots have multiple IO pins to facilitate additional sensors or circuitry. They have a 180MHz 32bit ARM microprocessors with a 4MB on-board flash memory and 512K RAM. This means that they are powerful potential embedded systems.

The basestation includes radio communication based on 802.15.4. It plugs into the USB port of the computer and does not have a battery, IO pins or internal sensors. It is used solely for communication purposes from the FR nodes to the PC [7].

The Sunspot is capable of having external sensors plugged in through the various IO pins on the eDemo board (application board that comes with the Sunspot) [26]. Sometimes, however, the sensors interface specification does not match that of the eDemo board i.e. the sensors might be required to work at a different voltage. If this is the case additional circuitry needs to be used to amplify or divide the voltage.

2.4 Wii Balance Board

The Balance Board was developed by Nintendo and intended to be used for their Wii games console. It is made up four pressure sensors. Two pressure sensors for each foot. It uses Bluetooth as its form of communication which can be utilized by the WT32 Bluetooth chip. The WT32 chip can setup and communicate using the Bluetooth HID profile. The Balance Board also has a synchronize button used for establishing communication to a Bluetooth device [15]. It provides different status reports that can be used to calibrate, extract and format the pressure sensor data into Kilograms [29]. It can be set to send the calibration data continuously.

The Balance Board performs as good as commercial force platforms for a fraction of the price [32]. For these reasons it is a highly effective technology to use with the developed Posture Correction system. It helps in providing a cost effective and reliable solution to produce real time corrective feedback

2.5 Android

Android is an operating system, built mostly in Java, which is primarily deployed on smart-phones. Android is now used on other devices (such as tablets) but the main focus is on mobile phones. The main components of an Android Application are:

- Activities – These are primarily used for GUI. The initialisation of any GUI components will usually be done here.
- Intents – These are system messages that are set running and used to notify the application of any changes to the system.
- Content Providers – These provide a level of abstraction from persistent data storage. This would be useful if we are persistently storing configuration values or logging data for our Smart Soles application.
- Services – These are designed to be kept running even if the application is shut down. They can keep on running in the background.

This structure provides us with a more "crash resistant" application [33]. Also since the development is in Java which is the same as the Sunspot there is not much of a learning curve for development. It may be possible to use some of the same utility classes on the Sunspot (e.g. parsing data that may be sent back and forth over Bluetooth).

2.6 Eclipse

Eclipse is used to build an Integrated Development Environment that in turn can be used to develop software applications for the Android operating system. Eclipse is a composition of different components that create the environment. The Java Development Tools and the plugin environment are the critical pieces of Functionality for Java Development [34]. These can be built upon further, with unique plugins for various purposes. With the Android Development Toolkit (ADT) plugin, Eclipse provides us with an easy solution for creating, developing and debugging Android projects.

2.7 Netbeans

Netbeans is an Integrated Development Environment that can be used to develop software for the Sunspot Microprocessor. A Sunspot plugin for Netbeans simplifies the creation and deployment of Netbeans projects. It provides menus that run built-in Ant scripts to perform various functions such as building and deploying code to a Sunspot in just one click. This greatly simplifies the running of projects and allows more time for software development.

Similar to Eclipse, to get up and running with Netbeans is quick and easy. This was the major advantage in choosing the Netbeans IDE for the Sunspot side of this Smart Soles Project.

3. Smart Soles Overview

The primary aim of this project was to create a mobile application that provides real-time corrective feedback and can be used by anyone. This mobile application is interfaced with a Sunspot and Wii Balance Board. The system as a whole is designed to be non-intrusive. The smart phone will provide the user with a Graphical User Interface suggesting corrections to their posture. The Wii Balance Board is part of the Nintendo Wii fit package which retails for $99.99 US [25]. This provides us with a cost effective non-intrusive platform on which to build our system.

An overview of the system is shown in figure 1 below. This shows the Balance Board sending and receiving signals to and from the Sunspot microprocessor. The Sunspot microprocessor interfaces with a Bluetooth chip which is used for communication. After receiving mass information from the sensors in the Balance Board the Sunspot formats it, converts it to KGs and sends it (via the Bluetooth chip) to the smart-phone.

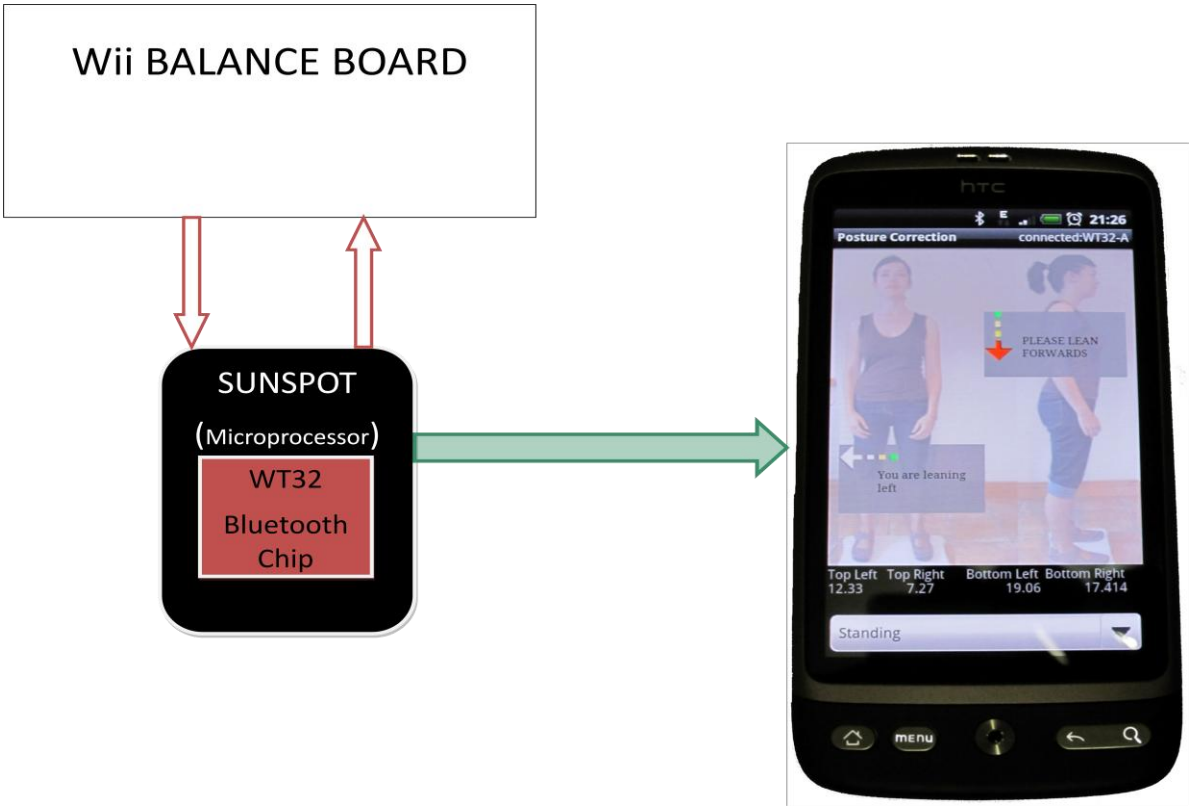


Figure 3.1: System Overview

The user will have a choice of exercises which were designed in collaboration with physiotherapists. The exercises are used to retrain a person's internal motor skills

which may be damaged due to a neurological disorder. They may also be used to retrain muscle groups after sporting injuries (e.g. a shoulder injury).

The correctional feedback provided to the user is in the form of arrows indicating which way the user is leaning. These arrows are complemented by text informing the user they are leaning backwards, forwards, right or left. See the arrows and descriptions below.

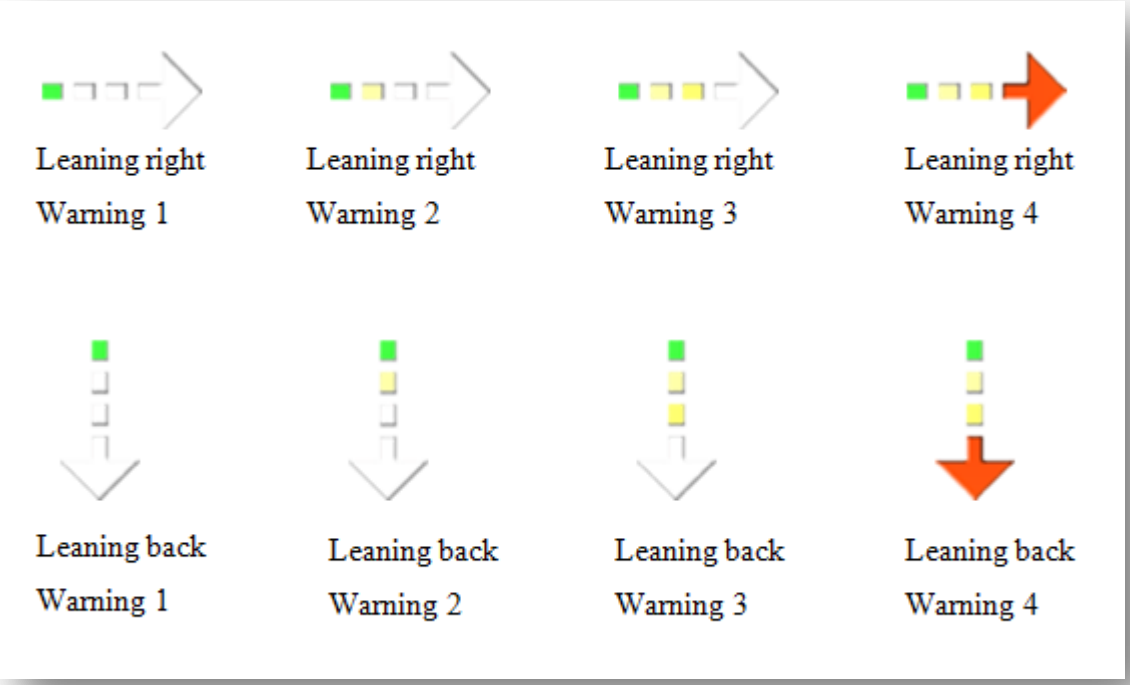


Figure 3.2: Warning Levels

Similarly, there are arrows for leaning forward and left. There is also a green tick presented to the user when they have the correct posture.

4. State Of The Art

4.1 State Of The Art Introduction

The related work section has been structured in such a way that the more general topics are discussed first. As an understanding of the general topics is gathered, gradually more specific topics are introduced. Towards the end of this section State of the Art applications related to Smart Soles and posture measurement are reviewed. This chapter provides the underpinnings required to understand the various work and research that has been done in the field.

It starts with an introduction to wireless sensor networks (WSNs). Wireless sensor networks are networks that provide information about the world around them. They have specific sensors that measure temperature, speed, sound, etc. These do a specific job and generally upload information to a basestation for further processing. Since microchips and sensors are getting smaller, WSNs are getting more unobtrusive and in many cases can fit pervasively in to the world around us. This leads onto a discussion on Body Sensor Networks (BSNs). Sensors are shrinking at such a rate that they can fit into the fabric of clothing or even into the body itself. A variety of opportunities have arisen from this in the healthcare field. Some of these applications are examined in section 1.2.

In the following section Sunspots are introduced. They are a good example of the hardware used in setting up a wireless sensor network or even a body sensor network. Sunspots provide a good platform for Research and Development into wireless sensor applications. They are programmed at a high level using Java. For the purposes of our project “Smart Soles”, the attachment of external sensors to the Sunspot will be a requirement in order to get a pressure reading back from the foot to identify the posture.

In order to correctly identify a person’s posture an understanding of what posture is and what a neutral posture should look like is needed. The Posture section also covers how to relate posture to the different pressure points on a person’s feet.

To gauge the pressure exerted by a person’s foot the Wii Balance Board is a good source. How the Wii Balance Board is used for these measurements and various applications pertaining to this are detailed in the Wii Balance Board section.

The Rehabilitation Applications section discusses a variety of rehabilitation applications. Many of these applications consider posture and fitness. The correct posture is very important while carrying out many exercises in order for a full recuperation.

Finally this document considers the State of the Art applications in this area. An investigation into applications that measure posture from foot pressure is carried out. This includes what technology is used, experiments carried out and how the results are processed.

4.2 Wireless Sensor Networks (WSN's) and Body Sensor Networks (BSNs)

"Thus far a range of applications have been proposed for the use of WSNs and they are likely to change every aspect of our daily lives" [1]. In the last few years the semiconductor industry has made huge progress with the miniaturization of microchips. They have made microprocessors smaller than a pinhead and cheap enough to be considered disposable. Much work has also been done in other fields regarding low powered wireless communication, battery life, processing power and sensor design. This miniaturization and the wide adoption of these microchips from a large variety of vendors have made it possible to create ubiquitous applications and we will continue to see a rise in such applications in the near future. [1].

A large scale platform that is considered highly ubiquitous is Smart Dust [1]. This consists of very small sensors that are scattered around a battle field. These sensors communicate together to relay information wirelessly back to a basestation that will provide real time updates of enemy activity. Small low cost wireless sensors known as motes are used to monitor a variety of external environmental conditions.

New WSNs can be seen as falling into three categories, "monitoring environments (indoor, outdoor, urban or countryside), monitoring objects (such as machines and buildings) and monitoring the interaction of these objects with environments" [1]. From the aforementioned categories it can be seen that WSNs tackle a wide range of problems.

A big factor in the fast emergence of WSNs has been the open source operating system for motes called Tiny Microthreading Operating System (TinyOS) [1]. This provides the capability to conserve power, can be run on very small processors and is free to use. It also helps with sensor measurements and communication. This has made programming motes easier with a quicker transition for a higher level programmer.

WSNs have a wide variety of uses; however they are not exactly suitable to measure movements from the human body. For this reason Body Sensor Network (BSN) platforms have been proposed.

BSNs have arisen as a related platform to WSNs. This is because BSNs have different challenges associated with them. Some of the main challenges involved are listed here. BSNs work on a lot smaller scale. Nodes on a body need to be smaller. There are also fewer nodes on a body than there can be in a WSN. Although there are fewer nodes they generally need to

be more accurate. Power supply is a bigger concern in BSNs. Generally nodes need to be careful and intelligent in how they consume power. In WSNs power is also a factor but may be more easily supplied. There are many more challenges that BSNs face. However, from the main ones listed above it is clear to see that WSNs and BSNs are very different platforms and may behave in different ways. Therefore some of these points needed to be kept in mind when negotiating the Smart Soles system. Many of the challenging factors faced in BSNs were not a troubling issue from the point of view of developing this Smart Soles system. This is due to the fact that the all sensors are external, whereas many BSNs need to be biocompatible due to being implantable sensors.

In [2] Paolo Bonato discusses the emergence of miniature sensors and their use in monitoring patients in their own environment. This technology is expected to save a lot of time and money and provide the clinic with accurate data about a patient's health.

Bonato discusses the impact that this technology could have. The data gathered is very useful to a clinic as it provides them with patient's health or movement data in their own settings. One of the main focuses was initially to monitor motor functions. The technology that was primarily used for this was accelerometers. As clinicians sought greater granularity, the level of complexity necessary for monitoring motor functions increased. Patients were given certain tasks to do and these tasks needed to be monitored for patterns of movement. "The combination of multiple sensors allows one to estimate the kinematics of movement with a reliability that cannot be obtained by solely relying on accelerometers"

There were 3 areas are identified by Bonato associated with making the change from wearable sensors just being used as an initial part of the monitoring to making it a daily practise:

1) The sensors need to appear unobtrusive and reliable;
2) Multiple sensors should be used and this data should be readily available when it is needed by the clinic;
3) Data needs to be reliable and the clinicians should be able to dynamically modify sensors in such a way as to improve the movement of a patient.

Examples of systems to measure movement and gather data and prototype applications for physical medicine are given in Bonato's paper. We will see from these applications that wearable technologies have a significant relevance in the field of healthcare. The main focus is on rehabilitation in a clinical environment. As more wearable technologies evolve this focus could be shifted to a field environment, e.g. the patient's home. This is an important motivator in the development of a Smart Soles system. This system can be used in

many types of environments and will provide valuable data that can be used to help health care professionals monitor exercises and health related to standing posture.

4.3 Posture

Analysing posture is an important part of many rehabilitation programs today [11]. Some of these programs or posture analysis systems are discussed in greater detail 4.5.

For the purpose of this project it is important to understand the different types of posture and the typical values we are likely to see when implementing the Smart Soles system.

Stancic et al. [11] use a dynamic force plate in order to determine human posture. The analysis is based on Pedotti's diagram. This is "a vector-diagram where each vector represents the projection of the ground reaction force into the sagittal plane."

An experiment to measure posture was carried out using 12 people [11]. A force plate was used to measure three degrees of freedom x, y and z. X is considered backward/forward, y side to side and z downward/upward. These values were measured from the force plate at a rate of 100Hz. From the 12 people who were experimented on, three main groups were identified:

"I. Forward slope $45^\circ < \alpha, \beta < 90^\circ$

II. Center slope $45^\circ < \alpha, \beta < 135^\circ$

III. Backward slope $90^\circ < \alpha, \beta < 135^\circ$"

"Inclination of boundary vectors is denoted by α, β"

The identification of these groups is of particular interest for the Smart Soles application. It identifies by how much regular healthy subjects posture might vary from each other. Out of 12 people, 7 people were analysed to be in group 1, 4 people in group 2 and only 1 person in group 3. This leads to the belief that the forward sloping posture is the most common.

Stancic et al. came to the conclusion that if they could experiment on a large range of people and determine an average standing position that it could be useful when analysing people with hearing impairments. However, I don't think this would be the case unless the person with the hearing impairment was analysed before and after the impairment. Otherwise neutral posture should be used as the correct standing position. This needs to be taken into account when using the Smart Soles system if it's being used by people with hearing impairments.

4.3.1 Neutral Posture

Identifying neutral posture is an important part in the development of the Smart Soles project. In order to provide useful corrective feedback to the user the initial correct posture is vital to ascertain. O'Sullivan et al. in [10] discuss the "Effect of Different Standing and Sitting Postures on Trunk Muscle Activity in a Pain-Free Population". The paper implies that different postures that may look the same could have very different effects on the muscles. This is why postural training is vital in many rehabilitation applications. The most important part of this paper as regards the Smart Soles system is the identification of different postures.

A neutral standing posture is achieved when the skeletal system is optimally vertically aligned. [10]. This can be seen from figure 4.3.1 below. This infers that the head shoulder knee and ankle are approximately 180° from each other. The head is erect and straight.

Figure 4.3.1: Erect standing posture: 177° angle between markers. [10]

4.3.2 Foot Pressure Points in Relation to Posture

Benocci et al. identify the most important pressure from the heel and toe [21]. This important identification of pressure points is reinforced in [12]. Ito et al. describe how ground reaction forces applied to both the heel and toe pressure points can affect posture. As with the Smart Soles system the posture detection only takes into account the sagittal plane (i.e. leaning forward and backward). The ankle acts as an actuator to keep ground reaction forces positive at both the heel and the toe to keep balance. [21]

In [21] a simulation is set up in order to determine the different ground reaction forces and the necessary switching from heel to toe to counteract these reaction forces. This makes a

person stay upright. This is in an ideal situation where there are no external environmental conditions like wind influencing the situation. For the purposes of the simulation in [21] the external factors are taken into account when determining posture changes. The conclusion that was reached is that if there is an eternal force such as wind a person will naturally lean into the wind affecting the ground reaction forces and natural posture. For the purposes of the initial Smart Soles prototype it is assumed that any measurement of posture giving real-time feedback is indoors and that there are no external environmental conditions influencing the analyses.

4.4 Wii Balance Board

The Balance Board is made up four pressure sensors. Two pressure sensors for each foot. Its shell is a white plastic cover. It looks like an elongated weighing scale. It is powered by 4 AA batteries and uses Bluetooth as a means of communication. It has a synchronize button used for establishing communication to a Bluetooth device. It also has an ON/OFF button [15].

4.4.1 Wii fit

Wii Fit is a set of video games that was developed by Nintendo. These employ various uses of the Balance Board. It can measure your BMI, how many steps you've taken, how many push ups you've done, etc. This data can be obtained from the 4 sensors in the Balance Board. It's a great form of entertainment and a way of people interacting through physical activity [25].

There is now also Wii Fit plus [25]. This is built upon the software for Wii Fit but has more tools for personalising different exercises. You can create training routines for yourself and keep track of them all. Although the Wii Fit is not very portable and the Smart Soles project is, the Wii Fit still provides great insight into the implementation of the Smart Soles project. It shows how only 4 sensors can provide a wide variety of applications and uses. In the next section other applications that use the Balance Board are discussed.

4.4.2 Balance Board Research Applications

Haan et al. in [15] describe how the Wii Balance Board can be used as a virtual reality device. Their system can use either continuous mode or discrete mode when accessing sensor values from the Balance Board. In continuous mode sensor data is repeatedly read from the sensors. In discrete mode the data read only consists of Boolean values based on a threshold that may be modified. Continuous input is used for 2 or 3 degrees of freedom e.g. for navigation or rotation. Discrete values are used for selecting objects or "control input".

The system sets up communication from the Balance Board to the PC via Bluetooth. This is done using the WiiUse library [15]. This is an open source C library. To retrieve the sensor values it is integrated with a Virtual Reality Peripheral Network (VRPN).

The Balance Board is initially calibrated by monitoring sensor values over 0.5 seconds and taking the average. Sitting and standing positions are determined by the system by Thresholding the sensor values. Thresholding is also used to "generate discrete on/off values".

Various uses and applications are described in [15] such as "3D Rotation Control", "Navigation Control" and "Abstract Control". 3D Rotation Control can be used to move around an object of particular interest. 3-axis have been defined x, y and z. A user can shift their weight around these axes to perform different rotations. It was noted that shifting weight to the y axis when seated was more difficult. Navigation control is useful for moving around virtual worlds. Leaning forwards and backwards controls speed. Pressure on the left and right provides a shifting movement to the left and right. Heel pressure on one foot and toe pressure on the other foot provides a turning movement. It was noted in this case that shifting left and right can be slow and requires careful calibration as it's difficult to put pressure on the heel side. Abstract Control involves using discrete ON/OFF values. Therefore only 1 dimension is needed. It can be used for "controlling time in a time dependent visualization and zooming in and out on an object of interest" [15].

Haan et al. have found that not all rotations happen in a smooth manner. This is because a user may be slower or find it more difficult to lean in a certain direction. This in turn slows down the system.

This paper gives a good insight into some uses of the Balance Board. It provides knowledge on how the values from different axes can be read and put to use. This is an important factor in the system for posture detection and feedback. It can be seen from this paper that the system needs to be calibrated and this will also be an important factor in the operation of the posture correction system associated with this study.

In [9] Gómez et al. describe a rehabilitation system that is designed for the home called "eBaViR". It is used for customizable rehabilitation programs based on posture and balance. Gómez et al. note that it is essential that the patient's motivation for the rehabilitation process remains high. In order to do this, the system needs to be engaging and easy to use.

eBaViR uses specifically configured games prepared with the help of physical therapists. These games have a simple interface and keep the user engaged in the

rehabilitation activity. The system does not require any calibration and the setup of the system is designed to be very easy for non-technical users.

Gómez et al. have noted that the Wii Balance Board has been compared to other systems and performed well as a physical rehabilitation tool. eBaViR games results are logged and can be then used by a physical therapist for further evaluation.

This paper gives us a good indication of the potential uses of the Balance Board for posture detection. For the purpose of the system developed for this study it is clear that the Balance Board provides a good tool with which to prototype the concept. Gómez et al. have not made the eBaViR system description very clear. It is hard to see how a user's motivation level will be kept high especially if there is no immediate feedback given. Users will want to know the benefit gained after an exercise. The system developed in this study looks into various feedback techniques.

4.5 Rehabilitation Applications

Different types of rehabilitation applications based on wireless sensor networks are discussed in this section. Here a description of a generic framework is described that can be used in rehabilitation applications. The need to identify *posture* for many rehabilitation applications is also an important feature that needs to be investigated. For example, determining neutral posture is crucial in analysing spinal conditions [14]. Some applications regarding this are described in the next section and in section 4.5.2 some fitness applications are discussed.

Zhang and Sawchuk in [7] have proposed a "highly customizable networked wearable sensor system for medical monitoring and physical rehabilitation" known as "RehabSPOT". Their paper mainly focuses on the software involved in this adaptive sensor system and outlines different body area sensor network (BASN) applications and how a framework can be used to incorporate the major components of these applications. Some of the major components are discussed here:

- They have identified that motor training can be used to rehabilitate people with temporary loss of function.
- Data that is recorded should be wirelessly transmitted to some base station where it can be monitored by trained medical physiologists.
- Regular patient data collection can be unreliable or imprecise.
- Patients are only under medical supervision for a limited time and as a consequence may be under-sampled.

- To get more precise data it is important to make the BASN as non-intrusive as possible. To facilitate different types of rehabilitation and a more personalized programs the system needs to be highly configurable.
- As the system is designed for medical staff with limited technological capability the system needs to be easily configured.
- Obviously the data also needs to be reliable or the whole system will fail.

With these points in mind Zhang et al. have developed a system architecture model for their RehabSPOT. The software part of this architecture is composed of a three tiers:

1) The first tier consists of free range nodes that can communicate together or with a basestation.

2) The basestation is the master node and makes up the middle tier. The basestation transmits any data it receives to the PC for further analysis.

3) Data from the PC is uploaded through the Internet to a remote server for further processing. This makes up the third tier.

The hardware used consists of Sunspots from Sun Microsystems (Oracle) as discussed in Chapter 2. Various types of sensors are used with the RehabSPOT. The model, make or specifications of the sensors are not described in the paper. The different sensors are connected via a "signal conditioning accessory board". This helps handle different voltages from the different sensors as it has built in voltage amplifiers and voltage dividers. The details of how the board was made are not included in this paper.

Zhang et al's main focus in this paper is the development and implementation of their software architecture. The software was written for the client and server architecture displayed in figure 4.5.1 and figure 4.5.2.

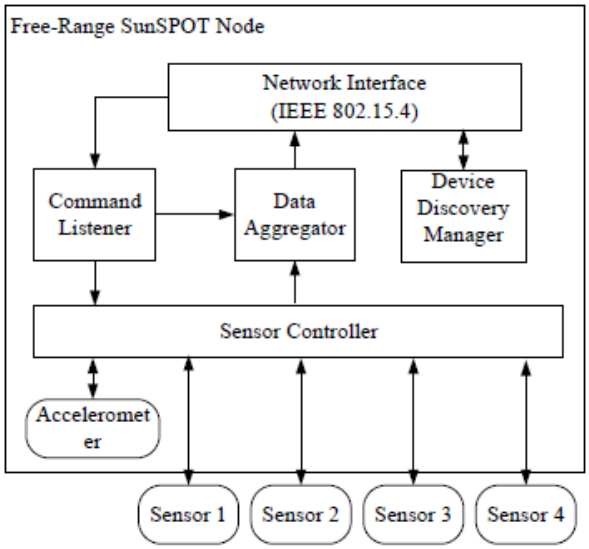


Figure 4.5.1. Client Architecture for the RehabSPOT [7]

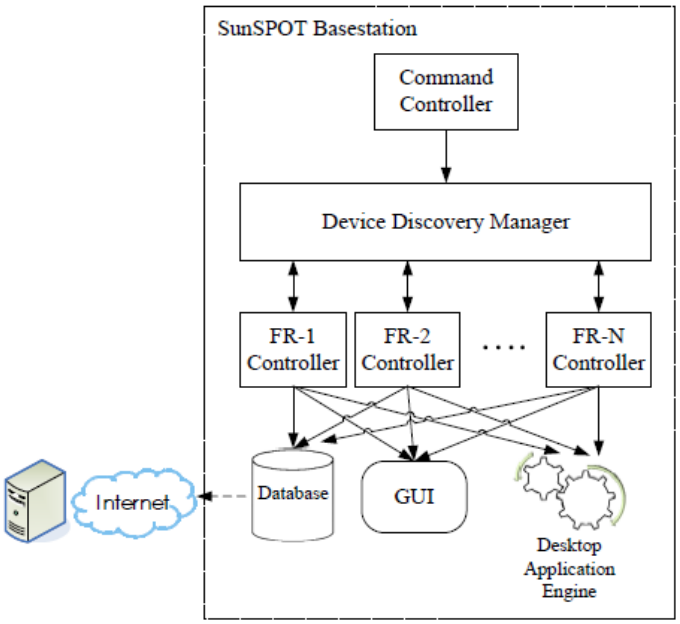


Figure 4.5.2. Server Architecture for the RehabSPOT [7]

The software is written in Java. The server program is written on the PC so can use the full Java library. The client program runs on a FR Sunspot and as a consequence uses a cut down version of Java with a few extras libraries provided by the Sunspot SDK. The communication used for the RehabSPOT is encrypted using Java and is based on "Elliptic Curve Cryptography". It's highly efficient and allows for more reliable communication.

The client program consists of 4 main components, "the device discovery manager, command listener, sensor controller and data aggregator". The device discovery manager handles the wireless communication and nodes being added in an ad hoc fashion. This is done using initialisation messages known as "*Heartbeat*" messages. These messages are broadcast over 1 hop. The messages sent back are known as "*Heartbeat_ACK*" messages. The command listener listens to commands from the network interface. It forwards commands onto the Data Aggregator or the Sensor Controller. The Sensor Controller is responsible for connecting or turning off and on sensors. It contains a low pass filter which is used to detect if a sensor has failed. The Data Aggregator sends replies to the PC (server). This has been identified as the bottleneck for data exchange. A message could take between 2 and 4ms to send. For this reason, multiple messages are sent across together.

The Device Discovery Manager on the server side dynamically assigns and releases port numbers using a bitmap. A bitmap in this case seems to be a Boolean array of values.

Each entry in the array is a given port number. If a particular port is in use the array value for that port entry is set to false. However, the implementation of the bitmap is not clear from [7]. "Compared to many static configuration methodologies, such as TDMA-based polling, which has been adopted in many Health Monitoring systems, our dynamic network construction methodology is much more flexible and fault tolerant"[7].

The FR controller on the server side sends and receives data from the server side Device Discovery Manager. The data it receives is parsed and forwarded to the database and the GUI. The GUI displays the data on a graph and the database keeps track of recorded data.

Zhang et al. carried out experiments using the proposed framework. They connected one Sunspot to a person's left wrist and another to the right wrist. An external Gyroscope sensor was added to each Sunspot. The Sunspots were labelled from the GUI "*LeftWristNode*" and "*RightWristNode*". The sampling rate was set to 100Hz or 10ms from the GUI. All results were graphed and stored in a database. Based on this experiment they were able to see that the set up of the sensors on a person may take between 5 and10mins. The initialisation of the system takes about 5 minutes. Zhang et al. have not made it clear why it takes 5 minutes. This may be due to the time it takes to deploy code to the Sunspot and initialising radio controls. According to Zhang et al. this system shows great potential for therapists who want a reliable, configurable health monitoring system with little user intervention.

The system in [7] definitely has some potential. For this reason we have taken out parts which can be exploited for the Smart Soles project. For example, the bitmap to keep track of assigned port numbers may be useful in an ad-hoc system. A reliable and highly configurable system is important for most health monitoring applications. A generic sensor controller is useful in helping with configuring the system.

However, there are also some points here that are not needed for our project. In this paper [7] they have identified that the Sunspots communicate together to discover devices. However, from what can be seen in the paper this is not needed. The PC also does device discovery and the free range nodes do not talk to each other in any other way. Wireless sensors in our Smart Soles system will need to communicate together for the purposes of group posture analysis which can then be compared to individual postures.

The system in [7] also provides real-time analysis. However, our Smart Soles application needs to provide real-time feedback to the user to suggest a correction if needs be. This should be done without the resource of a PC or a healthcare professional. The capability to connect a smart-phone to the application to provide the feedback is an important element

of unobtrusive health monitoring. In [7] the authors have identified a communication bottleneck (takes 2-4ms for a data packet to be sent over the air) which may make it difficult for real-time feedback. As sensor readings can be combined and sent back for analysis to the phone/PC this should provide pseudo real-time feedback.

4.5.1 Posture

Posture is an important part of many rehabilitation scenarios associated with motor skills and balance. This was an important research topic in the area of this Smart Soles project. In order to analyse one's posture, you must first be able to determine it. Farella et al. in [19] present a system known as "WiMoCA". This system is a wireless body area network to determine posture. They describe this system as well as "a custom software and network architecture for wireless body area sensor networks"

The WiMoCA consists of a Microprocessor Control Unit with sensors, radio frequency (RF) communication and a power supply. An ATmega8 microcontroller is used along with a 3-axis accelerometer. The RF communication works at 868MHz. The gateway also uses this frequency to communicate to other nodes. However the gateway is also responsible of communicating with the PC over "RS232, Bluetooth of Ethernet". The power supply works off 3.3v.

The posture recognition application consists of strategically placing 3 nodes around the body. One node is placed on the trunk, one on the thigh and one on the shin. With the nodes in place the application is able to detect "user posture among seven different possibilities (sitting, standing and lying in four different manners)". Accelerometer values are read and sent to the gateway node. The gateway node sends the results back to the PC where a Java program is running and based on the information it receives it can display the users' posture. The nodes are responsible for acquiring sensor data from the accelerometers. It works out the average value of 8 samples for this data to minimize false positives. The node also forwards its results to the gateway node. The gateway node is responsible for communication between nodes. It receives the sensor data from the different nodes. It combines the data from the different nodes on the host side to figure out the body position. This position is sent to the Java application to be displayed on the GUI. The gateway can tell what position the body is in through a "Body Location Table (BLT)" stored on the gateway. The different nodes readings are looked up in the table and according to this a user's posture is determined.

The host application on the PC, continually reads data coming in on the serial port. The user's posture is taken from this data and used in the GUI. The GUI consists of an image that represents the current body position. The GUI also provides information on what tables are being used (e.g. BLT) and what the initialization control commands are.

The communication layer uses a form of a "collision free MAC protocol". This doesn't have the overhead of extra packets that might be found the collision avoidance MAC protocol. It's also easy to implement. Effectively, nodes cannot communicate at the same time. In order to do this, nodes need to know how to schedule their packets. This scheduling information is kept in a message table on each node. It needs to be the same on all nodes. The table may be changed dynamically before or after a frame is sent/received but not during data acquisition.

The authors used a set of experiments to measure power consumption and posture detection performance. As expected, receiving packets used less power than sending packets. It was also noted that sending a bit as 1 used more power than sending a bit as 0. This could lead to data encoding techniques to reduce power consumption. Farella et al. have identified this as possible future work. The total power consumption at the maximum sampling rate was between 16.85mW and 46mW. The maximum sampling rate was set to 60 positions per second because "In practical cases, the maximum frequency of human movement is 30Hz". This provided a sampling rate with minimum loss of data. As the sample rate increases the power consumption also increases.

When Farella et al. talk about measuring posture they are just measuring limited body positions. The Smart Soles system focuses on measuring more fine grained posture changes in a standing position.

However the paper of Farella et al. holds similarities to the posture system developed in this study in that the nodes communicate to each other on one frequency (low powered radio) and then the gateway nodes use a more common means of communication to send data to the host (Bluetooth, RS232). Similarly, the body area network nodes (Left and Right foot) associated with this study communicate to each other over Zigbee. The gateway node of the Smart Soles system can then communicate over Bluetooth to a phone or PC. This system proposed by Farella et al. is similar to The Smart Soles system in that it provides near real time feedback to a Java program which displays a user's current posture.

This paper has also offered insight into how to keep track of whether a posture is correct from nodes readings. In the case of the WiMoCA system they use a BLT. Similarly, on the gateway node associated with this paper a Posture Correction Table (PCT) which

determines variations in posture can be implemented. The Smart Soles system is much more sensitive to changes in posture than the WiMoCA system. This will be of more use in rehabilitation applications where a higher degree of accuracy may be needed.

4.5.2 Fitness

There are many applications available for fitness. One that is of particular interest is portrayed in [13]. This paper describes a system that provides feedback for runners according to their pattern of arm swings. According to Gotoda et al. long distance runners may change their pace up and down, but it won't affect their overall performance. What really affect their performance are their motor patterns. This is the motivation behind only monitoring arm swings. The system described here has also taken into account that group training leads to better results. With this in mind Gotoda et al. have also discussed a "Group pace-training scenario".

The prototype system makes use of Sunspot technology described earlier. While a user is running the Sunspot keeps a log of data (arm swings). Feedback is given to a user according to what type of settings they have specified. Feedback comes in the form of an LED lighting and audio via an earplug. It can determine incorrect arm swings or distances between logged runners in a group scenario. Arm swings are determined by monitoring wave forms of acceleration and frequency.

Accuracy for the system is given as 66% for one runner based on four trials and 47% for four runners. Four trials are not enough to determine a reliable accuracy level. This paper is very vague on details about the system. For example no sampling rate is given, details about the group scenario are not clear, how the feedback is provided and interpreted is also not clear. However, the paper still provides an idea of how Sunspots can be put to use and of how the accelerometers in the Sunspot can measure momentum. It also gives the idea of how group rehabilitation scenarios might entice patients to partake more frequently or for longer periods of time.

Another novel application for fitness using wireless sensors is given in [16]. This paper involves assessing a user's arm movement and comparing it in real time to that of the correct arm movement. This may help in the rehabilitation of a user/patient as it suggests corrective measures and logs data to be analysed by a professional. This paper also describes how interactive media can help a user's motivation to complete the training or exercises. This is a similar trend to many of the papers described in this related works section.

In [16] Crossbow Micaz motes are used in combination with an MTS310 sensor board for their prototype system. This sensor board comes equipped with a 2-axis accelerometer. These motes with sensor boards are connected to a person's wrist and elbow. Another mote without the sensor board is connected to the PC and acts as a gateway for communication. The routing protocol used here is known as "Personal Activity Monitoring – Low Energy Master Alternating protocol (PAM_LEMA)". All sensors on the body are considered to be on a cluster. One of those sensors is elected as cluster head. The cluster head is responsible for collecting packets from all the nodes and forwarding them to the gateway node (the node connected to the PC). Each node on the body takes its turn as cluster head. The election process happens every 10 seconds. A random number is generated on each node and sent to the current cluster head. The node with the highest number is now the new cluster head. This spreads the increased power consumption of sending and receiving messages amongst all the nodes. If election messages are lost and data is sent to the old cluster head, it responds with packet containing the new cluster head. This process helps with power consumption and network failure.

The GUI contains a video of a virtual trainer demonstrating how the exercise should be undertaken. Besides that, there is a video showing how many repetitions the user has completed. The movements of the wrist and elbow are calculated and compared with the preloaded values - the correct movements. Users undergo 10 repetitions, after which they will be presented with their overall scores. The user is scored in the areas of speed, uniformity, regularity and execution. An overall score out of 10 is presented to the user.

This paper is useful for the PAM_LEMA protocol. Future designs of the Smart Soles system can benefit somewhat from the use of this protocol. In the current design there is no need to generate random numbers and keep routing tables, but as the Smart Soles system is extended there may be use for this. The paper [16] also uses comments to suggest correction techniques to the user. This is similar to the Smart Soles device where the mobile device suggests corrective procedures. Since the corrective procedures associated with Smart Soles are on a mobile device it adds to the mobility and range of uses the system can achieve and gives it an advantage over the system described in [16].

4.6 Real-time Analysis and Feedback

Real-time feedback needs to be implemented in our Smart Soles project to provide corrective feedback on posture stances to the user. In order to do this effectively we needed to compare a system that uses real time feedback on a mobile device over Bluetooth.

In [23] Zhang et al. have outlined an E-learning system that is used to provide lecturer and student interaction. The system is built using the Bluecove library for Java based on JSR-82 ("Java APIs for Bluetooth Wireless Technology") [23]. The server is running a J2SE (Java 2 Standard Edition) application with Bluecove. The client (mobile phone) is running J2ME (Java 2 Micro Edition).

In this E-learning system the lecturer uses a laptop as a visual guide for the lecture and to initialise the system. The students use mobile phones in order to connect to the system. As there can only be 7 active devices for one Bluetooth network (known as a Piconet) [23], this limits the amount of students that can be connected to the system at the same time. However in [23] Zhang et al. have proposed a way around this. They have identified that there can be 255 inactive Bluetooth devices in a Piconet. If a Bluetooth device is inactive and part of a Piconet it is known to be in parked mode [23]. All students in the class start off with their device in parked mode. A student who wants to send feedback to the lecturer can select from a list of predefined options on their phone or send a bespoke message. As the student presses send their device wakes up, establishes a connection, sends the message and goes back to parked mode. This approach is also useful for saving battery power.

The Smart Soles system will need to provide quicker feedback as the sampling rate of the sensors will be a great deal higher than a student sending a message. For this reason we cannot go into parked mode. However, the technologies used in this paper provide us an insight on how to proceed with pseudo real time communication with our system.

4.7 Posture – Foot Measurement Applications

Paper [21] is of particular interest as parts of it are semantically equivalent to the Smart Soles project. Benocci et al. describe a wireless system that a person wears in order to monitor gait. Areas of interest in this paper are regarding power saving, calibrating the IMU (Inertial Measurement Unit), common communication protocol, determining the best sampling rates and the most important aspects of foot pressure.

There have already been applications proposed in the multimedia and healthcare domains in order to measure the walking speed and the angle in which the foot moves in the sagittal plane. In order to do this accelerometers and gyroscopes have been used. Similarly in

this paper, accelerometers and gyroscopes have been used as part of the IMU. These measurements combined with pressure measurements from the insoles are used in order to determine gait.

The system described in Benocci et al. consists of two wireless sensor nodes, one for the right foot and one for the left foot, and a gateway (PC communicating over Bluetooth). The wireless sensor nodes are broken into five different categories as follows:

1) “Sensorized Insole”;
2) “IMU”;
3) “Processing Unit”;
4) “Radio Interface”;
5) “Power Unit”.

The Sensorized Insole consists of 24 foot sensors “based on hydrocells by Paromed” [21]. The cells are placed in an insole that is flexible. The cells are surrounded by fluid. The total pressure is measure by summing compression and raw force. The cells vary from each other in size. They are placed all around the insole from heel to toe. “41.8% of the plantar surface is covered” [21]. The Processing Unit consists of an MSP430 microcontroller. This provides a serial interface in order to connect to the IMU. The power supply consists of a 5v and a 3.3v supply with a 3.7v lithium ion battery. The insole is powered by 5v, whereas the microcontroller is powered by 3.3v. The radio communication implemented uses Bluetooth 2.0. This allows for interaction with a wide range of different applications.

The Inertial Measurement Unit (IMU) consists of a 3-axis accelerometer and a 3-axis gyroscope. It has a 5v power supply. It is interfaced with the microcontroller via a serial protocol interface (SPI) to provide the samples from the sensors.

The sample rates can be configured Over The Air (OTA). Benocci et al. describe the different sampling rates from 20Hz to 200Hz as having no significant difference on the results. In order to accommodate readings of frequent inertial data the sample rate in this paper was 80Hz. The system is capable of running up to 250Hz.

Experiments were carried out identifying important groups of sensors and indicating that heel and toe pressures can be used as on/off values. The primary goal in the experiments here was to identify standing posture and movement of the leg. The temporal movement of the leg is used in gait analysis. From the analysis of the data there were 4 main groups of sensors from the insole identified shown in figure 4.7.1. These were identified as having the most effect during stance and gait analysis. The heel and toe are most representative when detecting walking movements.

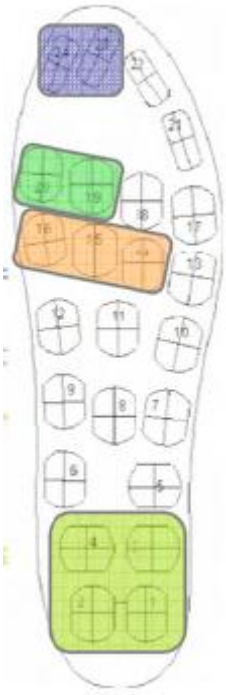


Figure 4.7.1. Four main groups of sensors [21]

During the leg movement phase, the heel and toe needed to be identified as being on or off. Thresholding the values may not be enough as there may be different pressure values from insole to insole. For this reason a derivative of the average value from the sensor data was used to determine on/off states.

IMU data from the accelerometer and gyroscope was analysed in the experiment. The authors only took into account movements in the sagittal plane i.e. forward and backward leg swings. Y-axis data from the gyroscope measured the angular swing of the leg.

Data calibration was considered an important part of the experiment. This is done in the phase known as “Quiet Stance”. The reason this is an important stage is because the sensor values can vary from person to person according to whether the sensors were mounted properly and what type of footwear the user has on. Only after this is done should the system analyse the data. According to the paper calibration and subsequent system analysis can be performed for up to 5 hours, “the lifetime of the system is 5 hours of continuous operation with 3.7V, 1200mA, Li-Ion battery” [21].

Future work mentioned in this paper consists of biofeedback to the user according to their sensor data. Providing real time data to the user was one of the goals of the Smart Soles project. Based on this data the user is told how to correct his/her posture in as non-intrusive a manner as possible. This study identified some different technology to be used in posture

detection. The technologies used in this paper, such as the IMU and the Paromed sensors, are very expensive even for a prototype. The cost has been reduced by not including a gyroscope and only including a set of sensors for each foot. A set of sensors for each foot is enough to accurately determine posture. This can be ascertained by the grouping of sensors by Benocci et al. They have identified the 4 main sensor areas that are covered by this study, 3 groups near the toe and 1 group near the heel. It has been identified from the work of Benocci et al. that pressures around the heel and toe are what need to be analysed.

This study also necessitated identification of how often these pressure points may need to be analysed. Paper [24] undertakes a very interesting study which helps with identifying this need. For the purposes of the Smart Soles project the technology Benocci et al. use and the sample rates involved is of most interest. Olguin et al. have tried to detect and analyse differing individual and group behaviours utilizing wireless sensors. They have tried to identify "physical activity, speech activity, face-to-face interaction (f2f), physical proximity, and social network attributes from sensor data".

During these experiments they monitored 67 nurses for a period of 27 days. Each of the participants was asked to fill out a survey at the end of each day. This survey consisted of 6 simple questions about how the participant felt in the workplace on that particular day. These surveys were then compared with the data collected from the sensor measurements.

The sensor measurements that are of most interest are the physical activity measurements. These were carried out using a 3-axis accelerometer such as the one found in the Sunspot for the Smart Soles system. The important point noted here was that the sample rate for monitoring physical activity effectively should be set to more than 30 Hz. "99% of the acceleration power during daily human activities is contained below 15 Hz" [21]. This information was very useful when considering sample rates for the heel and toes in developing the Smart Soles system.

4.8 State Of The Art Summary

From this related work chapter the need for a Smart Soles application that provides real time feedback on a mobile platform has been identified. The uses of Body Area Sensor Networks (BSNs) were discussed in the context of rehabilitation applications. They have been identified as an important part of future developments in terms of rapidly expanding reliance on technology in the health care sector [1].

The Smart Soles project examines areas of research such as what hardware could be used and its suitability to the Smart Soles project. Sunspots were identified as powerful

embedded systems that are relatively easy to program [26]. These Sunspots interfaced with a Bluetooth module and external pressure sensors provide great potential to be used in the Smart Soles prototype system as part of a BSN.

The main aim of Smart Soles prototype system is to provide feedback related to a person's standing posture. The reason the system has been designed to do this was due to complementing research in the area of posture. In order to correctly understand the ramifications of bad posture and identify neutral posture it was necessary to review papers in this area such as [3] [10] [12] [21].

To correctly analyse standing posture it was found necessary to identify pressure points from the foot [12]. To do this it was necessary to identify some hardware that could measure a person's foot pressure. The Wii Balance Board provides a great way of modelling foot pressure and is applicable to a mass market of consumers [25]. The Balance Board provides a Bluetooth module that sends unencrypted data Over The Air (OTA). Research has indicated that external pressure sensors can be added to the Sunspot and inserted into an insole to provide foot pressure values.

This study has identified that the Sunspots can be interfaced with a Bluetooth module and placed on the body of a person and that foot pressure values can be obtained from the Wii Balance Board (initially) to identify posture. It was then necessary to identify some way of putting them together. To do this some systems giving pseudo real-time analysis and feedback such as described in [7] [16] [21] [23] were identified. This provided an understanding of how to proceed with combining values from foot pressure, processing them and displaying feedback utilising a mobile device.

Applications concerning State of the Art in the area of Smart Soles were then discussed and how these can be improved upon. The pressurized insole applications that were identified, such as in [21], do not provide real-time corrective feedback. There are also some other posture feedback systems that cannot be used remotely, limiting their usefulness [9]. As a result of this, providing real time corrective feedback on a person's posture to a mobile device is the main goal of the Smart Soles system. It is felt that this could be of huge benefit in the field of healthcare.

Research papers evaluated in this related work chapter has provided the basis to proceed with implementing the Smart Soles prototype system. As the Smart Soles prototype system has been developed major points that have been identified from the Related Works are:

- Ease of use, including setup;

- To be installed on a person's body;
- Wireless capability;
- System will be used in any indoor environment;
- System will be as non-intrusive as possible;
- Highly configurable;
- Reliable;
- Provide instant corrective feedback;
- Cost effective.

With these points in mind it is envisaged that this system could be mass marketed to health care professionals, physiotherapists, physical therapists, fitness instructors, yoga instructors, sports professionals and the general population interested in fitness/health.

5. Design

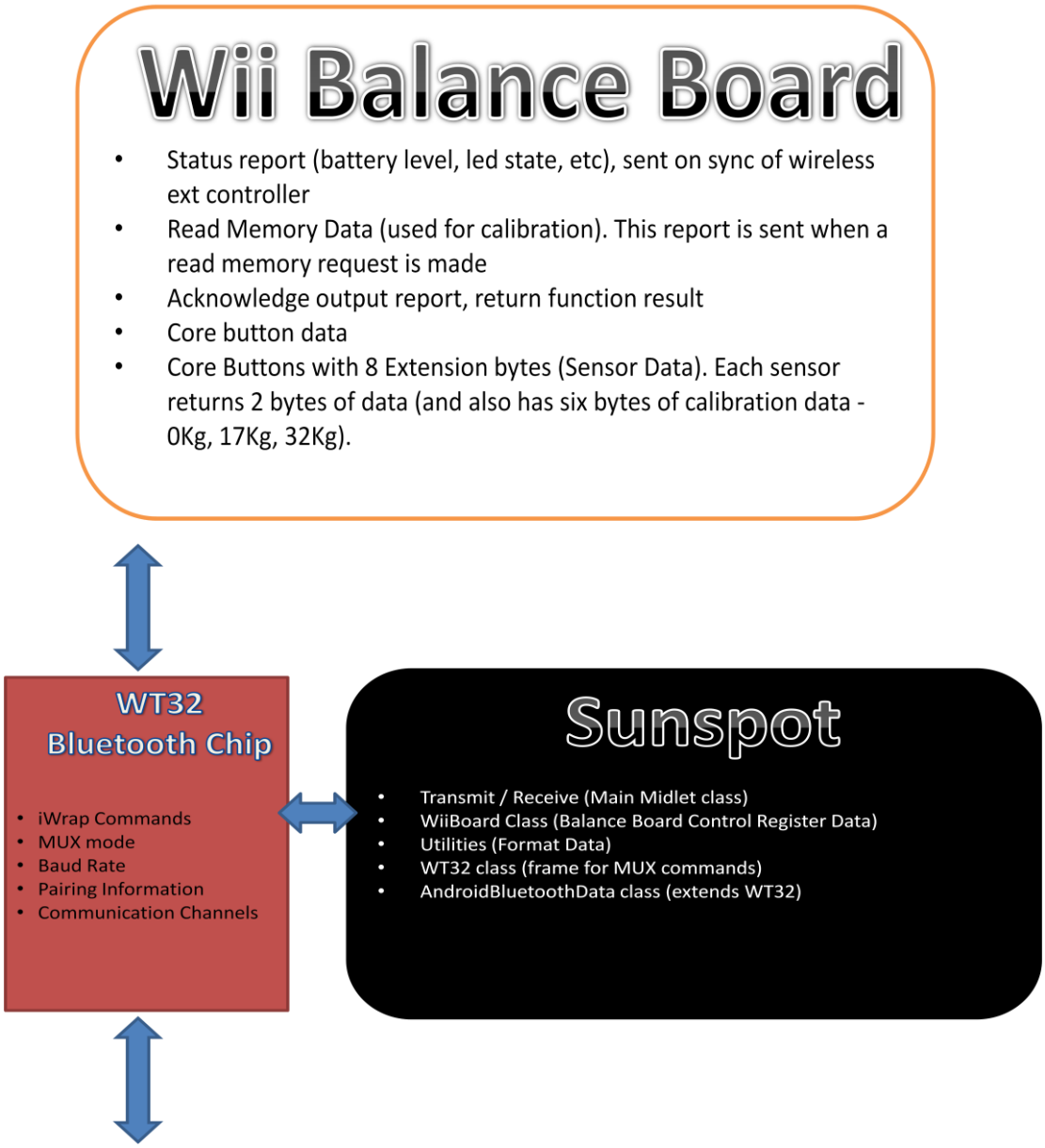
The initial project design involved connecting external sensors to a microprocessor. It was proposed that these external sensors would be placed in an insole which in turn was to be placed in a running shoe. The idea behind this was to be able to monitor different weight measurements based on foot pressure. There would be multiple benefits of having the pressure sensors in the insole such as the application being portable, ubiquitous and hence unobtrusive. These insole weight measurements would be used for evaluation of free weights training exercises. Different weight calculations and free weight exercise positions could be determined from the insoles. The initial design was always to display the analysis or feedback to a mobile device. The use of the microprocessor gives the potential for the system to be extensible. More wireless sensors could be added to different parts of the body to build up a Body Sensor Network (BSN). This is discussed further in the future work section.

For the actual project design the Balance Board was considered as a cost effective alternative for the external pressure sensors. As research progressed the design, involving the Balance Board, addressed more healthcare issues. Of these healthcare issues, posture rehabilitation was identified as a key area [1]. In collaboration with qualified chartered physiotherapists a set of exercises was designed to help correct or rapidly promote the rehabilitation of persons using this Posture Correction application. These are discussed in detail in the Experimentation and Evaluation chapter.

The Balance Board communicates via Bluetooth. In order to take advantage of this, the WT32 Bluetooth chip was wired up to the Sunspot microprocessor. This potentially allows the system to interface with millions of Bluetooth devices. The Android phone was used to display real-time corrective feedback to the user.

5.1 System Architecture

The top level components of the Smart Soles system architecture consist of a Wii Balance Board, Bluegiga WT32 Bluetooth chip, Oracle Sunspot microprocessor and a smart-phone (HTC Desire) running Google's Android operating system. The architecture below outlines these different components and shows how they are interconnected. A description of the design of each of these components is given in the preceding section.



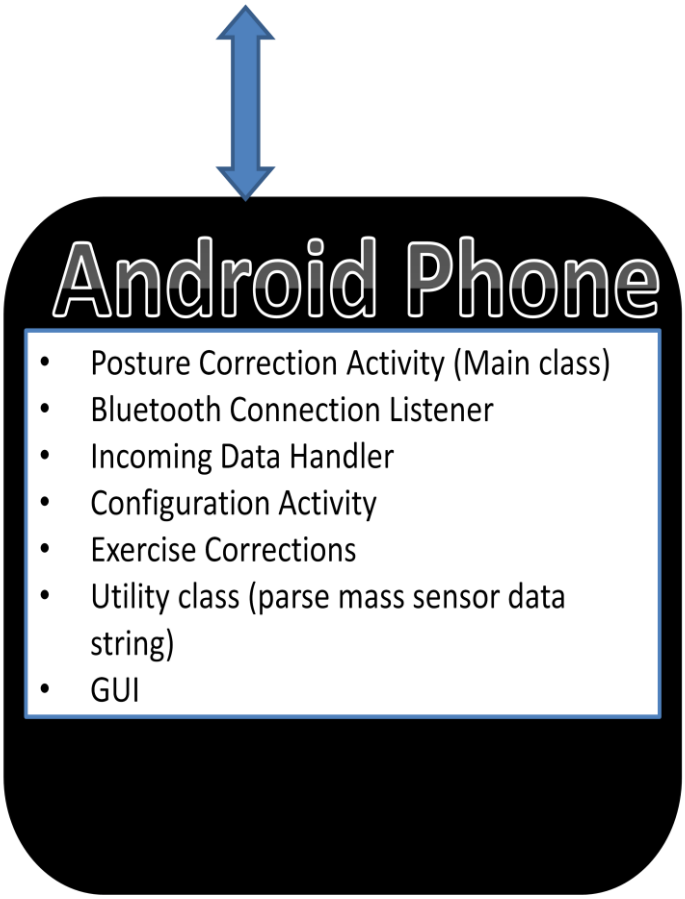


Figure 5.1.1 System Architecture

5.2 System Architecture Description

5.2.1 Wii Balance Board

The Wii Balance Board was used as a cost effective replacement prototype for the sensor insoles. The Balance Board provides 4 sensors, a power button, Bluetooth communication, status reports and data reports. These reports are in the form of a Bluetooth command followed by a report ID and different payload lengths of various bytes. The reports are all given in hexadecimal code. Wiibrew in [28] describes the different reports, their uses and their values. This is a work in progress and is not an official list of commands and values. Wiibrew is trying to reverse engineer the Wiimote and Balance Board to extract the meanings of the hexadecimal code being retrieved from the devices [28].

Reports that can be utilised from the Balance Board for the posture correction application associated with this study are as follows:

Status Report:

This report is sent either when we synchronize the Balance Board with the Bluetooth chip. The reporting mode needs to be set to continuous whenever a status report is received, otherwise no further data reports will be received. The structure of this report is as follows:

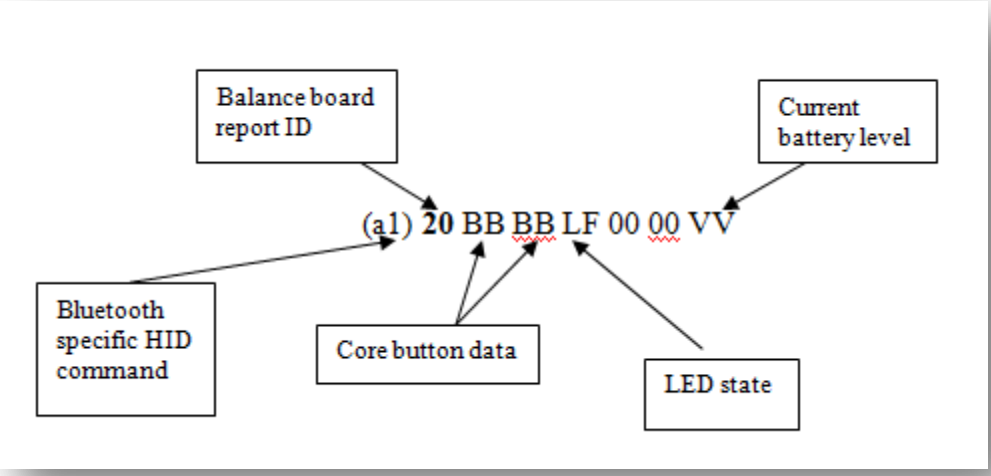


Figure 5.2.1. Balance Board Status Report Structure.

Read Memory Data Report

This is an important report as it is needed to calibrate the mass sensor readings from the Balance Board. When a read memory request is made to the Balance Board the data report is sent back. This data report contains 21 bytes of data in the payload, including calibration data for each sensor. There are 16 bytes of data used for calibration. If more than 16 bytes are requested than multiple packets are received with the offset increasing each time a new packet is received. The structure of this data report is as follows.

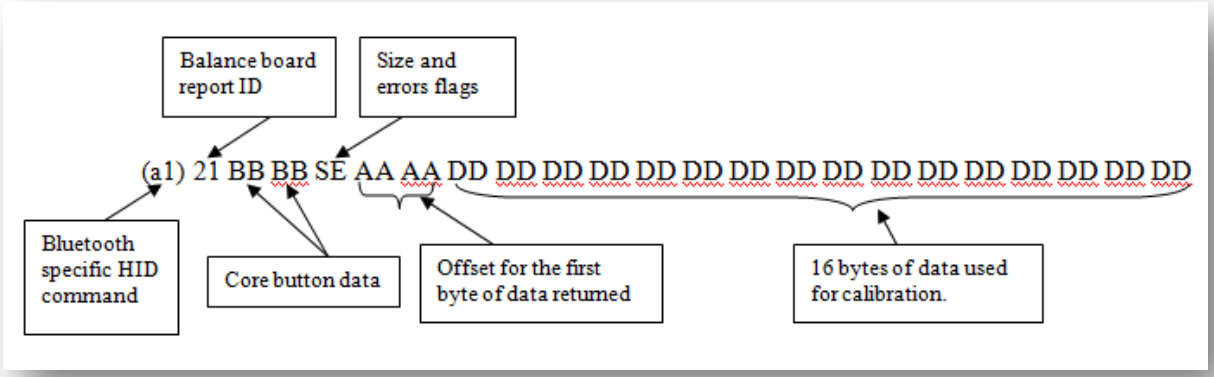


Figure 5.2.2. Balance Board Calibration Report Structure.

The calibration values retrieved from the Balance Board take the following form:

	Bit							
Byte	7	6	5	4	3	2	1	0
0	Top Right 0kg value<15:8>							
1	Top Right 0kg value<7:0>							
2	Bottom Right 0kg value<15:8>							
3	Bottom Right 0kg value<7:0>							
4	Top Left 0kg value<15:8>							
5	Top Left 0kg value<7:0>							
6	Bottom Left 0kg value<15:8>							
7	Bottom Left 0kg value<7:0>							
8	Top Right 17kg value<15:8>							
9	Top Right 17kg value<7:0>							
10	Bottom Right 17kg value<15:8>							
11	Bottom Right 17kg value<7:0>							
12	Top Left 17kg value<15:8>							
13	Top Left 17kg value<7:0>							
14	Bottom Left 17kg value<15:8>							
15	Bottom Left 17kg value<7:0>							
16	Top Right 34kg value<15:8>							
17	Top Right 34kg value<7:0>							
18	Bottom Right 34kg value<15:8>							
19	Bottom Right 34kg value<7:0>							
20	Top Left 34kg value<15:8>							
21	Top Left 34kg value<7:0>							
22	Bottom Left 34kg value<15:8>							
23	Bottom Left 34kg value<7:0>							

Figure 5.2.3. Wii Balance Board Calibration Values [29].

Acknowledgement Report

This acknowledgement is generated when the Balance Board receives an output report. Error codes are returned in the EE bytes. These refer to a specific error or 00 if the report is successful.

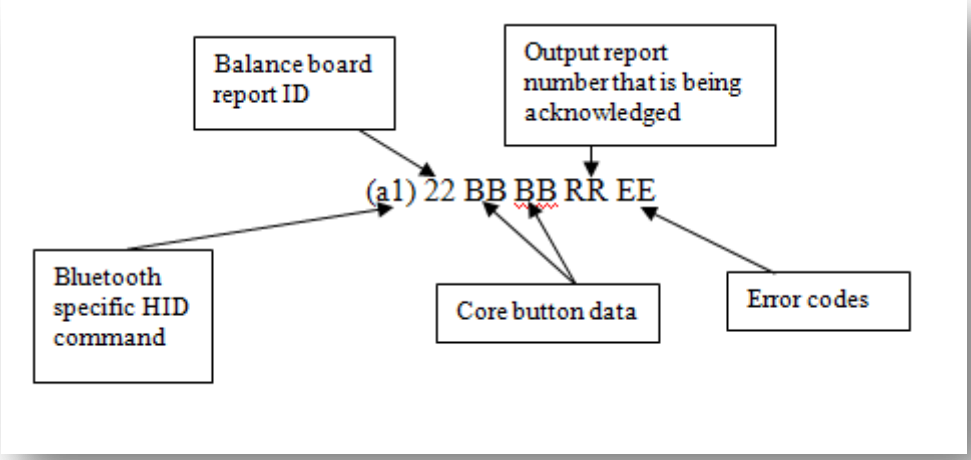


Figure 5.2.4. Balance Board Acknowledgement Report Structure.

Core Button Data Report

This report is sent whenever the button is pressed on the Balance Board. There is only one button on the Balance Board which controls the power. This is included in the design to make the system more extensible for any future changes. However, functionality behind the button press is not currently implemented. The reason for this is Core button data is also give in the 8 Extension bytes report and can be continuously read from there.

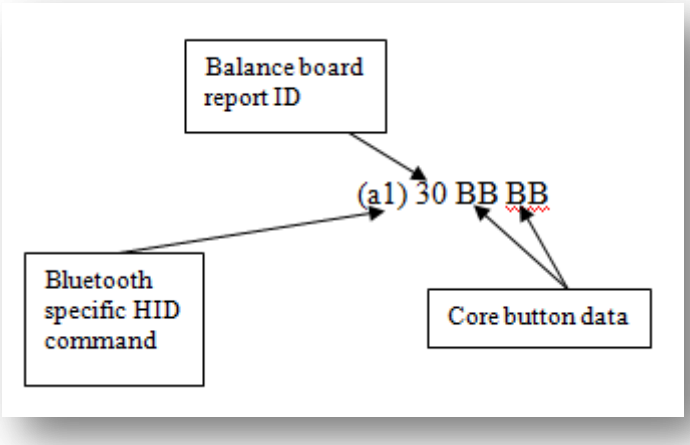


Figure 5.2.5. Balance Board Core Button Report Structure.

Core Button Data with 8 Extension Bytes Report

The Balance Board's sensors act in the same way as an extension to the Wiimote [28]. The 8 extension bytes contain the data from the four (16 bit) pressure sensors in the Balance Board. After attaining the calibration data, for the purposes of our project, this is the most important report. This report contains the pressure sensor data which is calibrated and formatted to kilograms. Once a command is sent to the Balance Board to set up continuous reporting the Balance Board continuously sends this report. The structure of this data report is as follows:

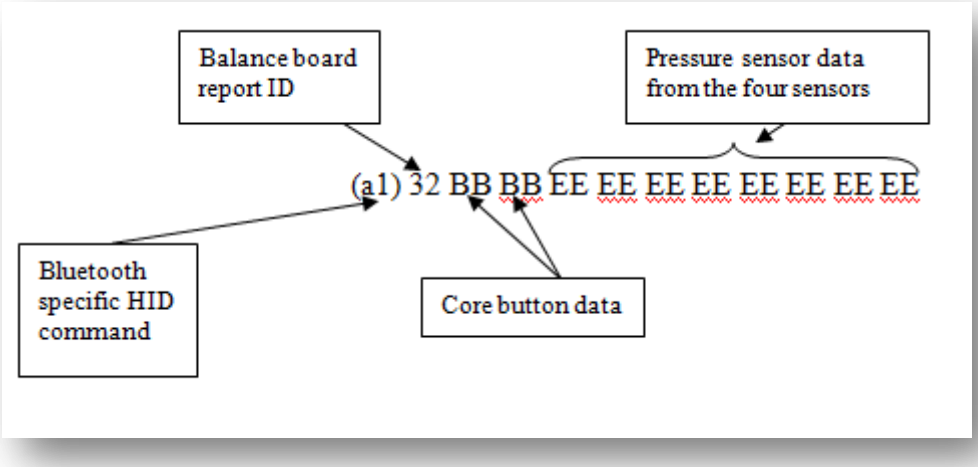


Figure 5.2.6. Balance Board Extension Bytes Report Structure.

The pressure sensor values are interpolated with the calibration values discussed above. The pressure sensor values retrieved from the Balance Board take the following form:

	Bit							
Byte	7	6	5	4	3	2	1	0
0	Top Right<15:8>							
1	Top Right<7:0>							
2	Bottom Right<15:8>							
3	Bottom Right<7:0>							
4	Top Left<15:8>							
5	Top Left<7:0>							
6	Bottom Left<15:8>							
7	Bottom Left<7:0>							

Figure 5.2.7. Wii Balance Board Pressure Sensor Values [29].

5.2.2. Bluegiga WT32 Bluetooth Chip

The Bluegiga WT32 was the Bluetooth chip of choice as it contains its own iWrap firmware which makes it easier to set up communication channels. It is a powerful and portable solution used on many handheld devices [27].

For the purposes of this project the WT32 was hooked up to the USB port on a PC via a TTL-232R-3V3 cable. The TTL-232R-3V3 is a USB to TTL serial converter cable. The process enabled the WT32 to be configured through various iWrap commands to match the needs of our project. To set up the WT32 for use with our project we set it to Multiplex (MUX) mode. This allows a saving of valuable processing time changing from command mode to data mode and back. MUX mode enables creation of up to four simultaneous connections without having to transfer between modes. Transferring between modes with Multiple Bluetooth connections can take up to 2 seconds [27]. After the WT32 is set to MUX mode, it only accepts hexadecimal code in a special frame format outlined by Bluegiga [27]. Any commands being sent to WT32 need to be wrapped in this special frame format. iWrap also allows configuration of the Baud rate to match the Baud rate of the Sunspot. This is important because if there are different Baud rates, data being transferred between the WT32 and the Sunspot will not be legible. The Baud rate refers to the amount of pulses per second on the line. The higher the Baud rate the quicker and more accurate the pressure sensor values from the Balance Board will be. The maximum Baud rate the eDemo board on the Sunspot is capable of achieving is 38400 Baud [26]. The maximum Baud rate the WT32 is capable of is 2765800 Baud [27], which is many times more than that of the Sunspot. Therefore to maximise accuracy and response time to the user the Baud rate was set on the WT32 and the Sunspot at 38400 Baud.

After these modifications were made to the WT32 Bluetooth chip it was ready to be hooked up the Sunspot Microprocessor. Inquiry, pairing, channel setup and breakdown of devices are done through iWrap commands sent from the Sunspot to the WT32. The WT32 handles the setup of the RFCOMM channel to communicate to the phone and the two L2CAP channels to communicate to the Balance Board.

5.2.3. Sunspot Microprocessor

The Sunspot Microprocessor facilitates hooking up external devices such as the WT32 Bluetooth chip, sending and receiving data from the chip, formatting the data and connecting to any other Bluetooth or Zigbee device in range. The Sunspot provides an extensible platform from which to build the Posture Correction system.

The Sunspot microprocessor is the control element behind the Posture Correction system. It controls the setup of communication channels to and from the Balance Board. It controls the formatting of the pressure sensor data, calibration data and all the other data from the Balance Board reports mentioned above.

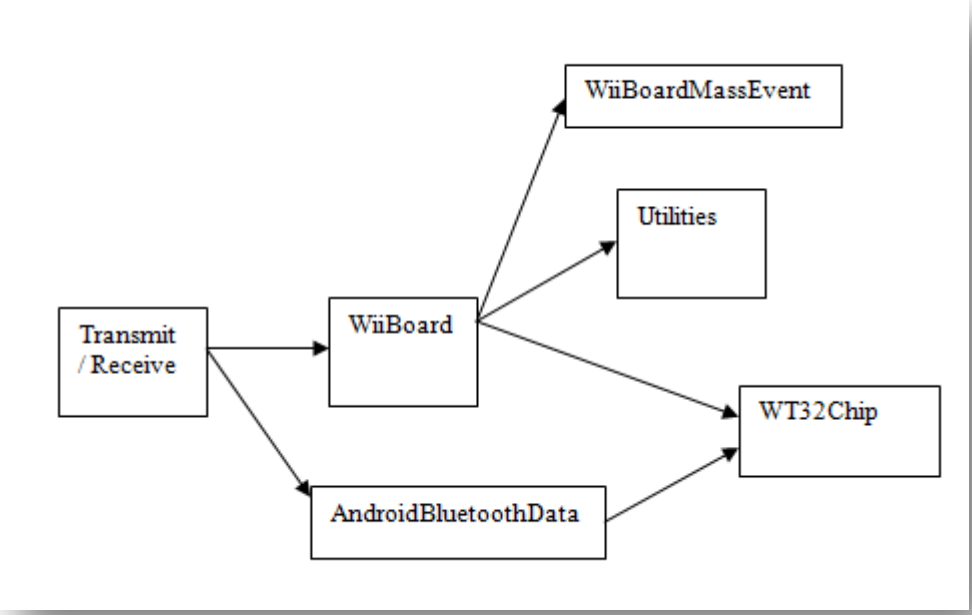


Figure 5.2.8. Sunspot Class Architecture

Transmit / Receive

This is the main class on the Sunspot. It sets up AndroidBluetoothData and WiiBoard instances and initialises the UART on the eDemo Board. It sets the Baud rate to 38400 to match the WT32. It checks for a valid connection with the phone and Balance Board devices. If there are valid connections the green LEDs light up on the Sunspot. Once the connections are established data is sent to the WT32 to be relayed to the Balance Board via our WiiBoard class.

WiiBoard

The Wii Board class contains all the commands to connect, read and write to the Balance Board. Firstly it establishes two L2CAP channels by sending iWrap commands to the WT32. These iWrap commands are sent in Hexadecimal format wrapped in the special frame format needed in MUX mode. Once the L2CAP channels are set up, it is possible to communicate to the Balance Board register informing it of the requirement to connect to it (the Balance Board acts as an extension and sends a command informing it of the requirement to connect to the

extension). Next it is necessary to send a command to set up continuous reporting on the Balance Board so as to facilitate reporting of the sensor data. This also includes getting the calibration data that correlates to the four pressure sensors.

AndroidBluetoothData

This class is responsible for setting up an RFCOMM channel to communicate to the Android phone. It firstly pairs with the phone, sets up the RFCOMM channel and informs the phone about initialisation of the other connections. This communication channel is setup first so that the user gets feedback from the phones GUI about the initialisation process. This class is responsible for sending the parsed pressure sensor data to the Android phone.

WiiBoardMassEvent

This is a mass event object class which contains getters and setters to WiiBoard mass events. A Mass event contains information about the force sensors on the Balance Board. To ensure the right sensors are being read, the orientation of the board should be so that the power button is facing the user as depicted below:

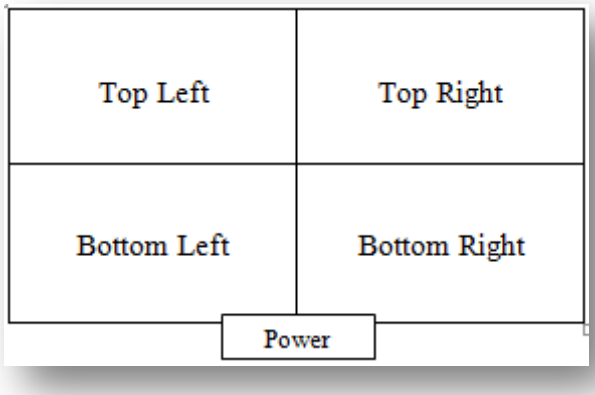


Figure 5.2.9. Balance Board Orientation

Android Phone

The system was designed with usability and configurability in mind. For this reason, the system is intuitive to use and can easily be utilised by a patient with no healthcare qualifications. The configuration details require more expertise. Any reference of the configuration screen is hidden from the home screen so it will not interfere with a user's experience.

The configuration screen can be found by pressing the menu button and touching the configuration option that is shown. A new frame is displayed with four fields to enter values into. These values are used to set the sensitivity of the system. For these values to be useful for users of different weights they are based on percentage of the user's mass. The four values refer to the different warning arrows displayed on the home screen. These warnings have default values when the application starts up. The default values have been agreed with physiotherapists as the most relevant initial values. The default values chosen are:

Warning 1 = 2.85 % (equals about 2kg of a 70kg person)

Warning 2 = 5.7 % (equals about 4kg of 70kg)

Warning 3 = 8.6 % (equals about 6kg of 70kg)

Warning 4 = 11.4 % (equals about 8kg of 70kg)

These allow for some variations in posture which is important when patients are starting their treatment. As their posture gets progressively better, the sensitivity can be increased.

The phone is responsible for listening for incoming Bluetooth connections, processing the incoming data, displaying available exercises, showing real-time corrective feedback based on the sensor data. An overview of how the main components are linked together is shown below:

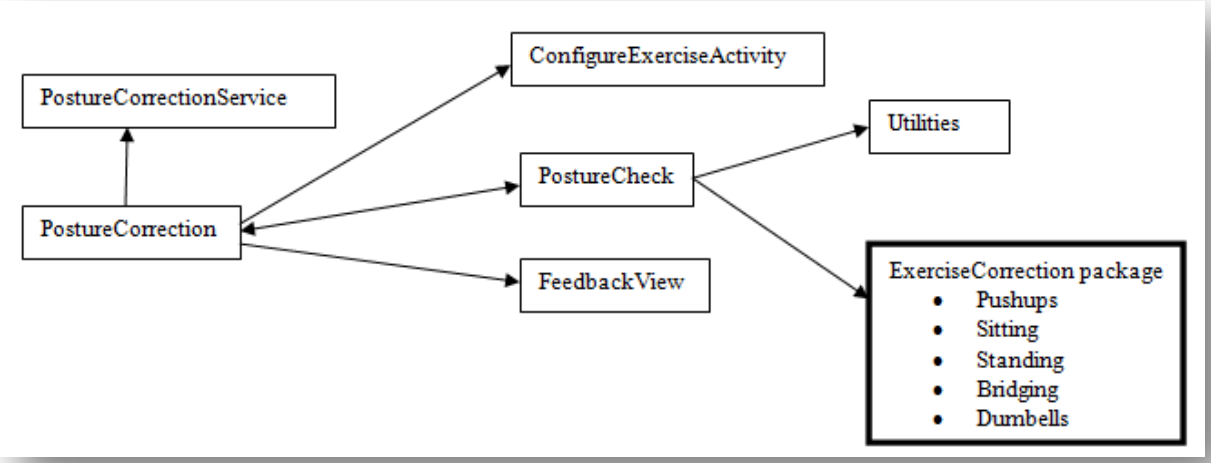


Figure 5.2.10. Android Class Architecture

PostureCorrection

The Posture Correction class is the main Activity. It provides the setup for all the GUI components. It initialises other services and Activities such as listening for incoming

Bluetooth connections, sets up a handler for incoming data, starts an intent for the feedback view and displays the relevant data in the correct format.

The GUI components on the home screen consist of a title, background image, text view and a spinner. The title shows the title of the application and the device connected to (WT32 in our case). The text view shows the mass data of from the different sensors in kilograms. This is updated in real-time. The drop down list or combo box (known as a spinner in Android) is shown at the bottom of the home screen. This drop down list gives options for different posture correction exercises. When an exercise is chosen a notification pops up giving a brief description of the exercise (known as Toast notification in Android) and the background image changes to show the current exercise to be carried out. Depending on the chosen exercise corrective feedback will be shown on the sagittal and/or coronal planes. The sagittal plane feedback informs users if they are leaning forward or backward. The coronal plane feedback informs users if they are leaning left or right. The image below shows the chosen exercise as standing, with sagittal and coronal feedback. The exercises are discussed in more detail in the experimentation chapter.

Figure 5.2.11. Standing Exercise showing Coronal and Sagittal Feedback

PostureCorrectionService

The Posture Correction Service is started by the Posture Correction class. Its main design feature is that it contains an inner thread class that listens for incoming Bluetooth connections. Once it has found a device it has another thread for connecting to the device. When it has established a connection it spawns a connected thread that sets up a handler to deal with the incoming data. The connected thread can also handle outgoing transmissions. For example if it was necessary to send the Sunspot commands to change the Baud rate, or sampling rate, etc. The handler in the connected thread shares the incoming data with the Posture Correction Activity which can display them on the GUI.

PostureCheck

The posture check class sets up the sensor values to be read and calls the appropriate exercise class according to the chosen value from the spinner. It gets back values depending on how correct the posture is for the given exercise. Any new exercise classes added to the application should be taken into account here.

Exercise Correction package

The Exercise Correction package contains an interface (ExerciseFeedback) which all exercise classes implement. This interface has a set of constant values depending on how correct the posture is for the different exercises. Depending on how correct the posture is, a value will be returned. Each of these values has an associated warning level indicator in the form of an arrow (bitmap image). See below for a full list of warning level indicators. Depending on the exercise a value for sagittal and/or coronal planes will be returned.

```
CORRECT = 0
FRONT1_SAGGITAL = 1
FRONT2_SAGGITAL = 2
FRONT3_SAGGITAL = 3
FRONT4_SAGGITAL = 4
BACK1_SAGGITAL = 5
BACK2_SAGGITAL = 6
BACK3_SAGGITAL = 7
BACK4_SAGGITAL = 8
LEFT1_CORONAL = 9
LEFT2_CORONAL = 10
LEFT3_CORONAL = 11
LEFT4_CORONAL = 12
RIGHT1_CORONAL = 13
RIGHT2_CORONAL = 14
RIGHT3_CORONAL = 15
RIGHT4_CORONAL = 16
```

Figure 5.2.12. Feedback Constants

For example if the exercise is standing, both coronal and sagittal values are returned. If the values returned are 1 and 12 the following feedback is displayed to the user:

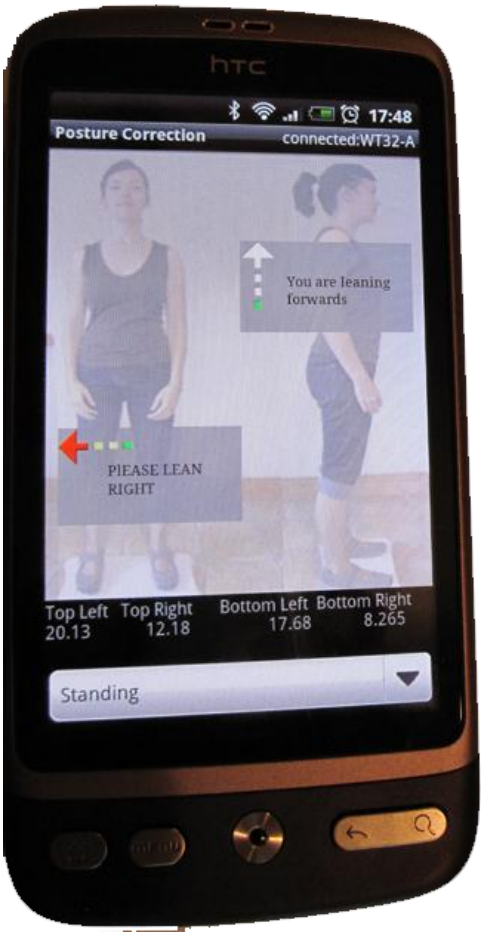


Figure 5.2.13. Feedback Example with Standing Exercise

The exercise package contains different classes for each exercise. This makes the exercises highly configurable and independent from one another. The exercise classes in our application are BridgingFeedback, StandingFeedback, SittingFeedback, PushupsFeedback and DumbellsFeedback. Additional exercises can easily be added to this package, The new exercise class needs to implement ExerciseFeedback interface. As discussed above they also need to be taken into account in the PostureCheck class.

Configure Exercise Activity

When a user selects the Configure Exercise option from the menu, the Configure Exercise Activity opens a new frame to the user. The user is presented with four fields to enter the sensitivity of the posture feedback. See the images below for the screen layout.

Figure 5.2.14. Configuration Screen Layout

The values entered in these fields are based on the percentage of body mass to make the posture correction application independent of the person using it. The four values correspond with the four different warning levels. For example if a user is 100kg and the values chosen for the four fields are 1, 2, 3 and 4, then the user will be displayed with the following warnings:

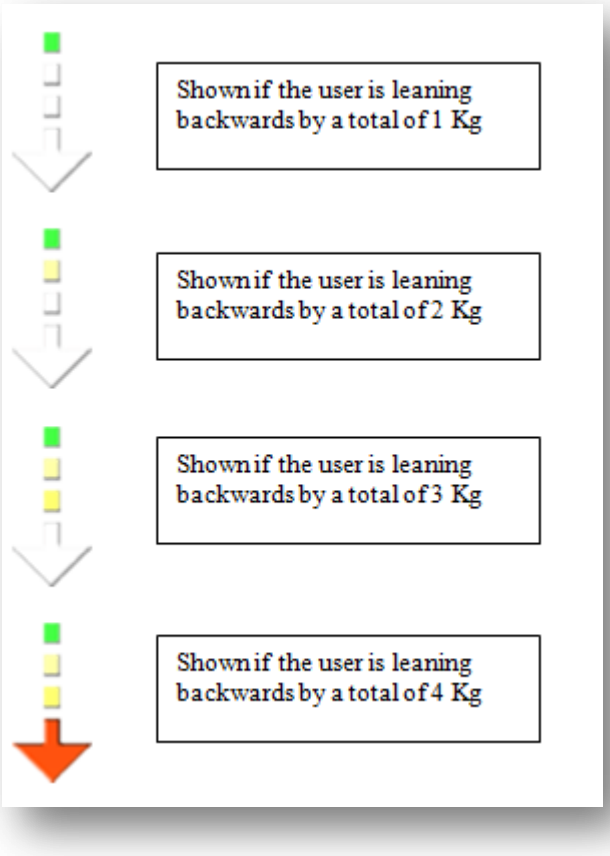


Figure 5.2.15. Backward Warnings

Please note the example given above just covers backward warnings. It will be the same for forward or side to side directions.

FeedbackView

The feedback view class contains a Canvas that needs to be frequently updated to provide the user with real-time corrective posture feedback. This View is set up from the Posture Correction Activity. The same View (two different instances) is used twice, one for sagittal plane feedback and one for coronal plane feedback.

According to the values received from the incoming data handler the Views will be refreshed with new values. A Canvas is needed for this as they will be refreshed frequently.

Utilities

The Utilities package contains a class to parse the incoming message from the WT32. The incoming message will be received as a String. The parse class will extract doubles from the

string and round the doubles to make them more suitable to display in the Text View on the GUI.

6. Implementation

For the purposes of this study, different platforms on which to write, test and run the code have been used. Netbeans IDE was the most efficient to develop the Sunspot code in. There is a plugin for the Sunspot microprocessor which makes it possible to create, build and deploy new projects using the Netbeans IDE menu. The Ant build tool can also directly be used to do this.

On the Android phone (Front End) side, Eclipse is the IDE of choice. Similar to Netbeans, there is an Android plugin for Eclipse. This provides Emulators, debugging ability, XML parsing (for android project resources) and menus for many more Android options.

The Bluegiga WT32 was initially configured using Bluegiga's BGTerm. This software permits iWrap commands to be sent to the WT32 chip. When the WT32 is set to Multiplex (MUX) Mode the BGTerm software automatically recognizes this and wraps any future commands in the special frame format needed. It also automatically converts ascii commands to Hexadecimal commands needed for MUX mode. This is the most important aspect of BGTerm as it allows the system to setup, pair, send data and test the different connections. A Screenshot of BGTerm is shown below after the SET command and L2CAP channels setup is sent to the WT32 chip.

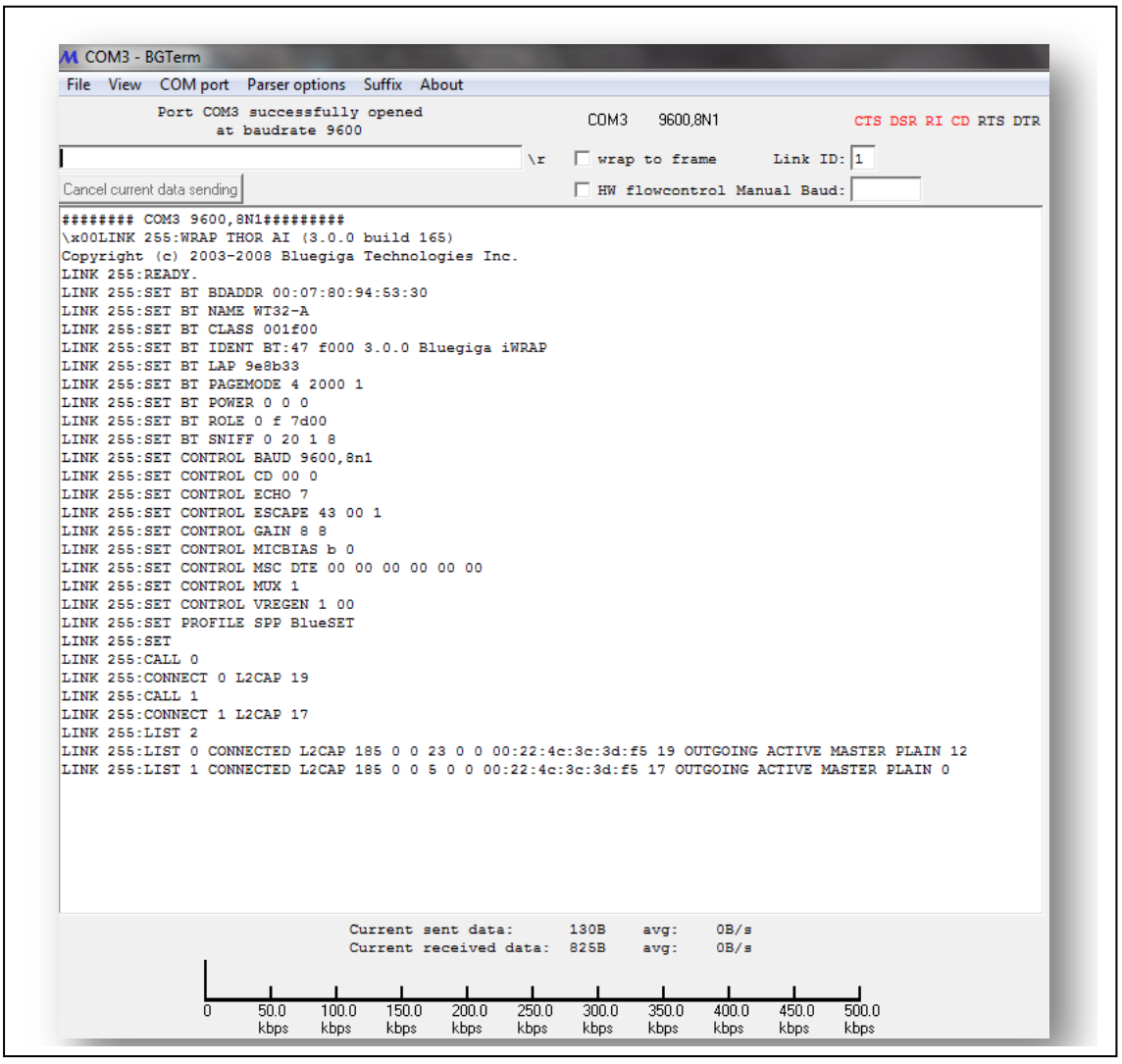


Figure 6.1. BGTerm showing settings and L2CAP channels setup

In order to use BGTerm, the WT32 chip is hooked up to the USB port on a PC via a TTL-232R-3V3 cable. The Rx (receive) on the WT32 is connected to the Tx (transmit) on the TTL-232R-3V3 cable. Similarly, the Tx on the WT32 is connected to Rx on the TTL-232R-3V3 cable.

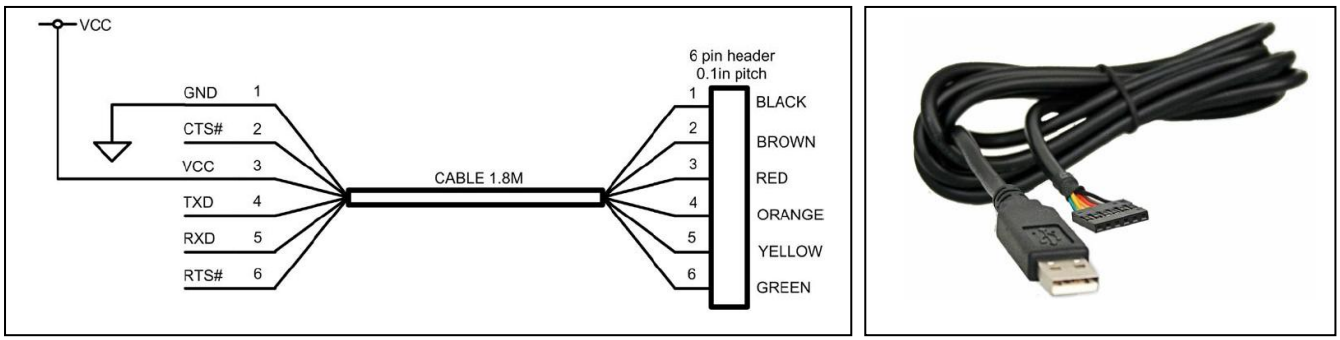


Figure 6.2. TTL-232R-3V3 block diagram and picture

Once the WT32 is set up with pairing data, the correct Baud rate 38400 and set to MUX mode so that it can be hooked up to the Sunspot. In order to do this we plug the Tx from the WT32 into D0 on the Sunspot, connect the Rx to D1 and ground to ground. The Sunspot is then used to create/handle commands to and from the WT32. This controls the data being sent and received from the phone and the Balance Board. The implementation details of how the Sunspot handles the communication and processing of data is discussed next.

6.1 Backend Implementation (Sunspot, WT32, Balance Board)

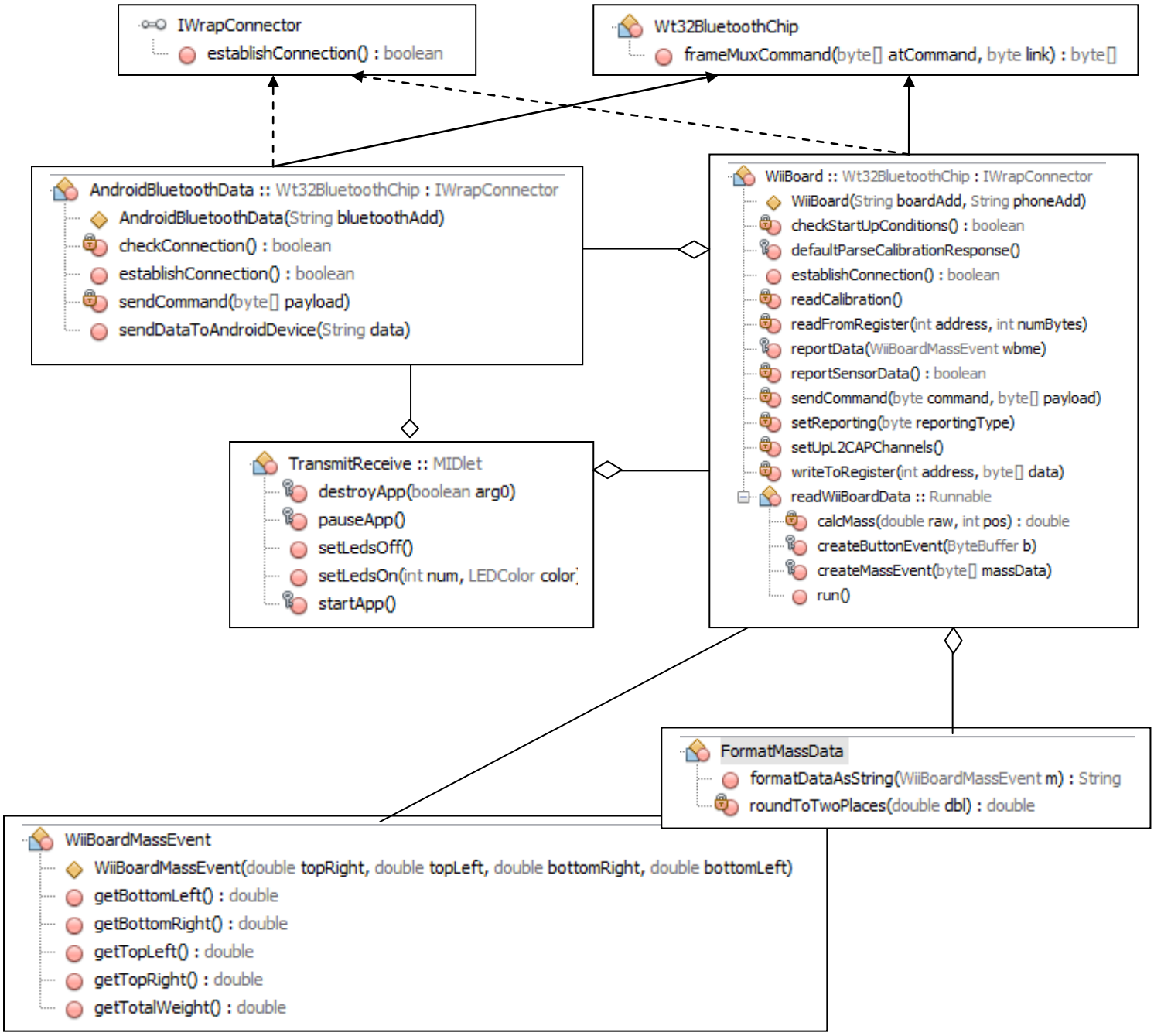


Figure 6.1.1. Backend Class Diagram

Connection Flow

The transmitReceive class is the main midlet class which initialises the AndroidBluetoothData and WiiBoard classes. The Communication channel (RFCOMM) with the phone is set up first. If the connection is successful, the 2 L2CAP channels (11 and 13) are set up to communicate with the Balance Board. Data can then be sent between devices.

The AndroidBluetoothData and the WiiBoard classes both implement the IWrapConnector Interface and extend from WT32BluetoothChip. The IWrapConnector interface just ensures that a correct connection is established before any further processing is done. The WT32BluetoothChip class contains the frame format for any commands that need to be sent to the WT32 when it is in MUX mode. Any classes sending or receiving Bluetooth commands or data will need to be wrapped in this frame and so extend this class. The details of this class are given below.

```
public class Wt32BluetoothChip {

    //byte link is the associated connection to send the command on.
    //0xff is control, 0x00 is link 1, etc.
    /* FRAME DESCRIPTION
    Length:         Name:       Description:                    Value:
    8 bits          SOF         Start of frame                  0xBF
    8 bits          LINK        Link ID                         0x00 - 0x08 or 0xFF (control)
    6 bits          FLAGS       Frame flags                     0x00
    10 bits         LENGTH      Size of data field in bytes     -
    0-800 bits      DATA        Data                            -
    8 bits          nLINK       {LINK} XOR 0xFF                 -
     * *
     */
    //private byte[] frameMuxCommand(byte[] atCommand, byte link) {
    public byte[] frameMuxCommand(byte[] atCommand, byte link) {

        //3 extra flags are needed for the frame in MUX mode
        byte framedCommand[] = new byte[5 + atCommand.length];
        //SOF
        framedCommand[0] = (byte) 0xbf;
        //Link
        framedCommand[1] = link;
        //frame flags
        framedCommand[2] = 0x00;
        //length of command
        framedCommand[3] = (byte) atCommand.length;
        //copy the payload into a framed array
        System.arraycopy(atCommand, 0, framedCommand, 4, atCommand.length);
        //nLink - end of frame
        framedCommand[framedCommand.length - 1] = (byte) (link ^ 0xff);

        return framedCommand;
    }


}
```

The AndroidBluetoothData class provides the implementation details for the establishConnection method. This method pairs with the Android phone and sets up the RFCOMM channel to transfer data. The implementation details are shown below:

```
public boolean establishConnection() {
    boolean connectionEstablished = false;

    //used as link id for control commands
    byte controlByte = (byte) 0xff;

    //pair with android device
    String pairDevice = "PAIR " + bluetoothAddress;
    String rfcommChannel = "call " + bluetoothAddress + " fa87c0d0-afac-11de-8a39-0800200c9a66 rfcomm";

    //get hex bytes for ascii commands
    byte pairCommand[] = pairDevice.getBytes();
    byte rfcommCommand[] = rfcommChannel.getBytes();

    //frame the commands
    byte framedPairCom[] = frameMuxCommand(pairCommand, controlByte);
    byte framedRfcommCom[] = frameMuxCommand(rfcommCommand, controlByte);

    //send the commands to wt32 via uart and allow time for setup
    System.out.println("PAIR DEVICE");
    sunBoard.writeUART(framedPairCom);
    Utils.sleep(2000);
    System.out.println("set up rfcomm");
    sunBoard.writeUART(framedRfcommCom);
    Utils.sleep(2000);
    //check connection was successful
    connectionEstablished = checkConnection();

    //FEEDBACK DISPLAY:
    if (connectionEstablished) {
        sendDataToAndroidDevice("initialising...");
    }
    //setup rfcommchannel with android device
    return connectionEstablished;
}
```

Once the connection is established, data can be sent to the Android phone using the sendCommand method which uses the super classes frameMuxCommand method. The send command takes an array of bytes as the command to be sent. In the case of the AndroidBluetoothData class, the command is sent over the RFCOMM channel, which will be the first link that is set up (link=0x00).

```
private void sendCommand(byte[] payload) {

    byte[] message = new byte[1 + payload.length];
    message[0] = 82;
    System.arraycopy(payload, 0, message, 1, payload.length);

    //These commands will be sent on RFCOMM channel (1st channel)
    byte link1 = (byte) 0x00;
    //frame the message to be sent.
    byte[] framedMessage = frameMuxCommand(message, link1);
    //send the message over UART
    sunBoard.writeUART(framedMessage);
    //System.out.println("SENT MESSAGE: "+new String(framedMessage));
}
```

Once this connection is successful the WiiBoard class establishes its connection to the Balance Board. It sets up the L2CAP channels 13 and 11 as the 2nd and 3rd links. After the channels are set up data can be sent on channel 11 (link 0x02).

```
public boolean establishConnection() {
    boolean connected = false;
    setUpL2CAPChannels();
    //data to the balance board will be sent on channel 1 (L2CAP 11)
    writeToRegister(0x04a40040, new byte[]{0});
    Utils.sleep(500);
    //set the balance board to continously send back data. Will be reset when board is disconnected
    setReporting(CONTINUOUS_REPORTING);
    Utils.sleep(500);
    //check if the board is connected and reporting data on the correct channel.
    connected = reportSensorData();
    return connected;
}
```

Communication to the Balance Board register informs the Balance Board of the requirement to connect to the extension (in this case the extension is the Balance Board itself). The command that needs to be sent to the Balance Board is:

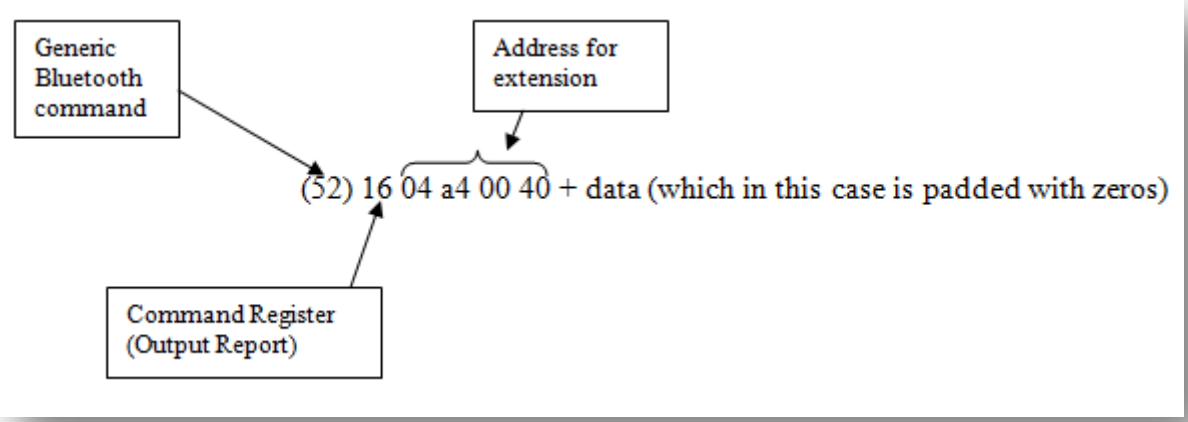


Figure 6.1.2 Command to Connect to Extension

Now the Balance Board is ready to listen to commands, so it is set to continuously report its data. This reporting mode needs to be reset every time a status report is sent back from the board. To set up continuous reporting it is necessary to send the following hex bytes to the board:

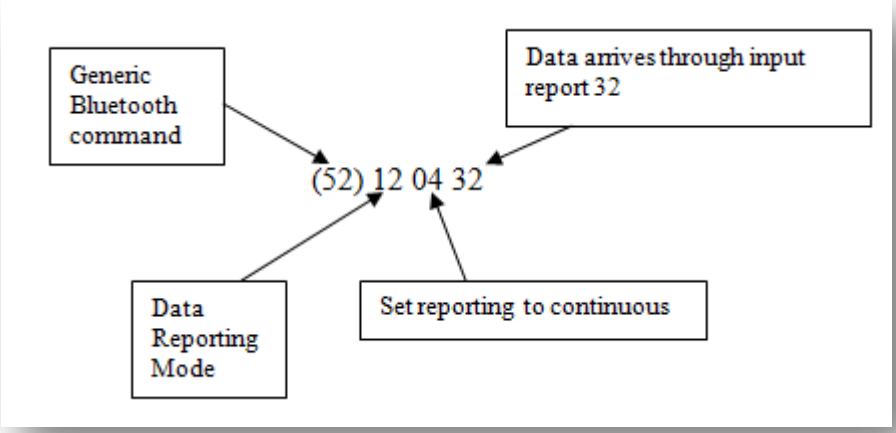


Figure 6.1.3. Set Reporting to Continuous Mode

Now sensor data can be reported. To do this it is necessary to read the calibration and any status reports that might be raised. If there are any status reports raised, the reporting mode will be reset and will need to be set to continuous again. Once the calibration values are read, the sensor data (coming in through report 32) can be interpolated with the calibration values. Each pressure sensor has two bytes of raw data (1 sensor reading) and six bytes of calibration data (3 calibration readings). The 3 calibration readings correspond to 0kg, 17kg and 34kg. It

is necessary to interpolate the sensor reading with the two calibration readings it falls between. The three calibration values for each sensor are then stored. There are three arrays of size four to store the data for each sensor. A byte array is read in and it is necessary to store double arrays. Since each calibration value is 2 bytes or 16bits, the first 8bits are masked and logically 'ANDed' with the second 8bits of the number to convert it to a double. This is done for all calibration values for each sensor.

```
calibration0 = new double[4];
calibration17 = new double[4];
calibration34 = new double[4];

// first packet of calibration data
int calib1 = 0;
for (int i = 0; i < 4; ++i) {

    calibration0[i] = (double) (((recDefault1[calib1] & 0xff) << 8) + (recDefault1[calib1 + 1] & 0xff));
    calib1 += 2;
}

for (int i = 0; i < 4; ++i) {

    calibration17[i] = (double) (((recDefault1[calib1] & 0xff) << 8) + (recDefault1[calib1 + 1] & 0xff));
    calib1 += 2;
}

int calib2 = 0;
for (int i = 0; i < 4; ++i) {
    calibration34[i] = (double) (((recDefault2[calib2] & 0xff) << 8) + (recDefault2[calib2 + 1] & 0xff));
}
```

The resultant calibration arrays can then be used to calculate the actual mass in kg by interpolating the pressure sensor value. The raw pressure sensor data is similarly retrieved and stored in array of doubles:

```
/**
 * Creates a WiiBoardMassEvent from raw data
 */
protected void createMassEvent(byte[] massData) {
    double rawTR, rawBR, rawTL, rawBL;
    double topRight, bottomRight, topLeft, bottomLeft;

    int massCount = 0;
    rawTR = ((massData[massCount] & 0xff) << 8) + (massData[massCount + 1] & 0xff);
    massCount += 2;
    rawBR = ((massData[massCount] & 0xff) << 8) + (massData[massCount + 1] & 0xff);
    massCount += 2;
    rawTL = ((massData[massCount] & 0xff) << 8) + (massData[massCount + 1] & 0xff);
    massCount += 2;
    rawBL = ((massData[massCount] & 0xff) << 8) + (massData[massCount + 1] & 0xff);

    topRight = calcMass(rawTR, TOP_RIGHT);
    topLeft = calcMass(rawTL, TOP_LEFT);
    bottomRight = calcMass(rawBR, BOTTOM_RIGHT);
    bottomLeft = calcMass(rawBL, BOTTOM_LEFT);


    WiiBoardMassEvent wbme = new WiiBoardMassEvent(topRight, topLeft, bottomRight, bottomLeft);
    //System.out.println("MASS EVENT: " + wbme.getTopRight() + "," + wbme.getTopLeft() + "," + w
    System.out.println("**********************");
    //System.out.println("TOP LEFT: "+wbme.getTopLeft());
    //System.out.println("TOP RIGHT: "+wbme.getTopRight());
    //System.out.println("BOTTOM LEFT: "+wbme.getBottomLeft());
    System.out.println("BOTTOM RIGHT: "+wbme.getBottomRight());
    reportData(wbme);

}
```

In order to do this, it is necessary to determine where the sensor data falls i.e. if the pressure value is 3618 (x) and calibration values for one sensor are [2293, 4004, 5725] (0y, 17y, 32y) then the value in kg can be worked out by the following equation.

17y * (x-0y) / (17y-0y) = 17kg * (3618-2293) / (4004-2293) = 13.2 Kg.

This equation is used on the calcMass method shown below:

```
/**
 * Calculates the Kilogram weight reading from raw data at position pos
 */
private double calcMass(double raw, int pos) {
    double val = 0.00;
    if (raw < calibration0[pos]) {
        //ignore the reading
    } else if (raw < calibration17[pos]) {
        val = 17 * ((raw - calibration0[pos]) / (calibration17[pos] - calibration0[pos]));
    } else if (raw > calibration17[pos]) {
        val = 17 + 17 * ((raw - calibration17[pos]) / (calibration34[pos] - calibration17[pos]));
    }
    return val;
}
```

Once the data is formatted to kg it can be sent to the Android phone for further processing. The data is formatted as a string of double values separated by tab space characters. An instance of the AndroidBluetoothData class is used to send the data to the phone.

```
protected void reportData(WiiBoardMassEvent wbme) {
    //dispatch the mass event so the sensor values can
    // sent to the phone
   //MassEventSet.setData(wbme);

   //send the data directly to the phone. The static methods were taking too long to get and send data
   FormatMassData fmd = new FormatMassData();
   String dataToSend = fmd.formatDataAsString(wbme);

   phoneConnection.sendDataToAndroidDevice(dataToSend);
   //this sleep call is important to allow the connection to finish sending before getting data
   //back from the balance board again. 10Hz is more than sufficient for human movement.
   Utils.sleep(100);
}
```

Once the data is sent to the phone it is handled there by the Android code and displayed on the GUI. The implementation details of this are handled in the next section.

6.2 Front End Implementation (Android phone)

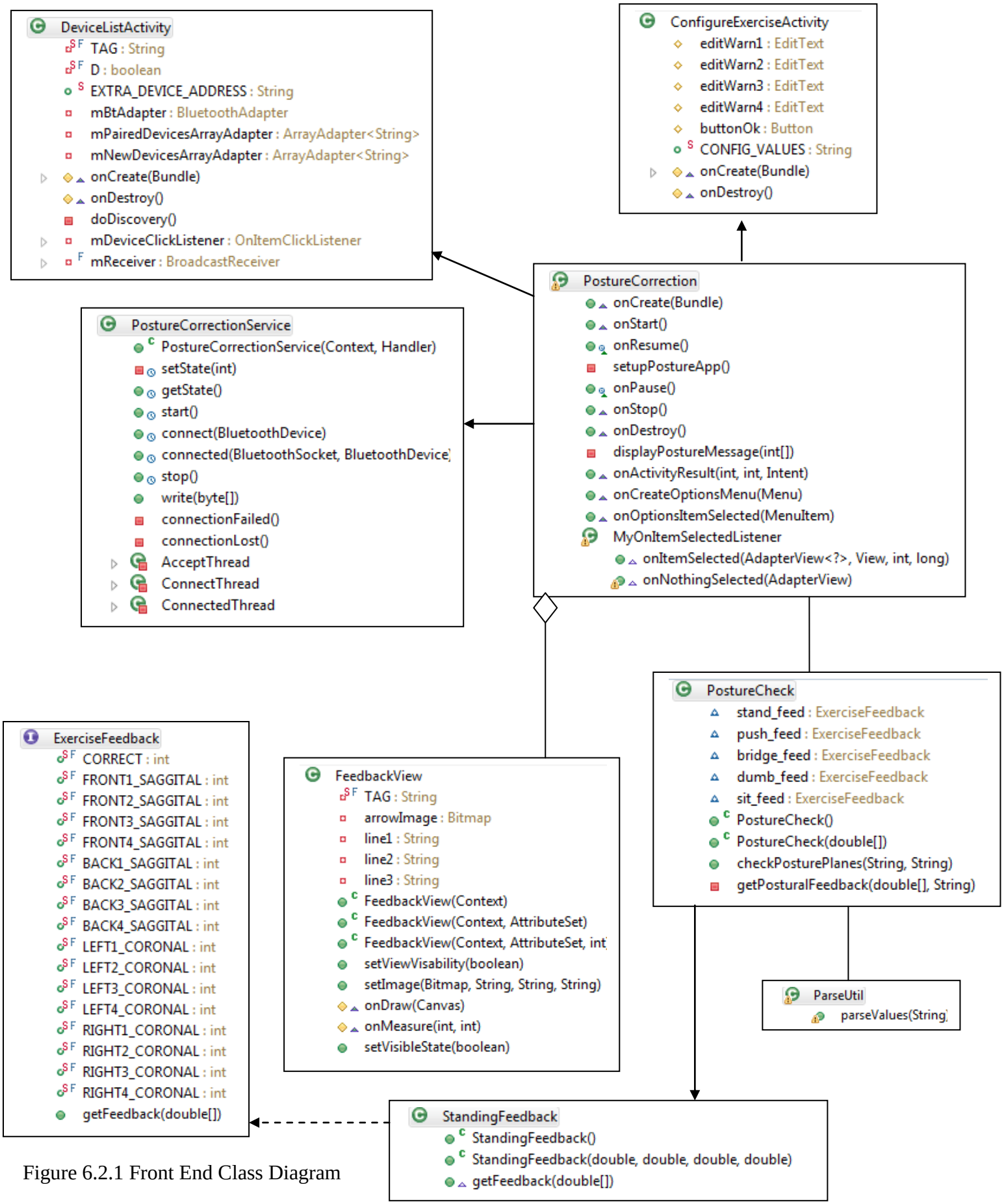


Figure 6.2.1 Front End Class Diagram

The PostureCorrection class is the main activity on the Android phone. All GUI components, listeners and handlers are initialised from this class. The PostureCorrection class extends Activity and overrides methods such as onCreate(), onStart(), onResume(), onStop(), onPause() and onDestroy(). In the onCreate method we set up GUI components such as the titles, the main layout and the spinner. The layout components are loaded from an XML file stored in the res – layout folder. Android automatically generates an R class file that can be used to access any resource components. An example of how a resource component can be used in this way is given here:

```
mFeedbackView2 = (FeedbackView) findViewById(R.id.view2);
```

A custom view called FeedbackView can be retrieved from resources with the above command assuming that the view has been entered into the XML file as follows:

```
<com.example.android.PostureCorrection.FeedbackView android:id="@+id/view2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
android:background="#DAAA"/>
```

The PostureCorrection class initialises the PostureCorrectionService to listen for incoming Bluetooth connections. It provides a handler to the service so that the PostureCorrection class can obtain messages from the handler, such as the sensor data being sent from the Sunspot.

```
// Initialise the PostureCorrectionService to perform Bluetooth connections
mPostureService = new PostureCorrectionService(this, mHandler);
```

In order to obtain messages from the handler it is necessary to create an instance of the Handler class. Here an instance with an anonymous inner class that overrides the handleMessage method is created. The switch statement deals with the various methods that may be obtained. The main case statement to look at here is the MESSAGE_READ. This will be continuously called because of the incoming formatted pressure sensor values obtained from the Sunspot.

```
@Override
public void handleMessage(Message msg) {
    switch (msg.what) {
    case MESSAGE_STATE_CHANGE:
        if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);
        switch (msg.arg1) {
        case PostureCorrectionService.STATE_CONNECTED:
            mTitle.setText(R.string.title_connected_to);
            mTitle.append(mConnectedDeviceName);
            mConversationView.setText("");
            break;
        case PostureCorrectionService.STATE_CONNECTING:
            mTitle.setText(R.string.title_connecting);
            break;
        case PostureCorrectionService.STATE_LISTEN:
        case PostureCorrectionService.STATE_NONE:
            mTitle.setText(R.string.title_not_connected);
            break;
        }
        break;
   case MESSAGE_READ:
        byte[] readBuf = (byte[]) msg.obj;

        mConversationView.setText("");
        // construct a string from the valid bytes in the buffer
        String readMessage = new String(readBuf, 1, msg.arg1);
        mConversationView.setText("Top Left\tTop Right\t\tBottom Left\tBottom Right\n");
        mConversationView.append(readMessage);

        int postureFeedback [] = pc.checkPosturePlanes(readMessage,currentExercise);
        displayPostureMessage(postureFeedback);

        break;
   case MESSAGE_DEVICE_NAME:
        // save the connected device's name
        mConnectedDeviceName = msg.getData().getString(DEVICE_NAME);
        Toast.makeText(getApplicationContext(), "Connected to "+ mConnectedDeviceName, Toast.LENGTH_SHORT).show();
        break;
   case MESSAGE_TOAST:
        Toast.makeText(getApplicationContext(), msg.getData().getString(TOAST),
                       Toast.LENGTH_SHORT).show();
        break;
    }
}
```

In the MESSAGE_READ case statement it sets up the TextView (mConversationView) and reads the pressure values in kg to display on the screen. It then calls a method entitled checkPosturePlanes using an instance of the PostureCheck class.

The PostureCheck class has two constructors. One empty constructor uses the default values that were discussed in the design chapter. The other constructor uses values that may have been entered in the Configuration Screen.

```
public PostureCheck () {
    stand_feed = new StandingFeedback();
    push_feed = new PushupsFeedback();
    bridge_feed = new BridgingFeedback();
    dumb_feed = new DumbellsFeedback();
    sit_feed = new SittingFeedback();
}

public PostureCheck (double [] config) {
    stand_feed = new StandingFeedback(config[0], config[1], config[2], config[3]);
    push_feed = new PushupsFeedback(config[0], config[1], config[2], config[3]);
    bridge_feed = new BridgingFeedback(config[0], config[1], config[2], config[3]);
    dumb_feed = new DumbellsFeedback(config[0], config[1], config[2], config[3]);
    sit_feed = new SittingFeedback(config[0], config[1], config[2], config[3]);
}
```

The config array refers to the fields on the configuration screen. The first field on the configuration screen refers to config [0] and so on. These values are then used in the calculation of the warning levels for the different exercises.

In order to calculate the warning levels the sensor data coming from the Sunspot is parsed. This will be a string of doubles parsed into an array of doubles. The exercise type is sent as a parameter based on what the user has chosen from the Spinner (drop down list). For example if the exercise chosen is standing, we call the StandingFeedback to provide the calculation of the posture.

```
private int [] getPosturalFeedback (double [] sensorvals, String exerciseType) {
    int [] feedbackVals = new int [2];

    if (exerciseType.equals("Standing")) {
        feedbackVals = stand_feed.getFeedback(sensorvals);

    } else if (exerciseType.equals("Bridging")) {
        feedbackVals = bridge_feed.getFeedback(sensorvals);

    } else if (exerciseType.equals("Push ups")) {
        feedbackVals = push_feed.getFeedback(sensorvals);

    } else if (exerciseType.equals("Dumbells")) {
        feedbackVals = dumb_feed.getFeedback(sensorvals);

    } else if (exerciseType.equals("Sitting")) {
        feedbackVals = sit_feed.getFeedback(sensorvals);

    }
    return feedbackVals;
}
```

Each Exercise has its own class. This makes it easy to modify each exercise so that each has its own individual characteristics. Each Exercise class implements the ExerciseFeedback interface and provides a concrete getFeedback method. The getFeedback method works out the weight variation of the user. It checks if they are leaning backwards, forwards, left or right. This is calculated based on their total body weight. Depending on how much variation there is, sagittal and coronal values are returned. This corresponds to the different warning levels displayed to the user.

This process involves working out the total body weight and what the weights on the different sides of the board should be for neutral posture. From research associated with this study it is clear that neutral posture requires that the angle from just below the hip to the shoulder should be approximately 177 degrees [10]. In order to achieve this, a person's weight should be evenly dispersed over the four pressure sensors. This information is store in a double array.

```
@Override
public int [] getFeedback(double[] vals) {
    int [] result = new int [2];
    int sagRes = 0;
    int corRes = 0;
    //values are TL, TR, BL, BR
    double totalWeight = vals[0]+vals[1]+vals[2]+vals[3];
    double initialOffset = (totalWeight * initialOffset_allowance);
    double offset2 = (totalWeight * offset2_allowance);
    double offset3 = (totalWeight * offset3_allowance);
    double offset4 = (totalWeight * offset4_allowance);

    if (vals[0] > 0 && vals[1] > 0 && vals[2] > 0 && vals[3] > 0) {
        //for neutral standing posture the weight should be
        //evenly dispersed among the sensors
        double evenSideWeight = totalWeight/2;
        double frontTotal = vals[0]+vals[1];
        double backTotal = vals[2]+vals[3];
        double leftTotal = vals[0]+vals[2];
        double rightTotal = vals[1]+vals[3];
```

With this information it can be worked out where the weight is dispersed and by how much it is dispersed in a given direction. Depending on the sensitivity of the application (shown as offsets), different warning levels are returned. For example, the sagittal plane values are worked out in the following way:

```
//Check front and back values are correct
if (frontTotal > evenSideWeight + offset4) {
    sagRes = FRONT4_SAGGITAL;
} else if (frontTotal > evenSideWeight + offset3) {
    sagRes = FRONT3_SAGGITAL;
} else if (frontTotal > evenSideWeight + offset2) {
    sagRes = FRONT2_SAGGITAL;
} else if (frontTotal > evenSideWeight + initialOffset) {
    sagRes = FRONT1_SAGGITAL;
} else if (backTotal > evenSideWeight + offset4) {
    sagRes = BACK4_SAGGITAL;
} else if (backTotal > evenSideWeight + offset3) {
    sagRes = BACK3_SAGGITAL;
} else if (backTotal > evenSideWeight + offset2) {
    sagRes = BACK2_SAGGITAL;
} else if (backTotal > evenSideWeight + initialOffset) {
    sagRes = BACK1_SAGGITAL;
} else
    sagRes = CORRECT;
```

Once the warning level values are returned (in the form of a 1-dimensional double array of size 2) to the PostureCorrection class, they are used in the MESSAGE_READ case statement shown above. The displayPostureMessage method is called from this switch statement and it takes the warning levels array as a parameter. Depending on the warning levels it will display the appropriate arrow and text as feedback to the user. The warning levels are Constant values and refer to different arrows. This is discussed in more detail in the Design chapter. Below is a snippet of the displayPostureMessage method showing only how sagittal feedback is displayed.

```
private void displayPostureMessage(int[] postureFeedback) {
    int frontAndBack = postureFeedback[0];
    int sideToSide = postureFeedback[1];

    Log.d(TAG, "displayPostureMessage(), frontAndBack: "+frontAndBack+", sideToSide: "+sideToSide);
    //may change to 2 switch statements
    //check for front and back corrections
    if (frontAndBack == 1) {
        mFeedbackView1.setImage(mArrow1, fl1,fl2,fl3);
    } else if (frontAndBack == 2) {
        mFeedbackView1.setImage(mArrow2, fl1,fl2,fl3);
    } else if (frontAndBack == 3) {
        mFeedbackView1.setImage(mArrow3, fl1,fl2,fl3);
    } else if (frontAndBack == 4) {
        mFeedbackView1.setImage(mArrow4, err_fl1,err_fl2,err_fl3);
    } else if (frontAndBack == 5) {
        mFeedbackView1.setImage(mArrow5, bl1,bl2,bl3);
    } else if (frontAndBack == 6) {
        mFeedbackView1.setImage(mArrow6, bl1,bl2,bl3);
    } else if (frontAndBack == 7) {
        mFeedbackView1.setImage(mArrow7, bl1,bl2,bl3);
    } else if (frontAndBack == 8) {
        mFeedbackView1.setImage(mArrow8, err_bl1,err_bl2,err_bl3);
    } else if (frontAndBack == 0) {
        mFeedbackView1.setImage(mArrow0, correctPostureSag,correctPosture2,"");
    } else {
        //do nothing for false values
    }
```

This method uses a FeedbackView to display the real time corrective feedback to the user. The FeedbackView is a custom View that was developed for both Sagittal and Coronal feedback. Different instances of the same view are used. FeedbackView extends Androids View class and overrides the onDraw and onMeasure methods. A canvas is needed to display the feedback as the graphics will need to be refreshed rapidly to the screen. The overridden onDraw method takes care of this.

```
@Override
protected void onDraw(Canvas canvas) {

    Paint paint = new Paint();
    paint.setAntiAlias(true);
    paint.setTextSize(12);
    paint.setTypeface(Typeface.SERIF);

    if(arrowImage != null)
    {
            canvas.drawBitmap(arrowImage, 0, 0, null);
            canvas.drawText(line1, 40, 40, paint);
            canvas.drawText(line2, 40, 55, paint);
            canvas.drawText(line3, 40, 70, paint);
    }
    else {
        //canvas.drawBitmap(defaultArrow, 0, 0, null);
        canvas.drawText(line1, 40, 40, paint);
        canvas.drawText(line2, 40, 55, paint);
    }

    canvas.restore();
}
```

The arrows are displayed according to the dimensions set in the overridden onMeasure class. The dimensions given are 150 X 80. The canvas will be redrawn when the invalidate method is called. The invalidate method is called from the setImage method in FeedbackView. The setImage is invoked from the displayPostureMessage as shown above.

The feedback views will be refreshed roughly every 10ms. The Sunspot samples are sent to the Android phone about every 10ms. This does not include the time it takes to read the data from the Balance Board and process it on the Sunspot. Also there may be a delay in the values that are read from the Balance Board. Therefore a change in the Baud rate can change the response time. Also on a few occasions the Balance Board reports 0kg which will not be taken into account by our system. The end result is that the user is given pseudo real time corrective feedback based on their current posture.

7. Evaluation

Experiments were carried out for an evaluation of the developed Posture Correction system. The main issues that caused problems at different stages of the design and development are outlined. The future work section outlines improvements and extensions that could be made to the Posture Correction system. The conclusion to the Posture Correction application highlights the main points that were targeted and implemented to provide an overall effective solution.

Experiments were carried out on the exercises that were designed in collaboration with two physiotherapists, David McGrath and Aileen Shaw, from Mayo General Hospital, Castlebar. The exercises identified are the five most commonly prescribed exercises for posture rehabilitation. The different exercises have specific focuses for areas of recuperation. The exercises are described below and the effects of these exercises using the developed posture correction application are discussed.

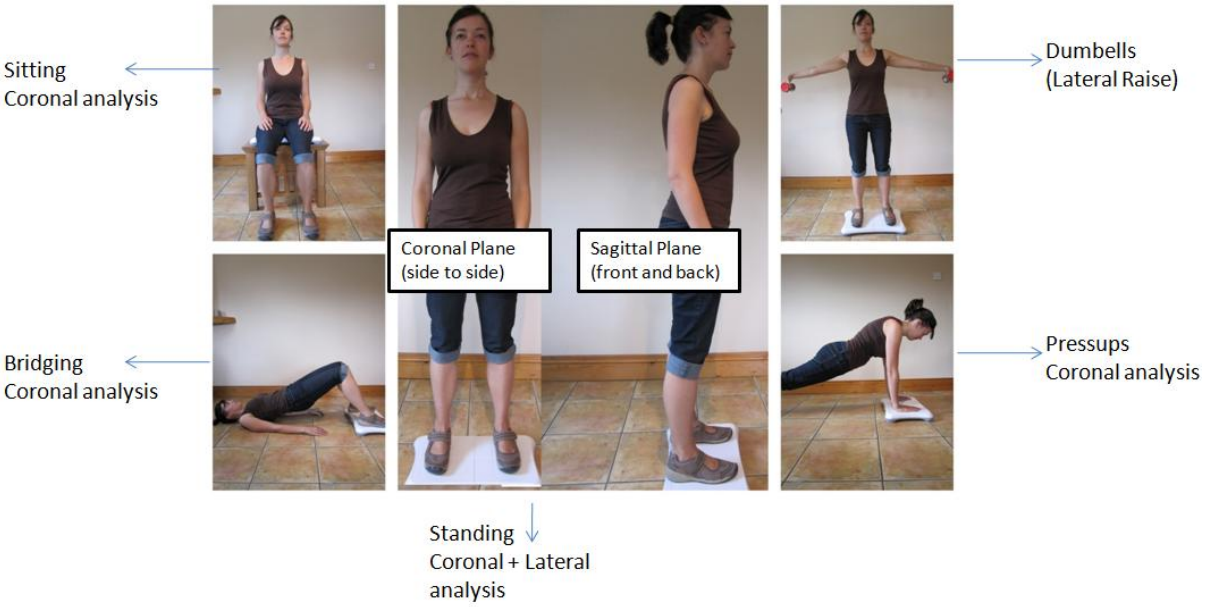


Figure 7.1. Exercises

Standing

This is the most common exercise. It can be used for post-stroke patients or patients suffering from a variety of neurological disorders. The internal motor neurons can be affected by neurological disorders. The patient's sense of posture is skewed and needs to be stimulated by external feedback such as the Posture Correction system developed through this study. The patient's internal motor neurons need to be retrained.

The person will try to stand up straight, shoulders upright and face forwards. He/She will try to keep this stance, slightly adjusting his/her stance according to feedback.

Dumbells

This dumbells exercise is similar to standing, except that light weights called dumbells are utilised. This is another core posture correction exercise. It can be used for people whose posture has improved after previous rehabilitation from different exercises.

The person will try to stand up straight, shoulders upright, face forwards and raise his/her arms simultaneously to the right and left. The dumbbells exercise also exploits lateral muscles.

Sitting

Sitting is a very popular exercise to improve a patient's posture for people affected by back/neck problems. The Balance Board is placed on a hard surface at about knee height. The patient sits upright in the centre of the board and is given coronal feedback based on their sitting posture.

Push-ups

Push-ups/press-ups are popular exercises for rehabilitation of certain sports injuries such as a dislocated shoulder. After the shoulder has strengthened and the surrounding muscles are starting to get stronger push-ups exercise can be used to determine if a person is putting too much weight on one side. It provides feedback for the person to put their weight evenly on the two sides. The user places a hand on either side of the Balance Board and keeping his/her back straight, he/she moves his/her body up and down slowly.

Bridging

Bridging is a popular technique used to help people recover their regular walking pattern after an operation such as a hip replacement. It provides corrective feedback on the coronal planes. Bridging is a technique whereby a person lies with his/her back on the ground. He/She places the soles of his/her feet on the Balance Board (so that his/her knees are up in the air.). They then move their hips slowly up and down. Real time feedback is given to the user to inform him/her if he/she is leaning too far left or too far right.

7.1. Experimentation

With the previous exercises in mind experiments with 4 volunteers were undertaken as part of this study. The volunteers were asked to answer a set of simple questions after they used the Posture Correction application. These questions were based on some of Nielson's usability heuristics [31]. The 4 volunteers were all healthy subjects in their late 20s. The investigation did not experiment with people who had any neurological disorders or sports injuries. The volunteer group was limited in numbers and age group. For these reasons these results are not conclusive.

Question 1: Is the application easy to use?

Volunteer 1: "Yes. It's very straightforward and exercises are explained appropriately"

Volunteer 2: "Yes. Everything you need is on the same screen"

Volunteer 3: "Yes. Background picture informs me of the type of exercise."

Volunteer 4: "Yes. Very intuitive."

Question 2: Is the application responsive enough for posture identification and correction?

Volunteer 1: "Yes. When I move it moves."

Volunteer 2: "Yes. It's very sensitive to my movements"

Volunteer 3: "Yes. When I move forward the screen shows me straight away"

Volunteer 4: "No. If I move very quickly the application doesn't update immediately"

Question 3: Are you surprised with your natural posture?

Volunteer 1: "Yes. I lean a bit forward."

Volunteer 2: "No."

Volunteer 3: "Yes. I lean slightly forward and left"

Volunteer 4: "Yes. Only if the application is configured to be more sensitive, I find myself leaning forward slightly"

Question 4: Do you think the application would help to improve any posture abnormalities?

Volunteer 1: "Yes. It would definitely help me to understand if I am carrying out an exercise correctly"

Volunteer 2: "Yes. But it would be better with sounds"

Volunteer 3: "Yes. But you would want somebody to hold the phone while doing the exercises"

Volunteer 4: “Yes”

Question 5: Would you use this application at home as a form of rehabilitation?

Volunteer 1: “Yes. It’s easy and more fun than conventional exercises”

Volunteer 2: “Yes. I can use it at home with my own Balance Board and phone”

Volunteer 3: “Maybe. I prefer more hands on approaches.”

Volunteer 4: “Yes. The more the exercises are carried out the more it would promote a more rapid recovery for me”

Question 6: Is the application easy to configure?

Volunteer 1: “No. Don’t know what amount to enter. Maybe it should be updated by a professional”

Volunteer 2: “No. A healthcare professional should enter percentages.”

Volunteer 3: “No. I don’t know what the boxes refer to and what values to put in these boxes.”

Volunteer 4: “Yes. If you know what values to use it is very easy to configure”

7.2. Problems Encountered

- Communicating via WT32 took a while. Learning the iWrap commands and how they work overcame this issue.
- The Bluetooth transmission needed to take place using MUX mode (on the WT32) to prevent the need of changing from data to command mode and vice versa. However there were initial problems with MUX mode as it requires hex commands, not Ascii. Once the Chip was in MUX mode I could not communicate with it. This problem was overcome by downloading BGTerm software which automatically converts Ascii commands to hexadecimal.
- MUX mode required a special frame that commands needed to be wrapped in. BGTerm automatically wrapped the commands. It was necessary to implement this frame into our Sunspot code.
- There was an initial difficulty finding out how to communicate to the HID profile on the Balance Board device. 2 L2CAP channels were used; one for sending data and one for receiving data.
- It was difficult find the correct control registers on the Balance Board to send commands to. This was overcome with a combination of help from Wiibrew [29] and trial and error.

- There was an issue initially receiving commands. It was necessary to re-select the receiving L2CAP channel after commands had been sent over the outgoing L2CAP channel.
- It was difficult to try and parse data coming from the Balance Board. There were different reports triggered, data UART overrun error causing different order of bytes. This was overcome using a large buffer and only selecting a fraction of those bytes discarding the rest.
- The Sunspot application was using an imported library for bytebuffer methods. This was causing a nullpointerexception. The exception could not be recreated using j2se. Needed a work around with byte arrays.
- Needed to create pairing password to pair with phone. Once the phone added wt32 to devices list the pairing authorization needed to be turned off as the Balance Board could not handle it.
- Updating graphics wasn't quick enough with a toast notification. Needed to use a canvas to update graphics quickly.
- Data wasn't reported quickly enough from the WT32. A faster Baud rate was needed. When the Baud rate was increased the system crashed. The receive buffer then needed to be increased.

7.3. Future Work

There are multiple enhancements that could be done to improve this Posture application. Given the time frame it was impossible to include everything that was needed. Some of the future work opportunities are listed below.

Independent Balance Board Connection

For the purposes of the developed Posture Correction System, the addresses of the Wii Balance Board and the Android phone were hardcoded in the Sunspot code. It is possible to create an independent connection based on a Bluetooth inquiry because the names of all Balance Boards are similar (e.g. Nintendo RVL-WBC-01). This would allow a connection to any Balance Board without any code modifications. The connection between the phone and the Sunspot could be carried out on the phone through a device search. This would allow the removal of the hardcoded Android phone address.

Logging

As one of the priority additions to the application logging should be enabled. There is an option in the menu of the application to turn on logging. However this currently does not do anything. In order to analyse data after a session it would be very useful to log the data and then it could be imported into a spreadsheet or a graphing tool to produce an overall view of the posture abnormalities. This would not require a large amount of work as the application is set up to allow this.

Persistent Configuration Settings

When the application is configured to a particular sensitivity, this configuration only lasts as long as the application is still running. Every time the application is started it will revert to the default values. A change could be made to the application to store the configuration values to persistent data. Then this data could be used as the sensitivity values upon start-up.

Sound

For a person's posture not to be affected by using our Posture Correction application, they should ideally not be holding the phone. The user should place the phone on a stand or get somebody to hold the phone for him/her. This is not always practical. To overcome this problem, sound notifications should be introduced. A different type of beep, tone or narrative could be used. Different types of sounds need to be used depending on whether a user is leaning forward, back, right or left. The tone of the sound for each direction could then be increased depending on the warning level. There may be problems with the sound if there are rapid changes in posture. Also in exercises where both sagittal and coronal planes are being used the sound bites may interfere with each other when feedback for both planes is given at the same time. The sound notifications may need to be interlaced with a timing period between notifications.

Sensor Insoles

Since our Posture Correction application is extensible it should be relatively easy to hook up external pressure sensors to the sunspot. These could then be glued or inserted into an insole. Once the insole is placed in the running shoe the sensors would need to be re-calibrated. Different effects of pressure on the different sensors according to where they are in the insole would also need to be analysed.

Having the pressure sensors in the insole would greatly increase the amount of exercises that the posture application can provide feedback for. For instance, real-time corrective feedback (in the form of sound notifications) could be performed for walking, running, skiing, golfing exercises. It also would provide a great platform on which to build a full Body Sensor Network.

Body Sensor Network

The Posture Correction application could be built upon using multiple microprocessors and sensors that could communicate together to give an overall picture of Body movement. It could be used to log and report minute posture abnormalities causing issues with particular muscle groups.

7.4 Personal Career and Development Record

This section outlines what we have taken from the project in terms of valuable experience and skills that will help further my career.

Balance Board and its control registers.

The WiiBrew site [29] provides a lot of useful information as to what commands need to be sent to the Balance Board and what reports will be retrieved. However it wasn't always received smoothly. Some data was initially received in separate buffers. Some data reports were not received when they were supposed to be. Accessing the reports from the Balance Board involved a lot of trial and error.

Hardware - WT32

Due to lack of expertise in hardware, wiring up the WT32 provided a challenge. Initially a connection was made by wrapping the wire around the connections, which wore away the metal on the chip. This caused problems which were not readily identifiable due to an intermittent connection. When pin connectors were soldered to the chip, this provided a solid connection.

Programming in Android.

Learning how to program in Android was a great experience. It's a useful platform to develop on. The layout is different to what I'm used to but provides a great structure to the project.

For example GUI components written in XML are kept separate to underlying code. Any strings can also be entered in an XML and stored in the resource folder. Separate layouts can also be written in XML and used multiple times. All these are considered as resources and are accessible through a generated class file named ‘R’. The Smart Soles project consists of the major components of any Android project such as Activities, Services, and Intents. This was a very valuable skill to learn and has provided valuable experience that may be used in future work places.

iWrap firmware commands

The Bluegiga WT32 Bluetooth chip comes preinstalled with its own firmware known as iWrap. This is a like a functional language that provides us with all the major functionality to communicate over Bluetooth such as setting up channels, inquiry, pairing, setting Baud rates, data modes, command modes, MUX modes. This required learning some iWrap commands and identifying how they worked. Although iWrap is unique to Bluegiga products [27] it is a useful skill to know as many handheld devices use such a Bluetooth chip.

7.5. Conclusion

Based on our research we came up with some major points that the developed application needed to be a success. The major issues identified were:

1. Ease of use, including setup
2. Suitable for installation on a person’s body
3. Having wireless capability
4. Capable of being used in any indoor environment.
5. Be as non-intrusive as possible.
6. Be highly configurable
7. Be reliable
8. Provide instant corrective feedback
9. Be cost effective

1. The set up of the Smart Soles system is very straight forward. It just requires opening the application on the phone, resetting the Sunspot and synchronising the Balance Board. The initialisation process only takes a matter of seconds.

2. The Sunspot Microprocessor can easily be installed on a person's body due to its small size. This provides potential for the system to be extended into a full Body sensor Network.

3. Bluetooth was chosen as the communication technology. The WT32 Bluetooth chip provides the system with wireless capability and potentially allows the system to be connected to millions of Bluetooth devices.

4. The system is primarily suited to any indoor environment. The main advantage of this is that a person can use this system in his/her own home promoting more rapid rehabilitation.

5. The wireless capabilities of the Balance Board, WT32 and Android phone enables the system to be non-intrusive. The user's body movements are not obstructed in any way by the technologies used in this system.

6. The application provides a Configuration option through its menu. This allows users to configure the system with any sensitivity values they want. Based on the survey carried out a user may not know what values to configure the system with. A physiotherapist may suggest values to the user based rehabilitation progress. This way the application can be adapted/tuned to each individual based on the severity of their disorder.

7. The Balance Board is a tried and trusted commercial product. It provides a reliable reading from the four pressure sensors [32]. All our posture correction feedback is based on these values being reliable.

8. Instant real time feedback is a major advantage of the system over other cost effective solutions. The speed at which the Balance Board reports are produced, the Sunspot processing speed and the WT32s high Baud rates all contribute to this real time feedback. This allows the user to re-train their internal motor neurons to perform correctly again.

9. The Balance Board provides a cheap alternative to force plates [32]. The Android platform is getting more and more popular with loads of aggressively priced Android handsets available on the market [33]. This provides the basis for a very cost effective system.

Based on the issues addressed above, there is a huge potential for a system just like this. It has the potential to reach a mass market of consumers given the cost effective solutions that were implemented. Using the Balance Board as a prototype gives people the potential to use technology that may be readily available to them in their home. The Wii Balance Board is a very cheap alternative to commercial force plates and just as good in performance [32].

The development of this Posture Correction application has been a rewarding experience. As well as gaining a lot of knowledge in the development of this application, it has provided innovative ideas that I may pursue in the future.

8. Appendices

8.1. Instructions on how to use the Posture Correction Application

- Start Posture Correction Application on Android phone
- Start Sunspot by pressing reset button
- When the phone displays the WT32 device has been connected, press the red sync button on the Balance Board immediately. (This can be found under the battery cover on the bottom of the Balance Board)
- If the green LEDs light up on the Sunspot the connections were successful.
- Now stand on the board or pick an exercise for real-time posture correction feedback.

8.2. Setting up the Development Platform

- Download and install the latest version of the Netbeans IDE (my version = 6.8).
- Download and install the latest version of the Eclipse IDE (my version = 3.4.2).
- Download and install the latest Sunspot SDK. (My version = red).
- Download and install the latest Android Development Kit (My Version = Platform 2.2, API Level 8)
- Download and install the Android Developer Tools (ADT) plugin for Eclipse (my version = 0.9.7)
- Download Bluegiga BGTerm application for the WT32. This may be used to change the settings on the WT32 Bluetooth chip.
- Wire up the WT32 to the Sunspot
    - Attach Tx to D0 on the Sunspot
    - Attach Rx to D1 on the Sunspot
    - Attach Ground to gnd on the Sunspot
    - Attach BTEN on WT32 to Vcc on WT32
    - Attach Ground and Vcc from WT32 to AA batteries (This means there are two wires connected to Ground and two wires connected to Vcc on the WT32)
- Now Sunspot code can be opened as a Netbeans project and Android phone code can be opened as a new android project from an existing source in Eclipse.

9. Bibliography

[1] G.-Z. Yang. (2006, Feb 2010). *Body Sensor Networks (1 ed.). 1*.

[2] P. Bonato, "Advances in wearable technology and applications in physical medicine and rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 2, p. 2, 2005.

[3] L. Nashner, "Adapting reflexes controlling the human posture," *Experimental Brain Research*, vol. 26, pp. 59-72, 1976.

[4] H. Labiod, *et al.*, Wi-Fi, Bluetooth, Zigbee and WiMAX: Springer Verlag, 2007.

[5] D. Jeong, *et al.*, "Classification of Posture and Movement Using a 3-axis Accelerometer," presented at the International Conference on Convergence Information Technology, vol. 1 p. 837-844 2007.

[6] T. O'Donovan, *et al.*, "A Context Aware Wireless Body Area Network (BAN)," presented at the 3rd International Conference on Pervasive Computing Technologies for Healthcare, London, UK, 2009.

[7] M. Zhang and A. Sawchuk, "A Customizable Framework of Body Area Sensor Network for Rehabilitation," presented at the 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies (Isabel), Bratislava, Slovak Republic, p.1-6, 2009.

[8] A. Pentland, "Healthwear: medical technology becomes wearable," *Computer*, pp. 42-49, 2004.

[9] J. Gil-Gómez, *et al.*, "Nintendo Wii Balance Board for Balance Disorders," presented at the Virtual Rehabilitation International Conference, p. 213-213, 2009

[10] P. O'Sullivan, *et al.*, "The effect of different standing and sitting postures on trunk muscle activity in a pain-free population," *Spine*, vol. 27, p. 1238, 2002.

[11] I. Stancic, *et al.*, "Stability in human postural control," in *Engineering in Medicine and Biology 27th Annual Conference*, Cavtat, Croatia, 2006, pp. 29-31.

[12] S. Ito and H. Kawasaki, "A standing posture control based on ground reaction force," in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000, pp. 837-844.

[13] N. GOTODA, *et al.*, "Supporting Skill Awareness for Runners through Wireless Sensor Network," presented at the Proceedings of the 17th International Conference on Computers in Education [CDROM]. , Hong Kong, 2009.

[14] A. Kiefer, *et al.*, "Synergy of the human spine in neutral postures," *European Spine Journal*, vol. 7, pp. 471-479, 1998.

[15] G. de Haan, *et al.*, "Using the Wii Balance Board™ as a low-cost VR interaction device," in *Proceedings of the 2008 ACM symposium on Virtual reality software and technology* 2008, pp. 289-290.

[16] S. Melzi, *et al.*, "The Virtual Trainer: Supervising Movements Through a Wearable Wireless Sensor Network," presented at the Conference on 6th Annual IEEE Communications Society for Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops '09. , 2009.

[17] A. Tognetti, *et al.*, "Wearable kinesthetic systems for capturing and classifying body posture and gesture," in *Engineering in Medicine and Biology 27th Annual Conference*, Shanghai, China, 2005, pp. 1012-1015.

[18] E. Jovanov, *et al.*, "A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 2, p. 6, 2005.

[19] A. Faheem, "A Wireless Body Area Sensor Network for Posture Detection," presented at the Proceedings of the 11th IEEE Symposium on Computers and Communications, 2006.

[20] I. Akyildiz, *et al.*, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, pp. 393-422, 2002.

[21] M. Benocci, *et al.*, "A wireless system for gait and posture analysis based on pressure insoles and Inertial Measurement Units" presented at the 3rd International Conference on Pervasive Computing Technologies for Healthcare, p. 1-6, 2009. PervasiveHealth 2009. , London, UK, 2009.

[22] R. Johansson, *et al.*, "Adaptation of multi-joint movements during postural disturbances," in 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2004. IEMBS '04. , 2004, pp. 149-152.

[23] Y. Zhang, *et al.*, "Mobile learning with bluetooth-based E-learning system," in Mobile Technology, Applications and Systems, 2005 2nd International Conference on 2005, pp. 15-17.

[24] D. Olguin, *et al.*, "Capturing Individual and Group Behaviour with Wearable Sensors," presented at the AAAI Spring Symposium 2009 in Human Behaviour Modelling, Stanford, CA. . 2009.

[25] Nintendo. *Nintendo Wii Fit*. http://wiifit.com/, retrieved March 2010

[26] Sun Microsystems. *Sunspot World*. http://www.sunspotworld.com/, retrieved March 2010

[27] Bluegiga. *WT32 Bluetooth Audio Module*. http://www.bluegiga.com/WT32_Bluetooth_Audio_Module, retrieved March 2010

[28] Wiimote – *Wiibrew*. http://wiibrew.org/wiki/Wiimote, retrieved July 2010

[29] Wii Balance Board. *Wiibrew*. http://wiibrew.org/wiki/Wii_Balance_Board#Extension_Controllers, retrieved July 2010

[30] TTL232R3V3, *Future Technology Devices International Ltd.* http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_TTL-232R_CABLES.pdf, retrieved July 2010.

[31] Nielsen, J., and Molich, R. Heuristic evaluation of user interfaces, *Proc. ACM CHI'90 Conf.* (Seattle, WA, 1-5 April), 249-256, 1990.

[32] Clark R. A., Bryant A. L., Pua Y., McCrorz P., Bennell K., & Hunt M. "Validity and reliability of the Nintendo Wii Balance Board for assessment of standing Balance". Gait & posture. Volume 31(3), pp. 307-310, 2010.

[33] M. Murphy, "The Big Picture," *Beginning Android 2*, pp. 1-4, 2010.

[34] International Business Machines Corp, "*Eclipse Technical Platform Overview*", http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf, retrieved August 2010.