UNIVERSITY OF CAPE TOWN

UNDERGRADUATE THESIS

Development of a Wearable Motion-Tracking Device and Corresponding Smartphone Application

Author:
Alexandra BARRY

Supervisor:
Dr. Simon WINBERG

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science in Mechatronic Engineering

in the

Research Group Name
Department of Electrical Engineering

October 22, 2018

Declaration of Authorship

I, Alexandra BARRY, declare that this thesis titled, "Development of a Wearable Motion-Tracking Device and Corresponding Smartphone Application" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 22/10/2018

UNIVERSITY OF CAPE TOWN

Abstract

Engineering and the Built Environment
Department of Electrical Engineering

Bachelor of Science in Mechatronic Engineering

Development of a Wearable Motion-Tracking Device and Corresponding Smartphone Application

by Alexandra BARRY

This thesis details the design and prototyping of a consumer-accessible motion-tracking, wearable sensor device and a corresponding Android application. The purpose of the device is to record motion data relating to a specific component of an athlete's technique or performance, such as the speed and motion of a golf swing or the stroke technique of a swimmer. This data is to be represented in a useful, real-time display on a mobile application to help the athlete, or their coach, improve their technique and overall performance.

The sensing device was built using a Nordic nRF52832 Bluetooth Low Energy System on Chip, and an InvenSens 9-Degree-of-Freedom Inertial Measurement Unit. The Android application was built using the Flutter cross platform mobile application Software Development Kit.

Each element added into the design was tested through the development phase.

The objectives were loosely met, as receiving data from only two BLE characteristics was achieved and the device and application developed are of a very basic level and require further development and refinement to become useful in their purpose.

Acknowledgements

There are several people I would like to thank for the support they provided me with along the course of this project, whether personal or project-related.

Dr Simon Winberg, my supervisor, for his consistent availability, incredible organisation, hand-crafted resources and expert guidance.

My parents: without whom, need I say, I would not be here.

To my extensive family, including my best friend Nicola, who have never stopped believing in me.

And to my incredible colleagues, without whom reaching this milestone would never have been possible.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	**A**nalog to **D**igital **C**onverter
BLE	**B**luetooth **L**ow **E**nergy
DK	**D**evelopment **K**it
EMG	**E**lectro**M**yo**G**raphy
GPS	**G**lobal **P**ositioning **S**ystem
IDE	**I**ntegrated **D**evelopment **E**nvironment
IMU	**I**nertial **M**easurement **U**nit
MDK	**M**icrocontroller **D**evelopment **K**it
MEMS	**M**icro-**E**lectro**M**echanical **S**ystem
PAN	**P**ersonal **A**rea **N**etwork
PCB	**P**rinted **C**ircuit **B**oard
SDK	**S**oftware **D**evelopment **K**it
SIG	**S**pecial **I**nterest **G**roup
SoC	**S**ystem on **C**hip
SQL	**S**tructured **Q**uery **L**anguage

To my beautiful Tori
who,
had she been given the chance,
would be here beside me.

Chapter 1

Introduction

1.1 Background to the Study

The Sports Industry was estimated to be worth 1.3 Trillion US Dollars in 2017[1]. Sports analysts play a crucial role in the success and development of the athletes and clubs that make up this vast industry[2]. Technology has contributed substantially to the potential of sports analysis by means such as allowing coaches to provide better and more comprehensive feedback to their athletes; assistance to referees and umpires in making in-match decisions; data collection and match statistics; assistance in athlete injury prevention and technique improvement and by providing a better viewing experience to spectators[3]. With the increasing pressure on athletes performing at high levels, the efficient use of technology for them and their coaches becomes imperative. Beyond the sector of professional competition, the improvements in price and efficiency of these technologies has made it accessible to a typical athlete looking to improve their performance or technique.

Optical motion capture and magnetic tracking systems have been the methods of obtaining human movement analysis data over the past years[4]. However, this requires expensive equipment and precise set up and calibration of the performance environment[4]. Athletes often require training data that cannot be generated in a laboratory environment and non-professional athletes require more convenient and affordable methods of analysis.

The advancements in micro-electromechanical (MEMS) technologies has provided an alternative for individuals seeking methods of motion analysis. These include accelerometer, gyroscope, magnetometer and goniometer sensors. MEMS sensors are light, small, low cost and are low power-consuming, meaning they are an efficient solution for on-body human motion or sports analysis outside of a laboratory environment[4].

The use of MEMS sensors in sports analysis has seen considerable growth over the last few years. This study will contribute to the research and development of sports analysis using this technology.

1.2 Research Focus

The focus of this study is on investigating the most efficient and cost-effective methods of implementing sports motion-tracking devices, with the use of a linked mobile application to provide real-time performance analysis, such that can be accessible for

non-professional and professional athletes alike. Although consumer-available devices and applications are found in the market, there are several drawbacks to the existing technology. This technology is relatively new and there are many reports of the devices still being in development phases, with bugs and inaccuracies occasionally appearing.

The sensor types most used in sports tracking devices are: heart-rate monitors, IMUs, GPS and EMG. The device to be developed in this project is aimed at tracking motion to provide feedback on technique and performance, thus heart-rate monitors and EMG will not be included in the design.

As the devices are intended to be used with the corresponding smartphone application, the smartphone's internal GPS can be used, removing the need for the device to contain a GPS module.

1.3 Objectives

The objective of this research project is to build an 'ActionTracer' motion-tracking device with a connected smartphone application that may be extended to, or adapted for, physical performance-analysis of many different sporting disciplines. Figure 1.1 shows conceptual use-cases of this device and application.

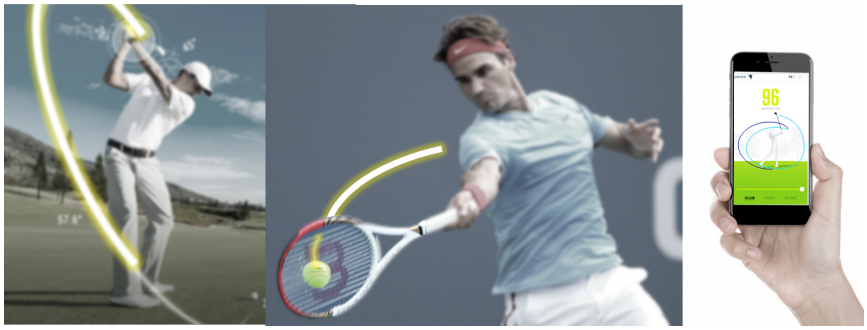


FIGURE 1.1: Conceptual Images of Swing Tracking. (Images adapted from [5]–[7])

Problems to be Investigated

Satisfaction of the project's objectives require the investigation into the following subtopics:

- The best-performing, low power and cost-effective components required to build such a device
- The optimal method of communication and transfer of data between the Motion-Tracking device and smartphone
- The development of a smartphone Application Programming Interface (API) for receiving and manipulating motion data into useful statistics and performance feedback

1.4 System Requirements

The system requirements, adapted from the topic outline and research conducted, are detailed in the following User, Functional and Technical & Design requirements tables.

User Requirements

User Requirement ID	**Requirement**
Motion-Tracking Device	
RD.U.01	The device should be small enough to fit, unnoticeable, onto a piece of sports equipment (e.g. a golf club).
RD.U.02	The device should be self-powered and last long enough to capture data from a full training session, sport irrespective.
RD.U.03	The device should be affordable for athletes and non-professionals.
Smartphone Application	
RA.U.01	Motion statistics and performance details should be displayed graphically in real-time.
RA.U.02	Past event data should be easily accessible for the user.
RA.U.03	The application should not consume too much storage space on the smartphone, nor have large power requirements.
RA.U.04	The application should be user-friendly and easy to navigate.

Functional Requirements

Functional Requirement ID	**Based on Requirement:**	**Requirement**
Motion-Tracking Device		
RD.F.01	RD.U.02	The device should have a battery lasting at least four hours on full use.
RD.F.02	RA.U.01	The device should record and transmit real-time motion data.
Smartphone Application		
RA.F.01	RA.U.01	The application should receive a wireless motion data stream.
RA.F.02	RA.U.02	The application should have a page displaying activity logs.
RA.F.03	RA.U.04	There should be a logical flow between pages and a good visual layout.

Technical and Design Requirements

Technical Requirement ID	Based on Requirement:	Requirement
Motion-Tracking Device		
RD.T.01	RD.U.01	The size and weight constraints for the device are: 5cm x 5cm x 2cm and less than 10g.
RD.T.02	RD.F.01	Maximum current draw of 0.5mA, for approx. 3V supply
RD.T.03	RD.F.02	An Inertial Measurement Unit (IMU) is required to measure the device's motion and a microprocessor is required to extract useful motion data.
RD.T.04	RD.F.02	A wireless communication module is required to send the motion data and communicate with the smartphone application.
RD.T.05	RD.U.03	The cost of components used in the device should not exceed R1000.
Smartphone Application		
RA.T.01	RA.F.02	The application requires internal or cloud-based data storage.

1.5 Scope and Limitations

Financial Limitations

The budget for the project is R1200, supplied by the Electrical Engineering department.

Hardware Development Limitations

The development of the hardware for the Motion-Tracking Device is limited to:
–The use of surface mount components with pad sizes no smaller than the 0805 standard
–The use of surface mount components that do not have their pads on the underside, as the university does not have access to the soldering equipment required for this assembly
–The use of parts that are locally and commercially available

Software Development Limitations

Any software used in the project should be either open-source or relatively cheap, so as to fit within the project budget.

Time Limitations

12 weeks was the given time frame for the completion of this project.

1.6 Plan of Development

Figure 1.2 below shows a Gantt Chart detailing the budgeted time allocations for, and flow between, project subsections. The yellow diamonds show the draft and final report hand-in deadlines.

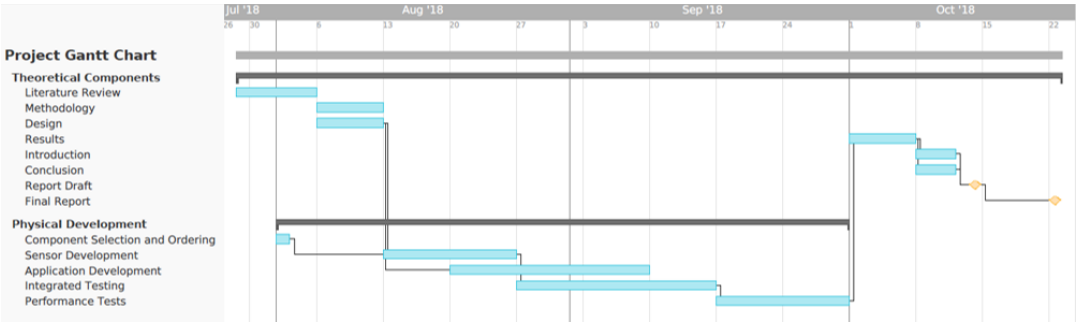


FIGURE 1.2: Gantt Chart for the Full Project Development

Figure 1.3 shows the development of the project from the initial stage of research through to the obtaining of results and writing up the report.

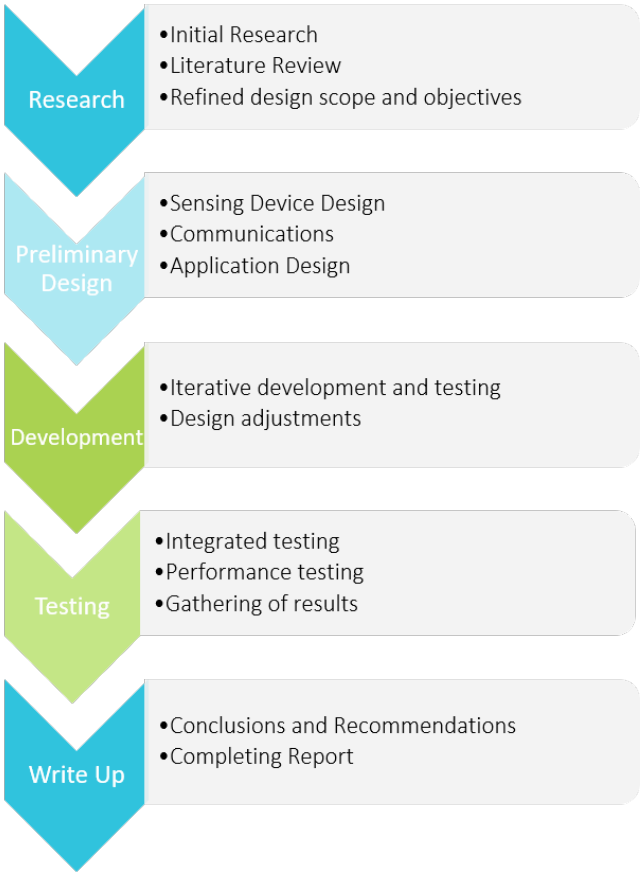


FIGURE 1.3: Plan of Project Development

Chapter 2

Literature Review

The following subsections detail the literature and current practices relating to the development of a wireless, wearable motion-tracking sensor network. Figure 2.1 shows the flow and relationships between these subsections.

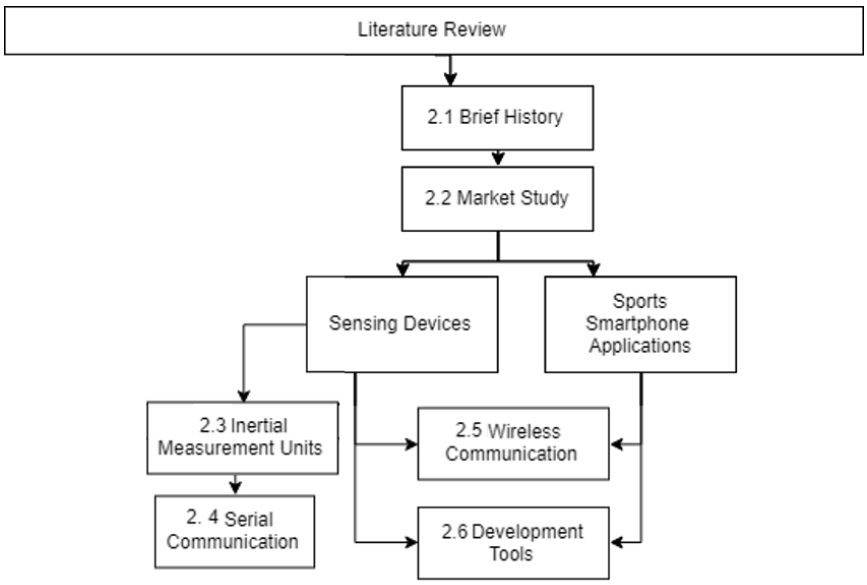


FIGURE 2.1: Graphical Overview of the Literature Review

2.1 Brief History of Performance-Analysing Technology

Methods of measuring athlete technique and performance date back to the early 1900s. The 1900 Paris Olympic Games saw the use of 'chronophotography'[8], which was used to show a succession of frames of an athlete's movement. Other methods used at these games include rowing biomechanics recordings, shown in Figure 2.2b, and the use of a 'sphygmograph': a portable device that records features of one's pulse and blood pressure graphically. A drawing of this device is seen in Figure 2.2a. These scientific investigations were conducted to help coaches and performers better understand the winning techniques and strategies.

Although Inertial Measurement Units (IMU) were introduced in the 1930s, they were originally only used in aircraft navigation systems and large devices due to their

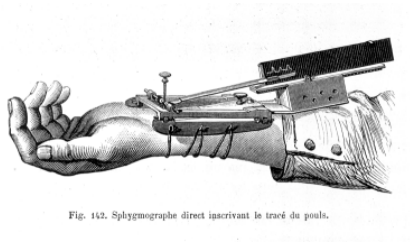
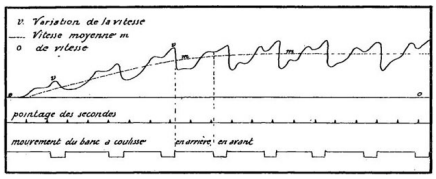


Fig. 142. Sphygmographe direct inscrivant le tracé du pouls.

(A) Drawing of a Sphygmograph. (Image taken from [8])



(B) Rowing Stroke Analysis, 1900 Olympic Games (Image taken from: [9])

bulky size, cost and power consumption[10]. They were not used in sports sensor devices until the recent introduction of micro-electromechanical system (MEMS) IMUs, which are compact, utilize low processing power and are low cost[10].
Since then, the sports analysis industry has evolved rapidly to the present technology, utilizing: microsensors, Global Positioning Systems (GPS), heart rate monitors, electromyography (EMG) and other use-specific techniques.

2.2 Market Study

A review of devices and applications leading the industry of sports-performance-analysing sensing is provided below.

Motion-Tracking Devices used in Sports Analysis

There are several advanced sports sensing devices currently available on the market. These range from the most consumer-friendly smart watches, such as offered by Garmin, FitBit and Apple, to sensor networks available for developers, and equipment with built-in sensors, such as Babolat's tennis racquets and Intel's sensing cricket bats for professional use. Several of the top, portable sensing devices available to consumers are considered. Companies that produce devices, still intended for a typical athlete or a range of purposes, more targeted at developers and research are also considered.

Zepp

'Zepp' is a company that provides multi-sport sensors and a training platform for athletes and coaches[11]. They offer sensors suited for several 'swing' sports: golf, baseball-softball and tennis as well as slight variations designed for soccer which tracks the players and provides game statistics such as kicks, sprints, distance and maximum speed. Figure 2.3 shows the various mounting options for the different sporting applications.

Device Specifications
Their 'swing' sensors measure approximately 25.4mm x 25.4mm x 12.3mm and weigh 6.25g. They use Bluetooth Low Energy, built-in rechargeable Lithium Ion Batteries (advertised to last 8 hours), Dual Accelerometers and Dual 3-Axis Gyroscopes. They are advertised at $149.99[11].

(A) Baseball Bat Mount (B) Tennis Racquet Mount (C) Golf Glove Mount (D) Soccer Mount

FIGURE 2.3: Zepp Sensor Mounts. (Images taken from [11]

Trace

'Trace' offers devices for Action Sports and Soccer, similar to Zepp [12]. Their Action Sports, however, deal with water and snow sports. *Device Specifications*
These devices include GPS tracking, 9 Sensors, are said to last 7 hours on a single battery charge, are 100% waterproof and can be synced to a smartphone at the end of the activity to avoid the battery drain caused by continuous connection and syncing.
Device Specifications
They offer their devices for $199 [12].

Notch

A market-leading company offering sensor networks ready for athletes or developers who will create use-specific software for the device, is 'Notch'. They describe their product to be "an affordable, mobile-friendly movement analysis and motion capture system for any activities from healthcare to sports to entertainment"[13]. Figure 2.4 below shows: a pack of Notch sensors available for purchase, example body-positioning of these sensors and a screenshot of the 3D reconstruction on their mobile application.

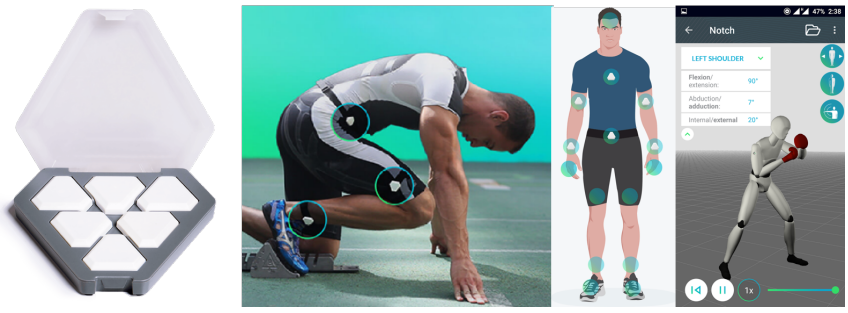


FIGURE 2.4: Images of the 'Notch Interfaces' Products. (Images taken from [13])

Device Specifications
Table 2.1 below outlines the core features and capacities of the Notch sensors.

Price Range
Their sensor packs are advertised at a starting price of $379 for a pack of 6 devices

TABLE 2.1: Notch Sensors' Core Features

Feature	**Description**
IMU Sensors	9 Axis: Accelerometer, Gyroscope and Magnetometer
Network Capability	1-16 Modules (suggested that with some minor adjustments more devices can be connected)
Battery Life	Expected to last long enough to record continuously long enough to fill the onboard memory (Approx. 6 hours, configuration-dependent)
Onboard Storage	FLASH memory with the capacity of 2+ Hours at 40Hz
Wireless Communication	Bluetooth Low Energy. A maximum of 6 notches can stream data in real-time simultaneously.
Sampling Frequencies	5Hz, 10Hz, 20Hz, 40Hz, 50Hz, 100Hz, 125Hz, 200Hz, 333Hz, 500Hz
Additional Features	Fully Waterproof. Comprehensive developer support, including: Android and iOS SDKs, a Cloud API, a manual, installation guides, online support and video tutorials.

which can record motion for one or two limbs (including a charging dock, USB cable, 12 straps, carrying pouch and a Developer License)[13].

Mbient

Mbient offers similar products to those of Notch, with more of a focus on developers and research[14]. They offer a selection of kits available for purchase: the Developers Kit, the Research Kit or the Monitoring Kit. Each kit contains a sensing board and full access to their MetaBase application. One can alternatively purchase the sensing board alone. The Research kit additionally includes a device housing and a choice of wearable mount - i.e. sleeve, wristband, adhesive or clip on mount and the Monitoring Kits are tailored for their use-specific case. There is full, open-source documentation of their sensors' hardware and development APIs. Figure 2.5 below shows the design and features of the 9-axis, Round model Developer Kit.

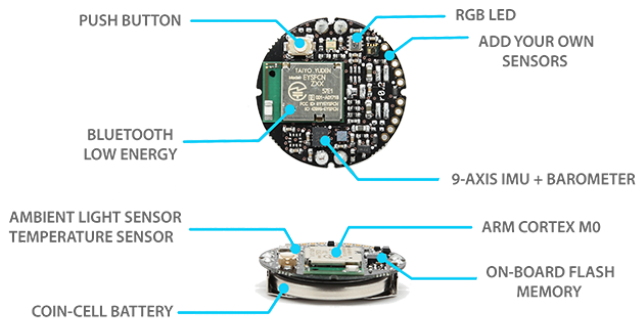


FIGURE 2.5: Mbient Development Kit Sensor (Image taken from [14])

Device Specifications
Their selection of sensing boards include feature variations of:
- 6-axis IMU with Temperature or 9-axis IMU with Temperature, Light and Barometer Sensors

- 24 x 17 x 4 mm Rectangular or 24 diameter x 4 mm Round
- 8MB or no Onboard Memory
- USB Rechargeable 100mAH li-ion battery or CR2032 200mAH Coin-cell battery
- Battery life of 2 days to 2 weeks
- Weight of 5.7g

Price Range
The sensing boards with the lower range features from the above list sell for $53.99 on the Mbient website while the boards with improved features sell for $75.99. The most basic Research Kit, the adhesive mount package, sells for $89.99 – $94.99 (depending on the sensing board selected)[14].

XSens

XSens offer 'MVN Analyze': a "software package optimized for use in research, sports science, ergonomics and rehabilitation"[1] and 'MTW-Awinda' which are "highly accurate, small and lightweight 3D human wireless motion tracker[s] for real-time applications[2]." The Awinda radio protocol provides 10 μ s, or better, time-synchronization between each MTw on a wireless body area network. XSens is a market leader in this 3D motion-analysis technology and their solutions produce results validated even in heavily magnetically-distorted environments.

Additional Device Manufacturers

Other consumer-available wearable sensor devices include StatSports' 'Apex Athlete Series' wearable tracking device and vest[3], which sells for $299.99, and TuringSense's Pivot devices which are still in the development phase, but are aimed at tennis and yoga[4].

2.2.1 Sports Smartphone Applications

All of the aforementioned companies offer a device-compatible mobile application. These applications all include similar features: device connectivity, activity tracking and processing, activity history and are usually free for download on the Google Play Store[5]. Some include GPS real-time location tracking and community networks, where users can communicate with and share their progress with their friends and co-athletes. Several applications are built based off of other companies' sensors, such as HEAD's 'HEAD Tennis Sensor' application which uses Zepp-powered sensors. Figure 2.6 shows several promotional application screenshots from HEAD's tennis application. These application pages are typical of most of the applications available for download.

2.3 Inertial Measurement Units

As mentioned in section 2.1, it wasn't until the introduction of micro-electromechanical system (MEMS) IMUs that IMUs were small enough or efficient enough to be used

[1] https://www.xsens.com/products/xsens-mvn-analyze/
[2] https://www.xsens.com/products/mtw-awinda/
[3] https://statsports.com/
[4] https://www.turingsense.com
[5] https://play.google.com/store/apps

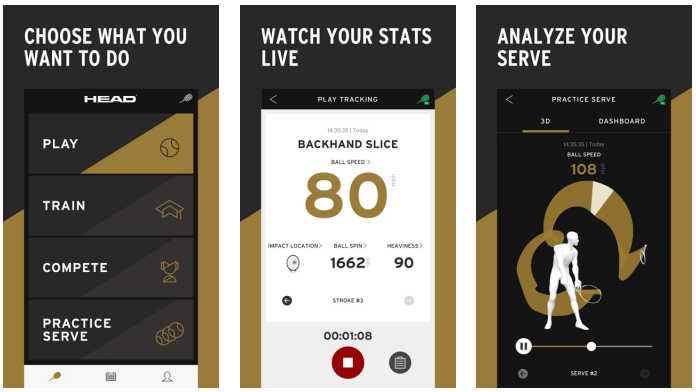


FIGURE 2.6: 'HEAD Tennis Sensor' Application Promotional Screenshots. (Images taken from [15])

in wearable or equipment-mountable sensing devices. IMUs are primarily used in devices to measure velocity, orientation and gravitational force. The manufacturers competing for the best IMU designs are currently said to be: Invensense, Honeywell, STMicroelectronics, Microstrain, and X-Sens[10]. IMUs can contain an accelerometer and a gyroscope, or a combination of accelerometer, gyroscope and magnetometer. These three sensors and their contributory effects are outlined below.

Sensor Types

Accelerometers

Accelerometers are used to measure acceleration forces: static, such as the continuous force of gravity, and dynamic forces such as movement or vibrations[16]. Accelerometers are accurate in the long term as there is no drift and the Earth's centre of gravity does not change. However, accelerometer readings are generally noisy and can be unreliable in short motion runs [17].

There exist two types of accelerometer devices: digital and analogue. Digital accelerometers output readings over serial communication protocols such as I2C, SPI or USART while analogue devices output voltage levels which are converted using Analog to Digital Converters (ADCs).

MEMS Accelerometers
MEMS contain miniaturized mechanical and electro-mechanical elements that convert some measured mechanical signal into an electrical signal[18]. MEMS Accelerometers are sensors that measure acceleration in units of mV/g. The technology can be modelled as a mass, spring and damper system, as seen in figure 2.7.

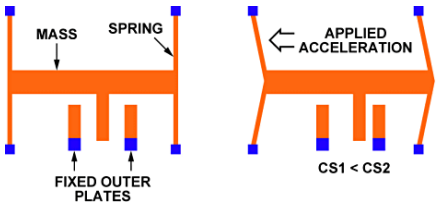


FIGURE 2.7: Micro Structure of a MEMS Accelerometer

The figure shows how, when acceleration is applied in a particular direction, the mass moves, resulting in a change of capacitance between the plates. This change of capacitance is measured and will correspond to a particular value of acceleration.
For measuring acceleration in more than one direction, rows of these structures are placed at 90 degree angles.

Gyroscopes

Brief History of Gyroscopes
Gyroscopes are used to measure orientation and angular velocity. The first uses of gyroscopes are said to date back to the 1700s where spinning devices were used for sea navigation in foggy conditions[19]. Years later, around 1916, gyroscopes were adopted into aircraft systems, where they are still commonly used today. Optical gyroscopes using lasers were introduced in the 1960s. This saw the introduction of gyroscopes in aeronautical and military applications. Technical advances lead to MEMS gyroscopes, which have been used for the past 10-15 years and are mass-produced and implemented in many successful products[19].
MEMS Gyroscopes
Smaller and more sensitive devices were created with MEMS vibrating mass technology. The two primary types of MEMS gyroscopes are the tuning fork model and the vibrating ring model.
Coriolis force is used to measure angular rate. Figure 2.8 shows the effects of the rotation of the device.

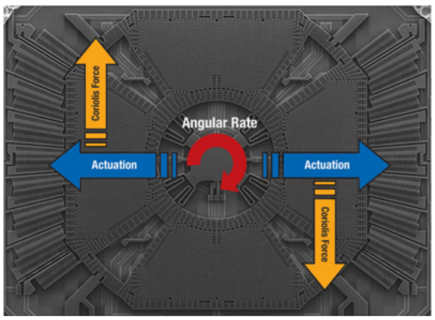


FIGURE 2.8: Illustration of the Workings of a MEMS Gyroscope

The angular rotation causes a perpendicular displacement of the mass. This causes a change in capacitance which is measured, processed and corresponds to a particular angular rate.

Magnetometers

Magnetometers are useful in allowing one to calculate yaw, in combination with accelerometer readings. They measure the earth's magnetic field by using the Hall Effect or the Magneto Resistive Effect. However, the location of the device is required, even when calibrated, in order to calculate the correct values due to the effects of the Earth's magnetic field.

Sensor Mathematics

Typically, a combination of MEMS sensors is used to increase the accuracy and reliability of sensing devices. A model for each sensor type is given, before detailing the methods of measurement combinations.

Accelerometer Model

Measurements of linear acceleration can be described by the equation (taken from [17]):

$$\tilde{a} = a^{(g)} + a^{(l)} + \eta, \qquad \eta \sim N(0, \sigma_{acc}^2) \tag{2.1}$$

Where $a^{(g)} + \eta$ expresses noisy, upward gravity vector readings ($1g = 9.81m/s^2$) and $a^{(l)}$ is the combined value of the gravity vector and external forces when undergoing motion[17].

Gyroscope Model

Equation 2.2 below represents true angular velocity as a combination of: b, the bias, and η, the additive, zero-mean Guassian noise of readings taken. The bias value is temperature dependent and can change over time. It can, however, be approximated as a constant.

$$\tilde{\omega} = \omega + b + \eta, \qquad \eta \sim (0, \sigma_{gyro}^2) \tag{2.2}$$

Taylor expansion can be used to express these gyroscopic measurements as orientation:

$$\theta(t + \triangle t) \approx \theta(t) + \frac{d}{dx}\theta(t) \triangle t + \epsilon, \qquad \epsilon \sim O(\triangle t^2) \tag{2.3}$$

Where:

- $\theta(t + \triangle t)$ is the angle at the current time step - to be solved
- $\theta(t)$ is the angle at the previous time step - already measured
- the time step: $\triangle t$ is known
- ϵ is the approximation error

Sensor Fusion

As gyroscopic measurements produce drift, and acceleration readings include noise, the two sensing types can be combined to eliminate these adverse effects. For sensors with 3 degrees of freedom each, a 6-degrees-of-freedom (6 DOF) sensor can be formed.

IMU with Two Sensors

The drift from the gyroscope can be removed using a high-pass filter, and the noise from the accelerometer can be removed using a low-pass filter. Several filters are commonly used to implement the fusion of these sensor measurements, such as the complementary filter or the Kalman filter. The following equation can be used to combine accelerometer and gyroscope readings and result in angle readings with no drift nor noise:

$$\theta^{(t)} = \alpha(\theta^{(t-1)} + \tilde{\omega} \triangle t) + (1 - \alpha) atan2(\tilde{a_x}, \tilde{a_y}) \tag{2.4}$$

IMU with Three Sensors

3-Axis Accelerometer readings can be used to calculate the pitch and roll of a body. Pitch and roll angles are known as 'tilt'. A magnetometer, however, is required

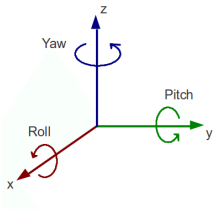


FIGURE 2.9: Diagram of Pitch, Roll and Yaw

should calculations of yaw be required. Figure 2.9 shows a diagram of pitch, roll and yaw for reference.

Several filters such as the Kalman, Madgewick/Mahony or Complementary filters can be used to transform motion readings from these devices into meaningful and accurate motion information.

2.4 Serial Communication

Digital sensing devices require serial communication protocols to communicate with a microprocessor. Most IMU devices use synchronous methods: Serial Peripheral Interface (SPI) or the Inter-Integrated Circuit (I2C).
Synchronous data transfer differs from asynchronous data transfer through the use of a common clock. Asynchronous communication requires that the two devices agree on the transmission time per bit of information.

Serial Peripheral Interface Bus

SPI requires a 'master' and 'slave' device [20]. There are four lines necessary for communication between these devices: SCLK - the clock signal; MOSI - master out slave in; MISO - master in slave out and $\bar{SS}$ - slave select or chip select. The master sets the clock speed and can communicate with more than one slave, in which case there is an additional $\bar{SS}$ required for each additional slave[20].

Inter-Integrated Circuit Bus

I2C also requires a master and slave devices and can have many slaves. Unlike SPI, I2C requires two communication wires: SCL - serial clock and SDA - serial data line[20]. I2C can host data transfers at speeds ranging from 10kbits/s to 400kbits per second.
Each I2C device has a 7-bit address, where typically some of these bits are set by the hardware.

2.5 Wireless Communication Protocols

Wireless communication protocols refer to methods of transferring data or power between at least two points, without the use of any electrical conductor connections[21]. This has allowed for long-range communication that would be impractical or impossible with wire-based methods.

Brief History of Wireless Communication

The term 'wireless', when describing communication protocols, was first used around 1890. This, however, was used to describe the first radio receiving and transmitting technology: 'wireless telegraphy', which saw the transfer of encoded characters of text, usually in Morse code. In around 1920 this term was replaced by the word 'radio' and only in the 1980s-1990s was the term 'wireless' again used to describe technology that can communication without wires, such as: two-way radios, wireless networking, cellular telephones, the Global Positioning System (GPS) and so on[21].

There are several methods of achieving wireless communication. These include the use of: electromagnetic wireless technologies, ie. radio (the most common), electric fields, light or magnetic or the use of sound. Radio wave communication allows for ranges varying from a few meters, eg Bluetooth, to the extent of millions of kilometers, such as in deep-space radio communication[21].

Data Communications

There are several wireless technologies created for the purpose of data transfer, such as Wi-Fi, cellular data services, low-power wide-area networks (LPWAN) and wireless sensor networks. These each target different application types, based on range, transfer speed, power requirements and other features[22]. Figure 2.10 shows a diagram of the broad context of network types and how each relate.

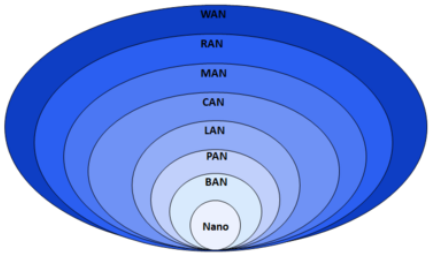


FIGURE 2.10: Computer Network Types by Spatial Scope[5].

The network types considered for use in this project are those on the IEEE 802.15: Wireless Personal Area Networks (WPAN) standard. The IEEE Wireless Standards are networking stands covering the physical layer specifications of technologies from wireless to Ethernet[23]. Networks within this standard are: Bluetooth, UWB, ZigBee and Mesh Networks. Figure 2.11 shows block diagrams of Bluetooth and Bluetooth Low Energy device implementation architectures. This shows the layering of the protocol stack, where it is evident the physical layer (PHY) is at the lowest level. 'Protocols' are methods and rules that define and govern the synchronization of communication between two entities. They can be implemented in the software of hardware or a combination of these. A 'protocol stack' is the software implementation of several layered protocols. Lower levels deal with low-level hardware communication and the higher levels add more features and capabilities[23].

[5]Source: https://en.wikipedia.org/wiki/File:Data_Networks_classification_by_spatial_scope.png

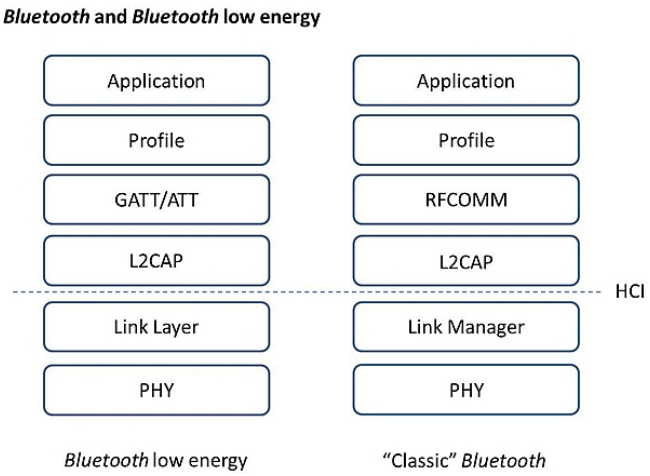


FIGURE 2.11: Bluetooth Protocol Stacks

Wireless Personal Area Networks

Zigbee, on 802.15.4, is used for short range wireless sensor networks. It is a Local Area Network (LAN) and mesh network protocol, suited to the medium-range transfer of small amounts of data. Ultra-Wideband (UWB), 802.15.3a, is a short range and high-bandwidth link. It has been used more recently in precision locating, tracking and sensor data collection applications. Bluetooth Low Energy (BLE) is a personal area network (PAN), meaning it has a fairly short range. Despite its shorter range than that of ZigBee, it has a far greater data rate and a much lower amount of transmission power[24].

2.6 Development Tools

A range of development and design tools were investigated for use within this project.

ARM Cortex-M Development Tools

SEGGER Embedded Studio

"SEGGER Embedded Studio is a powerful C/C++ Integrated Development Environment (IDE) for microcontrollers[6]." It is free for non-commercial and non-profit educational use and is available on Windows, macOS and Linux. It comes with two toolchains: GCC and LLVM. The IAR or ARM/KEIL compilers and other external toolchains can alternatively be used. Embedded Studio also includes J-Link/J-Trace debug probes which allows for the use of advanced J-Link technology.

Keil Microcontroller Development Kit

Keil offers Microcontroller Development Kits (MDKs) in three different packages: MDK-Lite, MDK-Essential and MDK-Plus. MDK-Lite is the only free version and has a code size restriction of 32 Kbytes. These software development solutions are for Arm-based microcontrollers and include "all components that you need to create,

[6]https://www.segger.com/products/development-tools/embedded-studio/

build, and debug embedded applications[7]."

The 'μVision IDE' is included in the MDK-Core and is very similar to that of SEGGER's Embedded Studio. The MDK includes "two Arm C/C++ Compilers with assembler, linker, and highly optimize run-time libraries that are tailored for optimum code size and performance."

Arduino

"Arduino is an open-source electronics platform based on easy-to-use hardware and software[8]."

The Arduino programming language is a strong abstraction, 'high-level' language, based off of 'Wiring': an open-source programming framework for microcontrollers, making it easy to use for new developers. Their IDE is also aimed at being easy-to-use for beginners, but allows flexibility in using a variety of boards and microcontroller platforms.

Android Application Development

Android Studio

Android Studio provides a development and testing environment for building applications for every type of Android device. The software includes useful tools, such as device emulation; CPU, Memory and Network profiling and benchmarking.

Cross-Platform Development

Designing applications using cross-platform development tools saves the time and effort of application developers. There exist several cross platform development platforms, such as Flutter, React Native and Xamarin.

PCB Design Software

Altium

Altium is a market-leading printed circuit board (PCB) design software. It is not free for use, however UCT has several licenses available for students' use[9].

Autodesk EAGLE

Autodesk Eagle is a slightly less professional PCB design software than Altium, however it is open-source and therefore used by many companies such as SparkFun[10].

[7] http://www2.keil.com/mdk5
[8] https://www.arduino.cc/en/Guide/Introduction
[9] https://www.altium.com/
[10] https://www.autodesk.com/products/eagle/free-download

Chapter 3

Methodology

This chapter details the processes followed to achieve the project objectives. Four main phases were created: the Initial Research and Project Planning Phase; the Research Phase; the Design, Development, Testing and Validation Phase and the Conclusions and Recommendations Phase. Figure 3.1 shows a flow chart of the full project execution.

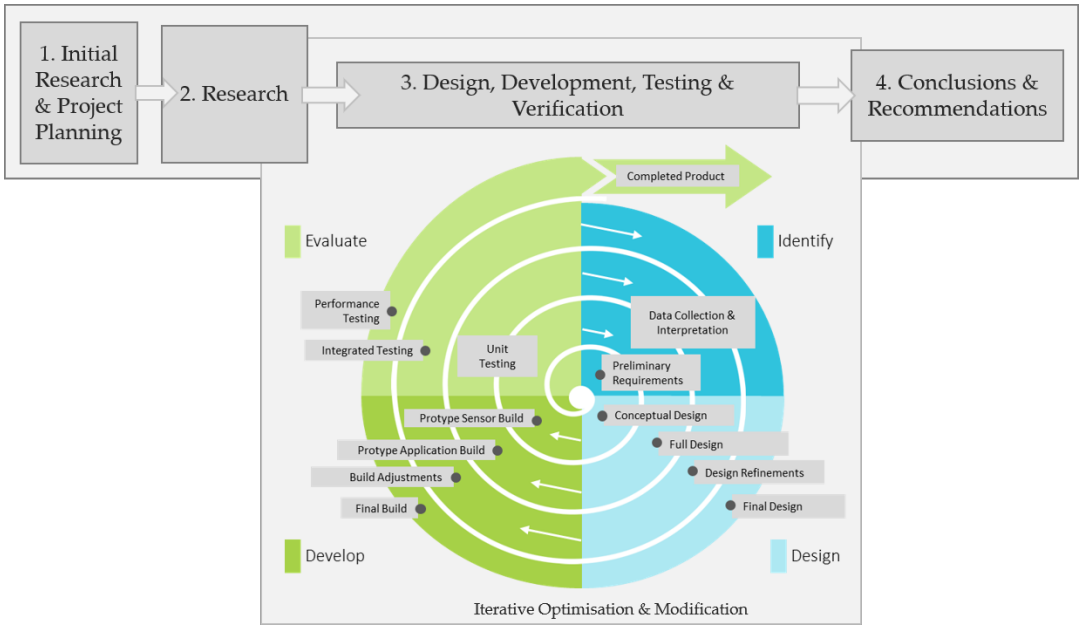


FIGURE 3.1: Project Development Diagram

3.1 Phase 1: Initial Research and Project Planning

Initial research was conducted to refine the project specifications and to assess and compare the product's purpose against other methods and devices already in the field. Additionally, a full plan for the project's execution was developed.

Mendeley Desktop was used for the management of all the papers acquired in the Research Phase and all the project files were continuously synced to a Google drive to remove any risks of data loss.

3.2 Phase 2: Research

Journal articles, websites, lecture notes and tutorials were consulted to develop a full literature review. Existing market technology, optimal methods of communication for low-power, wearable sensor devices and techniques of motion-tracking using IMU sensors were researched.

It is noted that the component selection and ordering process had to occur early in the timeline, as seen in figure 1.2, to reduce the risk of long delays reducing the amount of development time. Thus, this section of the research was conducted first.

3.3 Phase 3: Design, Development, Testing and Validation

The sections of Design, Development, Testing and Validation were grouped together in one phase as these processes did not occur linearly, but in a V-model, where verification and validation of each, increasingly complex, step of development was conducted. The development section of the phase was in the form of an iterative spiral model. Figure 3.1 shows an outline of the spiral model of development in the overall pipeline of the project.

Design

A conceptual design of the full system was developed. The design was broken into two main sections: the Motion-Tracking Sensor Modules and the Mobile Application. The design of the Sensor Modules includes the physical module design and the software. The Application design includes: the development language, desired features and logic and the Application aesthetics.

Development

The development of the Sensor Modules and Mobile Application happened concurrently as the assessing of certain functionality checkpoints for each module depended on one another.
Due to the V-model approach, continuous testing throughout the development phase was conducted and compared against the requirements set out in section 1.4. The tests conducted increased from Unit tests, to Integrated tests and finally Performance and Evaluation tests were completed.

Unit Testing

Each feature was tested as it was added to the prototype.

Tests and development checkpoints for the Sensor Device were:

1. Establish a wireless connection with a mobile phone
2. Successfully connect the IMU sensor and achieve motion readings
3. Verify the accuracy of the IMU readings

Tests and development checkpoints for the Mobile Application were:

1. Reliable navigation between pages
2. Implementation of wireless connectivity
3. Ensure application is compatible with a range of Android devices

Integrated Testing

The integrated development tests include:

1. Establish connection between the Device and the Application
2. Successfully receive data from the Device and verify its accuracy
3. Transform received data into meaningful information

Performance Testing

The performance of each module and the system of whole was tested by the following categories:

- Power consumption of each module, while transmitting and receiving data over the wireless connection
- Range of device connection
- Memory and CPU use of the modules

And, finally, the functional and technical requirements were verified and the user requirements were validated, confirming the satisfaction of the project objectives.

3.4 Phase 4: Conclusions and Recommendations

Conclusions and recommendations were drawn based on the results obtained, the comparison of the results against the project objectives and other devices in the field. Areas of potential performance enhancement and further inclusions were identified and commented on.

Chapter 4

ActionTracer Device Design

4.1 Conceptual Design

The motion-tracking, 'ActionTracer' Device will comprise of a four main components, as shown in Figure 4.1 below.

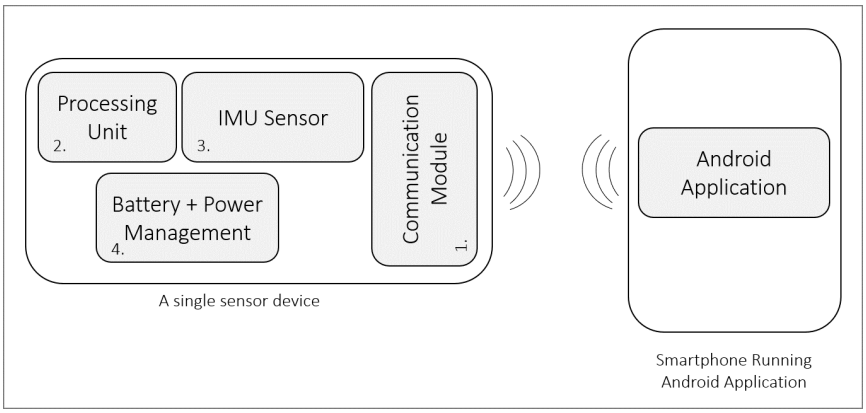


FIGURE 4.1: Conceptual Design of Sensor to Application System

The design for each of these modules is detailed in the following subsections.

4.2 Wireless Communication Protocol

According to Technical Requirement RD.T.04, listed in section 1.4.4, the device requires a wireless communication module to continuously transmit motion readings to a smartphone application.
Of the listed networks optimised for IoT applications, the following three were considered for use in the design: ZigBee, UWB and Bluetooth Low Energy (BLE). A comparison of these protocols is presented in Table 4.1 below.

Bluetooth Low Energy was selected for the device due to its greater level of support for the interface with Android devices, its sufficient range, and faster bit rate than the other compared networks.

4.3 Bluetooth Low Energy Modules

As the Bluetooth SIG (Special Interest Group) has a strict and comprehensive Bluetooth Qualification Process for the application of the Bluetooth License Agreements on products[1], only certified modules containing built in antennas were considered.

[1] https://www.bluetooth.com/develop-with-bluetooth/qualification-listing

TABLE 4.1: A Comparison of Wireless Communication Protocols for Wearable Sensors

	ZigBee	**UWB**	**BLE**
Frequency Band	868/915 MHz, 2.4 GHz	3.1-10.6 GHz	2.4GHz
Maximum Outdoor Range (m)	10-100	10	50
Network Size	8	64 000+	Undefined
Relative Development Support	Fair	Minimal	Excellent
Maximum Raw Bit Rate (Mbps)	0.25	110	1

To satisfy the size constraints set out in requirement RD.T.01, the package size of the BLE module should be as small as possible. The final factor on which the module decision was made is that of development support.

Based on these factors, three main chips and their corresponding third-party modules were considered: Nordic's nRF52832 and Espressif's ESP32.

Nordic's nRF52832

Nordic is a world-leading company in cost-sensitive, wireless solutions. Their chips are largely found in the PC peripherals, sports and fitness sensors, toys, gaming controllers and more[25].

Chip Features

The Nordic nRF52832 is a System on Chip (SoC) that supports multiple wireless protocols, including: Bluetooth Low Energy, ANT and 2.4GHz proprietary. Some of the core features of the chip are shown in a design comparison table in Section 4.3.3.

Third-Party Modules

Manufacturers producing nRF52832 modules, available at local electronics suppliers, are: Rigado, Laird, Taiyo Yuden, Fanstel, Garmin Canada Inc and several other less well-known suppliers. The price range for these modules begins at around R91 for Rigado's BMD-300 and extend to around R250 for Garmin Canada's modules. The modules are tiny in form factor and most measure approximately 1cm x 2cm. Due to the limitation of having no access to advanced soldering methods, only modules with accessible solder pads could be considered for use.

Development Options

The top two development boards for the development of nRF52832 chips are the 'SparkFun nRF52832 Breakout Board' and Nordic's NRF52-DK (PCA10040) (Development Kit).

The SparkFun board sells for R294,86 at Digikey (before shipping) and includes: a 32.768kHz RTC crystal, a user-programmable button and LED, a trace antenna and a pre-programmed bootloader to allow for serial programming[26].

The Development Kit is listed at R576,42 on Digikey (before shipping), however there are several available for use in UCT store. These kits includes the benefits of Segger J-Link Programming/Debugging and supported Nordic Software Development Tool-chain, using Keil, IAR and GCC.

Espressif's ESP32

Chip Features

The Espressif ESP32 offers very similar features to that of the Nordic chip, however it includes Wi-Fi support[2].
The 'ESP32-WROOM-32' sells for R100,50 and measures 25.5 mm x 18.00 mm

Development Options

There are several breakout boards available for the ESP32, from manufacturers including: Digilent at R384,13; Espressif's Evaluation Board at R134,50 and SparkFun's 'ESP32 Thing' at R286,09.

Chip Comparison

Feature	Nordic nRF52832	Espressif ESP32
Processor	32-bit ARM Cortex-M4F Processor	Two low-power Xtensa® 32-bit LX6 Microprocessors
Power Requirements	1.7V - 3.6V operation	2.7V - 3.6 V
Onboard Memory	512kB flash + 64kB RAM	520kB internal SRAM, 4 MB SPI Flash
Network Support	Supports concurrent BLE/ANT protocol operation	Supports BLE (Bluetooth v4.2) and Wi-Fi
Relative Documentation and Support	Excellent	Good

4.3.1 Module Selection

Despite the nRF52832 having slightly lower specifications that the ESP32, it was chosen due to the vast and comprehensive development support and the lower power requirements and energy saving capabilities.

[2] 520kB internal SRAM

4.4 Microcontroller

As a SoC was chosen, no additional microcontroller was required for the design. The SoC includes the ARM Cortex-M4F microprocessor.

4.5 Inertial Measurement Unit

As discussed in Section 1.2, only an Inertial Measurement Unit will be included in the design. The selection of the number of degrees of freedom and module to be included in the design are discussed below.

Degrees of Freedom Selection

Although not all sporting applications would require the device's full number of degrees of freedom, the use of a 9-axis IMU was chosen as it provides the most flexibility for device implementation with an minor increase in price.

Component Options

The IMU sensor was chosen based on: best cost to performance ratio, and the availability at local suppliers. Additionally, due to the soldering limitations, the chip would have to have a package option that allows for soldering access or a breakout board for development should be available.

As all the IMU options at local suppliers are only available in QFN packages, and therefore cannot be used, only breakout boards were searched for and compared. The modules available include[3]: the SparkFun IMU Breakout - MPU-9250; Adafruit's 'Breakout Board 9-DOF IMU BNO055'; Parallax Inc's 'LSM9DS1 9-Axis IMU Module' and the '9-DOF MPU-9250 Module 9 axis' from BotShop[4]. These modules include the 9-DOF IMUs: Bosch BNO055 System in Package(SiP)[5], InvenSense MPU-9250 SiP[6] and STMicroelectronics LSM9DS1[7]. The SparkFun IMU Breakout provides the exact same features as the BotShop breakout, yet the BotShop module is cheaper, so the SparkFun module was discarded from the comparison. A table comparing these modules and the IMU chips they're based on is shown in 4.2 below.

IMU Selection

InvenSense's MPU-9250 was selected as shows the best cost to performance ratio of the IMU selection. While Bosch's BNO055 offers top-end motion processing algorithms and fusion software, the inclusion of a microcontroller is unnecessary in this design as the BLE SoC already contains a microprocessor. The unit price of the STMicroelectronics is slightly less than that of InvenSense's MPU-9250, however the support available online is not as comprehensive nor does it include useful built in DMP and sensor firmware.

[3] https://www.digikey.co.za/
[4] https://www.botshop.co.za/product/9-dof-mpu-9250-module-9-axis/
[5] https://www.bosch-sensortec.com/bst/products/all_products/bno055
[6] https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/
[7] https://www.st.com/en/mems-and-sensors/lsm9ds1.html

TABLE 4.2: IMU Package Comparison

	MPU-9250	**BNO055**	**LSM9DS1**
Breakout Board Module	9-DOF MPU-9250 Module	Adafruit 'Breakout Board 9-DOF IMU BNO055'	Parallax Inc's 'LSM9DS1 9-Axis IMU Module'
Breakout Board Price	R174,56	R516,56	R387,83
Unit Price	R157,11	R178,39	R93,56
Size	3 x 3 x 1 mm	5.2 x 3.8 x 1.1 mm	3.5 x 3 x 1 mm
Sensing Scales -Acceleration	+-2g/+-4g/+-8g/+-16g	+-2g/+-4g/+-8g/+-16g	+-2g/+-4g/+-8g/+-16g
-Gyroscope	+-250/+-500/+-1000/+-2000 dps	+-125/+-250/+-500/+-1000/+-2000 dps	+-245/+-500/+-2000 dps
-Magnetometer	Full scale range of +-48 gauss	Typical: +-13 gauss (x-, y-axis) and +-25 gauss (z-axis)	+-4/+-8/+-12/+-16 gauss
Power Management	2.4 - 3.6 V supply.	1.7 - 3.6 V supply. Power saving modes. Typical low power current of 2.7mA.	1.9 - 3.6 V supply required. Eco power mode down to 1.9mA and Intelligent power saving options
Support	Software drivers fully compliant with Google's Android 4.1 Jelly Bean release. Excellent documentation and online repositories.	Built in top-standard software, well-documented	Well-documented
Additional Features	Includes a built in Digital Motion Processor (DMP) and MotionFusion and run-time calibration firmware	Built in 32-bit cortex M0+ microcontroller running Bosch Sensortex sensor fusion software.	ECOPACK, RoHS and "Green" compliant

Development Environment

The device will be developed using the NRF52 DK with SEGGER Embedded Studio (SES), as it is supported by Nordic and does not have the code size limitation of Keil. As mentioned, the NRF52 DK is freely available in the UCT component store and

offers a more flexible platform for software development.

The wiring shown in development set up, Figure 4.2, will be discussed in the following section.

The programming connections between the NRF52 DK and the SparkFun nRF52832 Breakout Board do not include the powering of the SparkFun board, an external source is required. The SparkFun 'Beefy 3 - FTDI Basic Breakout' was purchased as it simultaneously powers the board and can be used for serial programming.

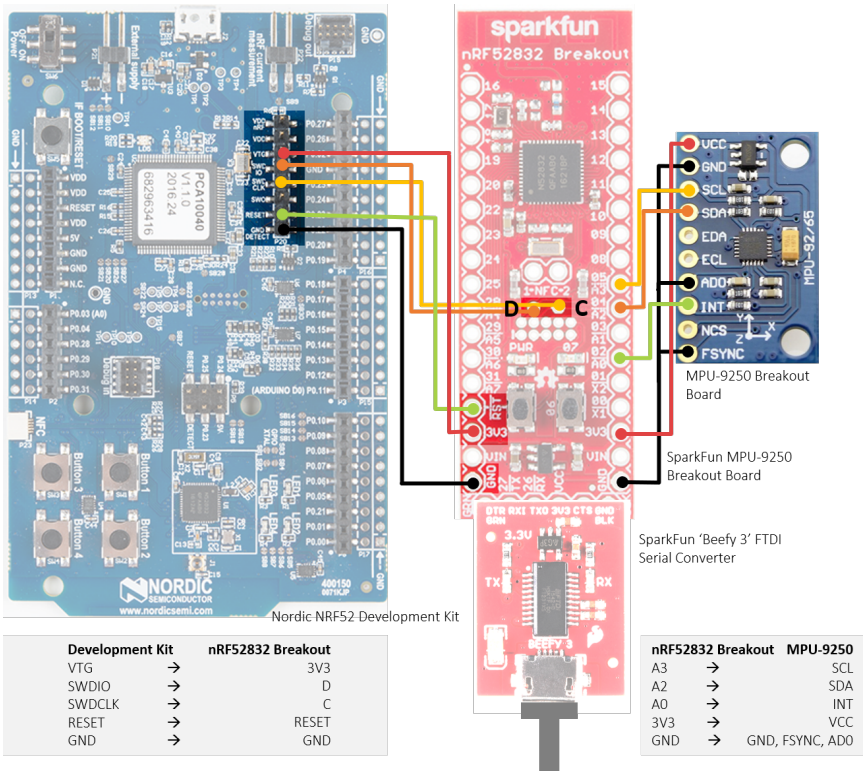


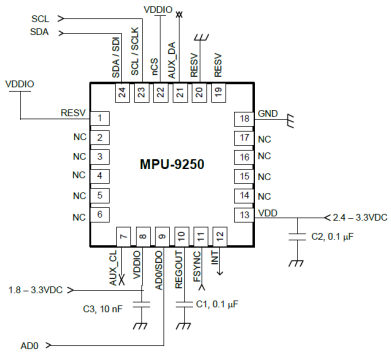
FIGURE 4.2: Development Set Up (Images taken from [27])

4.6 Electrical Design

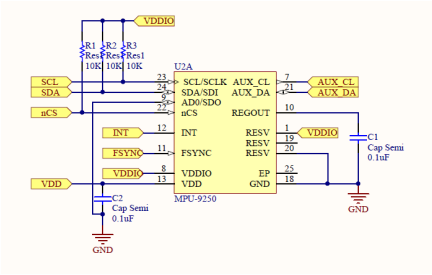
Module Interface

Figures 4.3a and 4.3b, below, show the pin-out and schematic diagrams for the MPU-9250 to operate with I^2C communication. I^2C was chosen as the communication protocol between the IMU and BLE SoC due to the large amount of support available and the efficiency of the transfer protocol.

The five pins required for this implementation were connected to the nRF52832 as shown in Figure 4.2.



(A) MPU-9250 Schematic for I2C Use



(B) Schematic of the MPU-9250 Breakout Board

Power Management

Module Power Requirements

The power requirements of the system components for likely scenarios, taken from their respective datasheets, are summarised in Table 4.3 below. It is noted that the IMU's voltage would be tied to that of the nRF52832 (to ensure IO compatibility) so these power requirements would be slightly higher than shown in the table.

TABLE 4.3: Power Requirements of the Components Included in the Motion-Tracking Device

	Current	Nominal Voltage	Total Power
MPU-9250: 9-axis	3.7 mA	2.5 V	9.25mW
MPU-9250: 6-axis (acc + gyro, no DMP)	3.4 mA	2.5 V	8.5mW
MPU-9250: Idle Mode	8 uA	2.5 V	20mW
nRF52832: Radio RX @ 1Mb/s BLE mode, Clock = HFXO	6.5mA	3 V	19.5uW
nRF52832: $I_{ON\_RAMOFF\_EVENT}$ - System ON, No RAM retention, Wake on any event	1.2uA	3 V	3.6uW

The SparkFun breakout board has a built in 3V3 voltage regulator with a maximum input of 6V, shown in Figure 4.4. However, the chip's 3.3V input can take between 1.7V to 3.6V meaning a coin cell battery or a pair of alkaline batteries can be used directly to supply the chip.

4.7 Software Design

Software Functionality

The Device software has two main functions:
–1. Bluetooth communication, and
–2. IMU reading and processing

Some additional software tools and features are described below. The development of the software is detailed in the Device Implementation section.

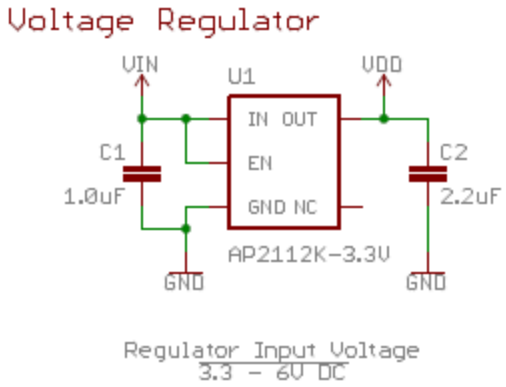


FIGURE 4.4: Voltage Regulation in the SparkFun nRF52832 Breakout Board

Nordic Software Development Kit

The nRF5 SDK includes a vast "selection of drivers, libraries, examples for peripherals, SoftDevices, and proprietary radio protocols"[25]. The SDK is ready to use with SES and requires no installation. v15.2.0 is the latest SDK at the time of this project and supports the SoftDevices: S112 v6.1.0, S132 v6.1.0, S140 v6.1.0 and S212 v5.0.0..

SoftDevices

SoftDevices are wireless protocol stacks that complement Nordic's nRF5 Series SoCs.They are provided by Nordic and are qualified, precompiled binary files. Applications can be built without the use of a SoftDevice, but all the Bluetooth Low Energy and ANT^{TM}examples in the nRF5 SDK require the use of a SoftDevice. There are different SoftDevices available for the various nRF5 chips and protocols. Figure 4.5 below shows two diagrams of the architecture of the nRF52 Series chips with the inclusion of a SoftDevice supporting Bluetooth, the leftmost being a higher-abstraction overview and the diagram on the right showing a more in-depth model.

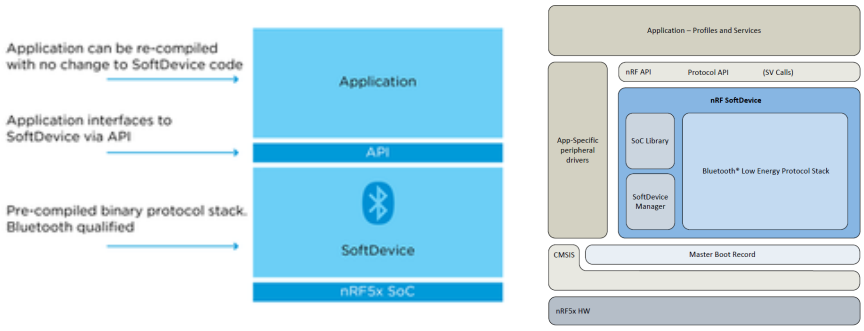


FIGURE 4.5: Diagrams of the nRF52 Series Software Architecture. (Images taken from [25])

The S132 SoftDevice was chosen for this design as it supports Central and Peripheral Bluetooth Low Energy on the nRF52832.

Device Firmware Updates

A very useful feature to be included in the software design is Nordic's Over the Air (OTA) Device Firmware Updates (DFUs). If the DFU bootloader is programmed onto the nRF chip, the firmware can be updated using one of Nordic's applications, desktop or mobile, or through the Development Kit, using nrfutil. Figure 4.6 shows the steps followed to implement a firmware update using the DFU mode[25].

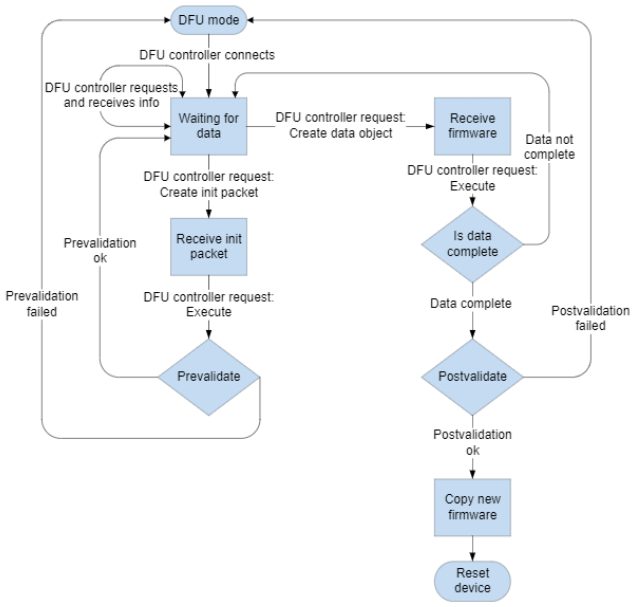


FIGURE 4.6: Process Flow on the DFU Target. (Image taken from [25])

The DFU initiates the transfer of a firmware image. The image is received and validated by the DFU bootloader before the device is reset and the image copied, replacing the previous firmware. It can be used to update the application, SoftDevice or existing bootloader.

4.8 Physical Device Design

Two designs were created for the Motion-Tracking Device. They were created using Altium Designer. The first is a model that includes the optimal components, disregarding the soldering limitations. Images of this optimal design are shown in Figure 4.7. The collision warning is there due to the software detecting the through-hole oscillator colliding with the coin cell battery holder on the back side of the board. This would not pose an issue as the wires would be clipped and the battery holder is non-conductive. The full schematic for this design is included in Appendix A. A Laird BL654 Module was chosen as it was the best documented, cost-efficient option. The design can be further optimised by removing the push button and LED but, as this is a prototype design, they were included for debugging purposes. The measurements of this design are 25.4mm x 33.6mm. This easily satisfies Requirement RD.T.01 of having an area of less than 5cm x 5cm, as it is assumed an outer housing would not contribute largely to the area measurement.

Another design was created for prototyping use within the university constraints. This model includes headers to connect an external IMU module, MicroSD memory

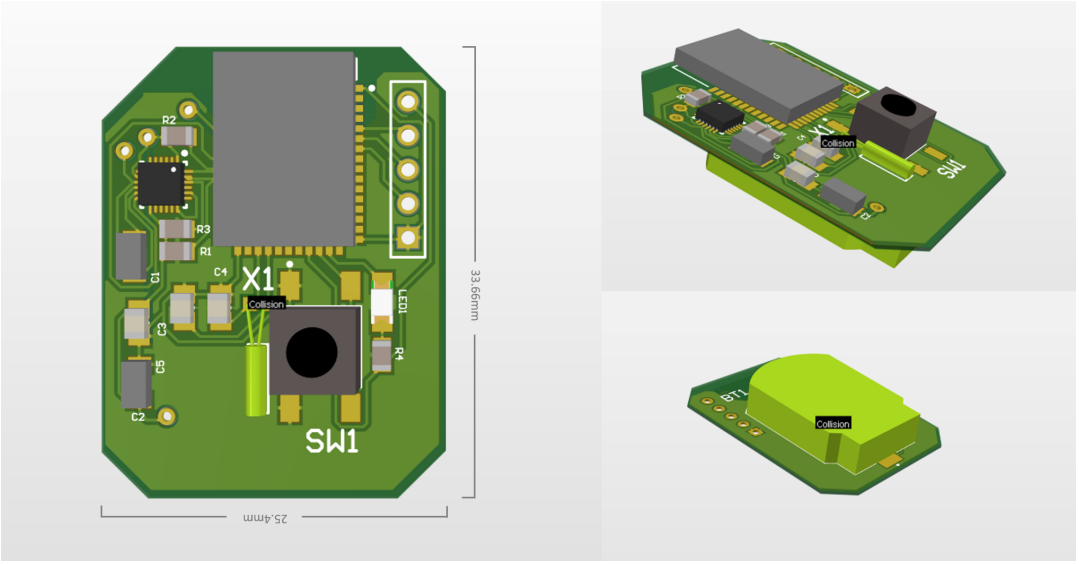


FIGURE 4.7: 3D PCB Model for Full Device Design

card and for programming the device. It also includes a power slide switch. The full schematic can also be found in Appendix A.

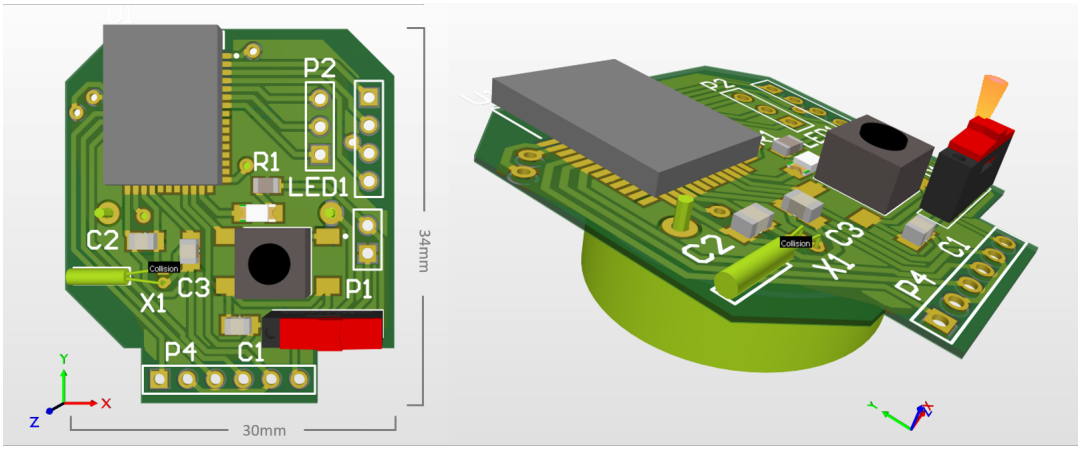


FIGURE 4.8: 3D PCB Model for Limited Device Design

Chapter 5

ActionTracer Application Design

5.1 Conceptual Design

Following the Requirements RA.F.01, RA.F.02, RA.F.03 and RA.T.01; the pages required in the Application Design are shown in Figure 5.1. The design of each of the components in this outline are detailed below.

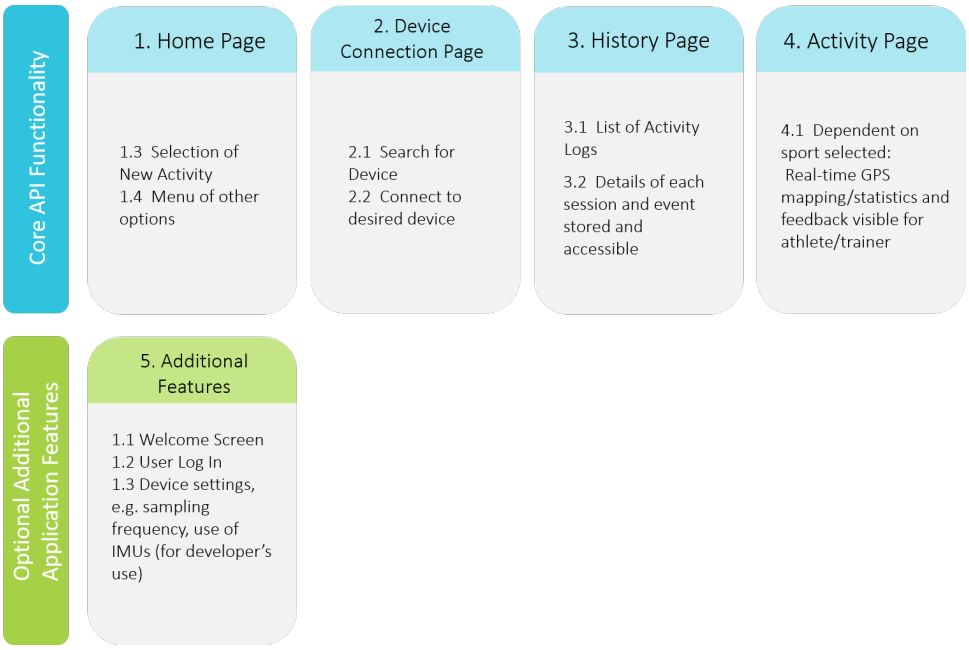


FIGURE 5.1: Overview of Required Pages for the Mobile Application

5.2 Development Environment

The application was developed in Android Studio v3.1.4 due to its vast support base and well-documented learning resources. Although AndroidStudio provides a built-in device emulator, this does not support Bluetooth Connectivity so the application was tested using a Samsung Galaxy S7.

The Flutter SDK was chosen to develop the smartphone application as its applications can be compiled for both Android and iOS and has well supported learning material and sufficient libraries for the required functionality.
An Android API Minimum Level of 19 was chosen as this covers 95.3% of Android devices. The full list of API Level options is shown in Figure 5.2.

The project file was synced to a Github repository to ensure frequent back ups and reduce the risk of data loss.

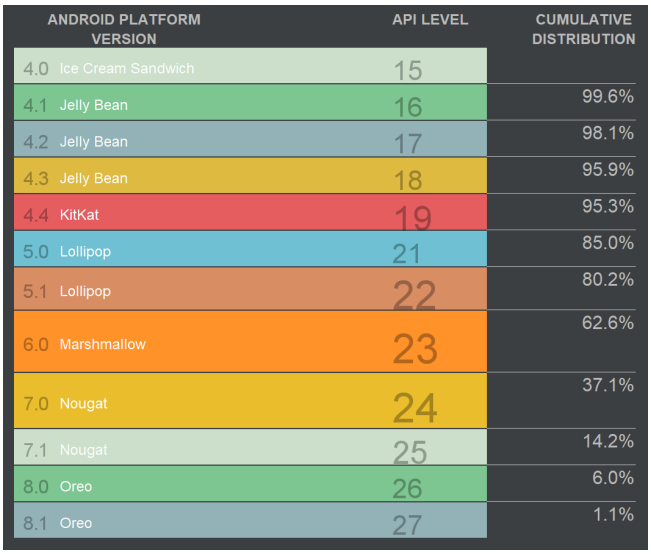


FIGURE 5.2: Android API Levels and Cumulative Percentage of Compatible Devices. (Image taken from AndroidStudio

5.3 Connectivity Library

The Flutter Bluetooth Low Energy (BLE) Library was used for the implementation of the BLE protocol. Figure 5.3 shows the process of implementing BLE functionality using this library.

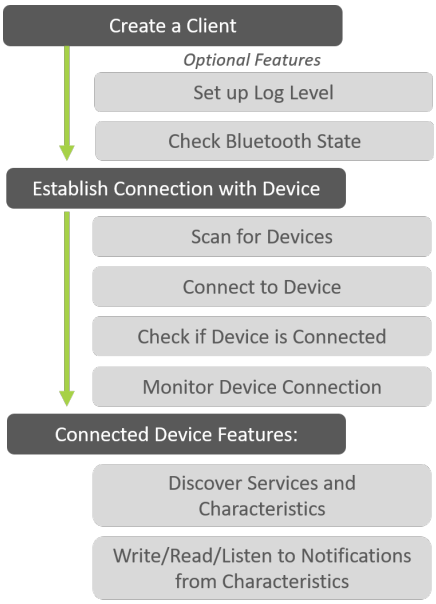


FIGURE 5.3: Implementation Flow Diagram for the Flutter BLE Library[28]

5.4 Application Framework

Several components are required for the application's framework. The elements required to satisfy the core API functionality are: Page Navigation, Lists and Charts.

These features can be implemented using the internal functions of the Material library, and the Flutter Charts plugin.

Navigation

A diagram of the routing logic between pages for a simple implementation of the API is shown in Figure 5.4 below.

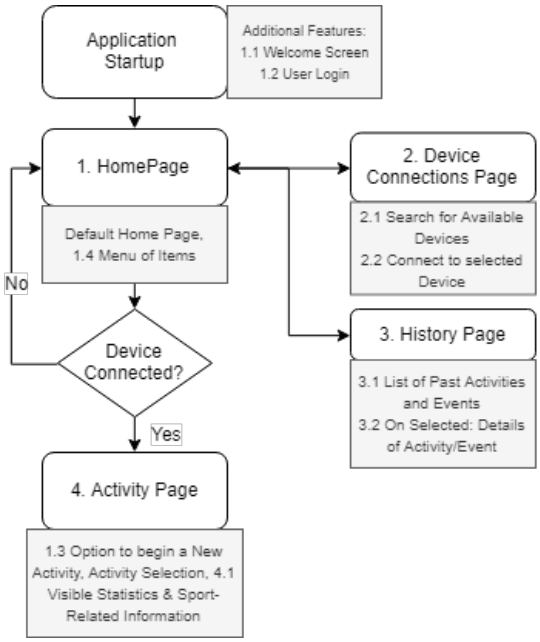


FIGURE 5.4

5.5 Data Storage

Offline Data Storage

The SQFLite Dart package can be used to implement database storage of the motion readings on the mobile phone being used. Figure 5.5 shows a conceptual design of the application page allowing the user to access their past activity records.
As a key requirement of the system is to be portable for use outside of a laboratory environment, it is likely there would be no wireless network for cloud data storage. Therefore, the data received from the Bluetooth stream should be stored locally and can optionally be backed up to a cloud storage facility on connection to Wi-Fi or similar.

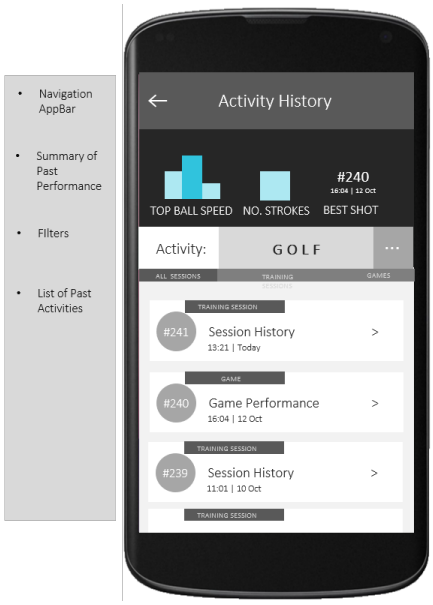


FIGURE 5.5: Application Page Design: Data Logs

Cloud Data Storage

Cloud-hosted data storage is useful to reduce the amount of memory space used on the smartphone. Google's Cloud Firestore has well-documented Flutter implementation guides and offers easy-to-use real-time database management.

Chapter 6

Sensing Device Implementation

6.1 Implementation Using the Nordic PCA10040 Development Board

6.1.1 Initial Set Up

The Development Kit 'Getting Started Guide' was followed to test the PCA10040 Board and to install all the necessary and useful tools for development through this board[29].

After installing SEGGER Embedded Studio and the nRF5 SDK, several other required tools were added. These include: the SEGGER J-Link Software and Documentation Pack, nRF5x Command Line Tools for Windows, nRFgo Studio, nRF Connect for Mobile, the Nordic nRF Toolbox app and nRF5x pynrfjprog. These tools are useful for debugging and testing of the device.

6.1.2 Programming a Custom Board through the DK

Several steps must be taken enable programming of a custom board through the PCA10040 Development Board. This is required to allow programming of the SparkFun Breakout Board through the PCA10040.

SparkFun provides a board definition script for the board. This file must be renamed 'custom_board.h' and placed in the *SDK_dir* > components > boards. It was noted that a few additional lines were required that were present in the SDK PCA10040 board definition file, so these were added into the SparkFun board definition file. In the boards.h file, located in the same folder, the use of a custom board must be defined by including the line: #define BOARD_CUSTOM.

Finally, in the Project Solution in SES, in configuration > c_preprocessor_definitions, BOARD_PCA10040 should be changed to BOARD_CUSTOM. Another point noted in the programming of the Breakout Board through the PCA10040, is that if the PCA10040 is powered down while connected to the Breakout Board, the reset wire is activated and so should be disconnected so as not to disconnect the Breakout Board.

6.1.3 Connectivity

Several of the BLE Peripheral example projects in: *SDK_dir*>examples>ble_peripheral were run and tested using the nRF Toolbox app. The successful output and screenshots from the application are shown in the Results section.

6.1.4 Addition of the IMU

Library files for the implementation of MPU-9250 features, using SPI or TWI were found, however these were based on the Keil Project Files. On migrating to Keil uVision and compiling the code it was found that the files were larger than the code size limitation. The Nordic Documentation was followed which details the importing of Keil projects into SES, yet this is only supported for earlier versions of the SES SDK. A full library migration would be required, so to save time a decision was made to begin the prototype development using Arduino. SparkFun hosts full Arduino development tools for the board and provides several example sketches. Although Arduino is not the most versatile nor powerful IDE, it provides a convenient and useful starting point for development. Later, more refined versions of the application could be developed using SES when a better working prototype and understanding of the system has been developed.

6.2 Serial Programming Using USB to Serial

6.2.1 Setting up the Development Environment

As programming the chip using J-TAG removes the factory bootloader that comes pre-loaded onto the SparkFun Breakout Board, this should be reloaded before the board can be programmed using serial-to-USB.

Re-flashing the Serial Bootloader

While the Board is still connected as in the Figure 4.2 development set up, the following steps are taken to re-flash the SparkFun serial bootloader. Firstly the bootloader file should be downloaded[1]. A J-Link connection is then established. This can be done through SES or through the J-Link software. The following commands are then run in the terminal[2] (nrfjprog has been installed as mentioned in the initial set up).

LISTING 6.1: Commandline Commands to Program the nRF52 Device using nrfjprog

```
mergehex -m s132_nrf52_2.0.0_softdevice.hex sfe_nrf52832_dfu.hex -o
    merged.hex // Merge hex files
nrfjprog -e // Erase nRF52
nrfjprog -r --program merged.hex // Program the breakout board and reset
```

Arduino IDE Set Up

Following the Hookup Guide provided by SparkFun[26], the Arduino development environment was set up as follows:

- Install support for the nRF52 Board in Arduino
  - In Arduino, navigate to: File>Preferences
  - Paste the link[3] into "Additional Board Manager URLs"

[1] https://github.com/sparkfun/nRF52832_ Breakout/tree/master/Firmware/bootloader-custom
[2] https://github.com/sparkfun/nRF52832_ Breakout/issues/6
[3] https://raw.githubusercontent.com/sparkfun/Arduino_Boards/nrf5/IDE_Board_Manager/package_sparkfun_index.json

  - Select 'OK'
  - Tools>Board>Boards Manager
  - Install the SparkFun nRF52 Boards package
- Set 'SparkFun nRF52832 Breakout' in Tools>Board
- To trigger the bootloader:
  - Hold down Button 6 and the Reset button
  - Release the Reset button, whilst still holding down Button 6
  - Wait for the blue LED (pin 7) to begin blinking
  - Release Button 6
- A sketch can now be uploaded onto the board

The provided demo sketches of basic BLE button state reading and LED state writing were loaded and tested with the Nordic nRF Connect mobile application.

6.3 Device Connectivity

The 'BLEPeripheral' library, created by Sandeep Mistry[4] makes adding Bluetooth LE connectivity to the Board highly user-friendly. As BLE requires SPI communication, both the <BLEPeripheral.h> and <SPI.h> libraries must be added to the sketch.

The initialisations for a Bluetooth peripheral device and a single service, named 'serviceName', and a single float characteristic are implemented as follows. A number of different characteristic types can be set up, ie. longs, doubles, unsigned integers and so on. The number passed into the BLEService() function is converted into a UUID for the BLE communication. The arguments 'BLERead' and 'BLENotify' give read and notify features to the characteristic being set up.

```
const char * localName = "Device Name";
BLEPeripheral blePeriph;
BLEService serviceName = BLEService("1234");
BLEFloatCharacteristic Name = BLEFloatCharacteristic("1234", BLERead |
    BLENotify);
```

To set up this device, service and characteristic, the following function should be called in the Arduino setup().

```
void setupBLE()
{
//Advertise name and service
blePeriph.setDeviceName(localName);
blePeriph.setLocalName(localName);

//Adding a Service
blePeriph.setAdvertisedServiceUuid(serviceName.uuid());
blePeriph.addAttribute(serviceName);
// Add characteristics
```

[4] https://github.com/sandeepmistry/arduino-BLEPeripheral

```
        blePeriph.addAttribute(Name);

        //initialise BLE
        blePeriph.begin();
        }
```

The characteristics can then be accessed in the script's main loop.

```
blePeriph.poll();
...
Name.setValue(value);
```

6.4 IMU Implementation

There are well-documented libraries for the MPU-6050 and the SparkFun MPU-9250 provided by Jeff Rowberg[5] and Kris Winer[6] These were slightly adapted to suit the MPU-9250 Breakout Board.

Processing was used to render real-time 3D animations of the device's motion. Additionally, a sample of measurements was taken for the Device while stationary to calculate the drift accumulated.

[5]https://github.com/jrowberg/i2cdevlib
[6]https://github.com/sparkfun/SparkFun_MPU-9250_Breakout_Arduino_Library.

Chapter 7

Application Implementation

7.1 Development Environment Set Up

To enable application testing on an Android phone, developer mode has to be activated. This was done by navigating to: Settings > About device > Software info. Here, 'Build number' was tapped seven times to enable developer mode. After this, the Flutter project could be easily compiled to the smartphone in 'Debug' mode.

7.2 Connectivity Implementation

A conflict was discovered between the library dependencies. The FlutterBle Library is no longer maintained as the developers have switched to developing the native libraries for iOS and Android, RxBluetoothKit and RxAndroidBle respectively. The FlutterBle Library depends on versions of Dart <2.0.0. This did not pose any issues until the introduction of the data storage and chart libraries, which require Dart >2.0.0. Therefore, a library migration was undertaken, following several Forum guidelines, a Dart Compatibility pull request by PabloPL on the FlutterBle Library Github page [28] and the Dart dart2_fix tool and guidelines[30]. The Protobuf library was affected by this migration and had to be adapted too. The changes made are detailed below.

7.2.1 FlutterBle Library Migration

Dependency overrides were added in the Pubspec.yaml file as follows:

LISTING 7.1: Dependency Overrides Required for the FlutterBle Library

```
dependency_overrides:
       flutter_ble_lib: "1.0.0"
       protobuf: "0.6.0"
```

The changes suggested on the Github page [28] were then added to the 'lib/source/lib_core.dart' file. The dart2_fix tool[30] was then run, using the following Flutter commandline commands:

LISTING 7.2: Flutter Commandline Commands to Run the 'dart2_fix' Tool

```
flutter packages pub global activate dart2_fix
flutter packages pub global run dart2_fix
flutter packages get pub global run dart2_fix --apply
```

Finally, the remaining errors were iterated through to replace the last few conversions not covered by the dart tool, such as: BASE64 -> base64, Endianness.LITTLE_ENDIAN -> Endian.little.

7.2.2 FlutterBle Library Implementation

A BLE Client is created as the user navigates to the 'bleDevicesScreen'. This screen was taken from the FlutterBLE Library example code and allows the user to search for and connect to available devices.

LISTING 7.3: Creating a Client and Navigating to the Device Scanning Page

```
FlutterBleLib.instance.createClient(null).then(
        (data) =>
        Navigator.of(context).pushNamed(
                ScreenNames.bleDevicesScreen)
),
```

The 'bleDevicesScreen' was adapted, however, to route the user to a 'dataPage' on connecting with the desired device. The information of the connectedDevice is passed to the 'dataPage'.

LISTING 7.4: Connect to Device Implementation

```
_onConnectButtonClick(ScanResult scanResults) {
        FlutterBleLib.instance.connectToDevice(scanResults.bleDevice.id,
            isAutoConnect: true)
        .then((connectedDevice) =>
                        Navigator.of(_mainBuildContext).push(new
                            MaterialPageRoute(
                                builder: (BuildContext buildContext) =>
                                new dataPage(connectedDevice))));
}
```

The 'dataPage' first discovers the services and characteristics of the device and then displays this information.

LISTING 7.5: Discovering the Device Services and Characteristics

```
FlutterBleLib.instance
.discoverAllServicesAndCharacteristicsForDevice(_connectedDevice.id)
.then((device) {
        setState(() {
                _serviceDiscoveringState = "DONE";
        });
FlutterBleLib.instance
        .servicesForDevice(_connectedDevice.id)
        .then((services) {
                FlutterBleLib.instance.characteristicsForService(_serviceId)
                .then((characteristics) {
                        FlutterBleLib.instance.monitorCharacteristicForDevice(
                        services[_serviceIndex].device.id,
                            services[_serviceIndex].uuid,
                            characteristics[_characteristicIndex].uuid, new
                            Uuid().v1())
```

```
                    .listen((value) =>
                        _action(value.characteristic.value));
            });
    });
});
```

7.2.3 Data Receiving

The data sent from the sensing device is encoded into Base64 before being transmitted. This needs to be decoded on the receiving end before the data can be used usefully in the application. Dart's ByteData class can be used "to pack and unpack data from external sources (such as networks or files systems), and to process large quantities of numerical data more efficiently than would be possible with ordinary List implementations"[31].

The following snippet takes in the string value received, 'value', as shown in Listing 7.5, and decodes it from Base64. The output is a list of hexadecimal values, which are then converted into Int8 bytes and finally back into its original form of 32-bit Float: 'receivedValue'. The 'Endian.little' argument was passed into the 'getFloat32()' function as the default for the function is big-endian while the data sent is little-endian.

LISTING 7.6: Data Conversion Using the ByteData Class

```
import 'dart:typed_data';
...
_action(String value) {
        final List<int> newVal = base64.decode(value);
        setState(() {
                ByteBuffer buffer = new
                    Int8List.fromList(newVal).buffer;
                ByteData byteData = new ByteData.view(buffer);
                receivedValue = byteData.getFloat32(0,Endian.little);
        });
}
```

7.2.4 Smartphone Permissions

For the Ble features to work on the Android device, Location permission had to be given to the application being debugged. This was done by first adding the following lines of code into the project's 'AndroidManifest.xml' file:

```
<uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission
    android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

and then navigating to: Settings > Applications > Application Manager > [Project_name] > Permissions on the smartphone and activating 'Location'.

7.3 Application Framework Building

Example code files from Udemy Flutter courses [32], [33] were adapted to suit the functionality requirements of the application.

The Flutter Charts Library was utilized to represent some of the received data in a graphical form.

Unit tests were continually performed as each element was added to the framework.

Chapter 8

Results

The results of the tests corresponding to those laid out in Section 3.3, are given below.

8.1 Unit Tests

Sensing Device Unit Tests

Bluetooth Connectivity

Bluetooth connectivity was successfully established between the nRF52832 breakout board and nRFConnect running on an Android device. The example projects for BLE peripherals, such as the demo 'Heart Rate Monitor' (HRS) and 'Beats Per Minute' (BPM), provided in the SDK were successfully loaded and connected to the nRF Toolbox application. Figure 8.1 shows screenshots of the nRF Toolbox application and data being received in the HRM example.

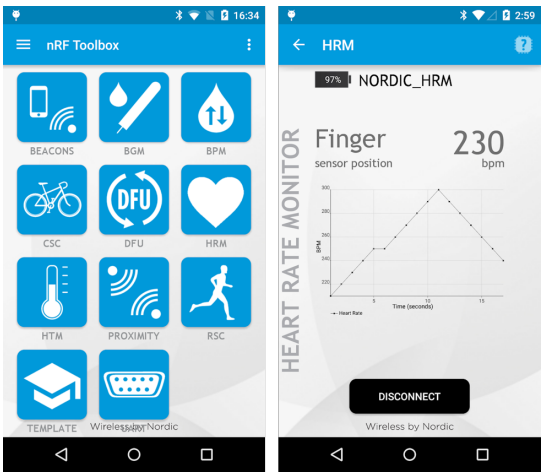


FIGURE 8.1

Figure B.1 in Appendix C shows further successful Bluetooth connectivity testing results, implemented using Arduino, such as the Sparkfun LED Control and Button State examples.

IMU Measurements

The MPU-9250 was successfully connected to the nRF board and readings were displayed in the Arduino COM terminal. Several screenshots from the Processing animation tests are shown in Figure 8.2. The drift results are shown in Table 8.1.

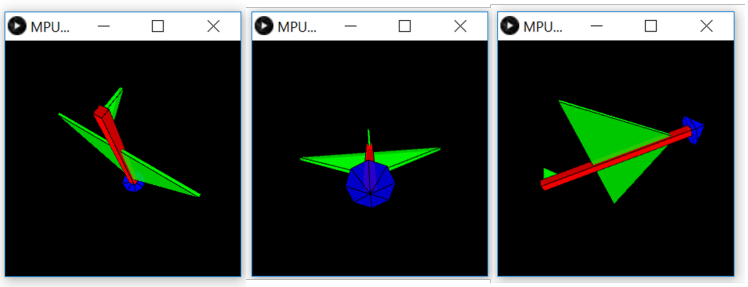


FIGURE 8.2: Processing Screenshots: 3D Animation of IMU Movement

TABLE 8.1: Calculate Yaw, Pitch and Roll Drift for a Sample of 40 IMU Readings

	Yaw	**Pitch**	**Roll**
Drift (deg/s)	0,022	0,014	0,017

Mobile Application Unit Tests

Application Framework

The tutorials' demonstration applications, which each included a particular feature such as the use of drawer menus, using lists, creating widgets and so on, were successfully created and navigation back and forth through the application's pages was reliably implemented.

BLE Connectivity

The Flutter BLE Library was successfully ported to work with Dart 2 and the included example project that was run worked as expected. Further stages of connectivity testing require the involvement of the Sensing Device and are therefore detailed in the Integrated Testing section.

Android Device Compatibility

The API chosen for the application development, discussed in Section 5.2, ensures compatibility with 95.3% of Android devices. Emulation of the project using the built-in AndroidStudio device emulator could not be used to test several device cases as it does not support projects requiring wireless connectivity.

8.2 Integrated Testing

Connectivity

Connectivity between the Device and Mobile Application was successfully achieved. Figure 8.3 shows several screenshots of the Application framework. The first screen shows the Drawer Menu, the second screen shows the list of devices found in the Bluetooth scan and the third screen shows the successful connection to the Sensing Device with several graphs displaying the values received.

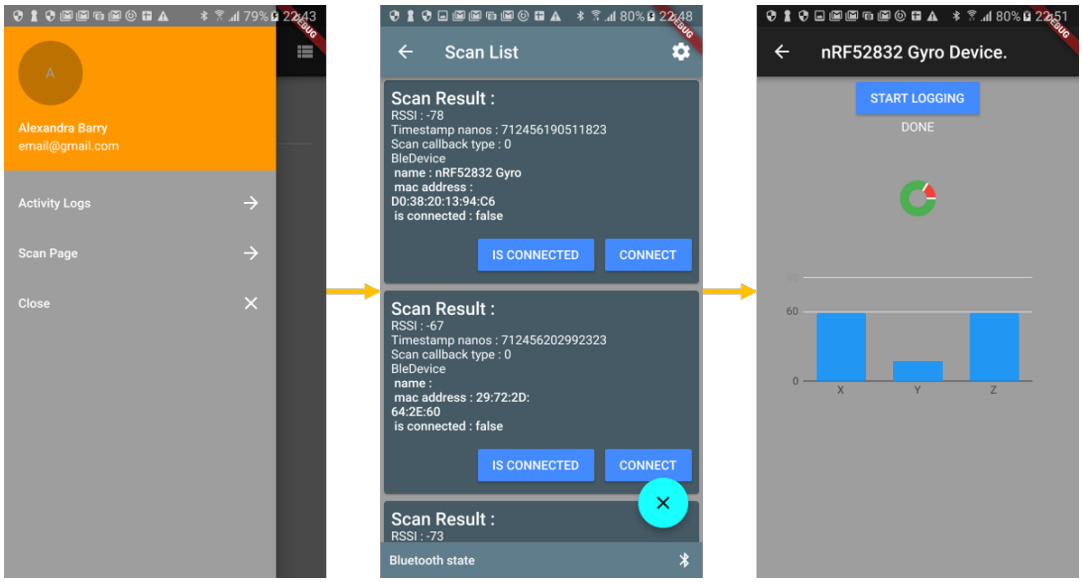


FIGURE 8.3: Application Screenshots

Data Transfer

As mentioned in the above subsection, Figure 8.3 shows a screenshot of data being received from the Sensing Device and transformed into graphs. The values of the data shown in the graphs were printed on the screen for debugging purposes and successfully verified to be the same as those being transmitted from the Sensing Device, seen in the Arduino COM terminal. Within the time frame of the project, however, receiving data from only two characteristics at a time was achieved.

8.3 Performance Testing and Satisfaction of Requirements

Power Consumption

The results of CPU profiling using AndroidStudio's built in profiling tool showed approximately 16% CPU use for the Application while receiving a stream of data from the Sensing Device. A screenshot of this can be seen in Figure B.2 in Appendix C.

The measured current being drawn from the Sensing Device while configured in 9-axis mode and transmitting a stream of data, over a single BLE Characteristic, was 1μA.

Memory Usage

The size of the Application on the Samsung Galaxy S7 smartphone it was tested on, is 87,34 MB. However, the memory used would be dependent on the number and size of the data logs, which was not achieved within the time limitation of the project. With successful implementation of Cloud data storage, this memory-use could be minimised for optimal functionality.

Connectivity Range

In an indoor environment the average range achieved, over a series of 10 tests, was

System Costing

The following table shows the costs of all the elements included in the project:

TABLE 8.2: Costing for the System

Motion-Tracking Sensor Device: Prototyping Model	
SparkFun nRF52832 Breakout Board	R294,86
MPU-9250 Breakout Board	R174,56
SparkFun Beefy 3 FTDI Basic Breakout	R 220,96
Total	R690,38
Development Tools	
Nordic PCA10040 Development Kit	Already in UCT Store (R576,42 Value)
Altium Designer	UCT License: free
Full PCB Sensing Device Model	
Laird BL654 Module	R167,92
MPU-9250	R157,11
Additional Components (Resistors, Capacitors, Oscillator, LED, Push Button, Coin Cell Battery Holder, Headers)	≈R80
Coin Cell Battery	≈R10
Total	≈R415,03

Chapter 9

Conclusions

9.1 Testing Results

All of the tests specified in the Methodology, Section 3.3, were successfully completed.

The drift measurements from the IMU were much larger than anticipated. This may suggest insufficient calibration of the module or inefficient implementation of the motion filters. Further testing and experimentation of the calibration and filters should be undertaken to solve this issue, as it is unlikely to be a flaw of the chip itself.

Another shortfall within the tests was in the receiving of data from only two characteristics concurrently. This was found to be due to the serialization of the Bluetooth gatt operation and, as such, each characteristic must be read one by one.

9.2 Satisfaction of Requirements

The device designs satisfied the Technical and Design Requirements: RD.T.01-RD.T.05 as the size of both PCB designs are well within the constraints of 5cm x 5cm; the testing results demonstrated a current draw of 1uA while in 9-axis mode and transmitting data; and the components used are within the budgeted amount of R1000. Due to the timing constraints, mobile device offline storage was not fully implemented and tested.

The Functional and User Requirements were all satisfied, and were verified and validated within the testing phases.

9.3 Overall Meeting of Objectives

An 'ActionTracer' Motion-Tracking Device prototype and the framework of a corresponding mobile application were successfully created. The objectives were only just met, as these modules have a large amount of room for development and refinement, but the core requirements have been successfully implemented and tested.

Chapter 10

Recommendations

As the objective of this project was to create a basic framework of a Sports Sensing Device and Mobile Application, there is a large amount of potential for growth and refinement of the system elements. Several improvements and expansions for the Device and Application are detailed below.

10.1 Sensing Device Development

Software

The next stage of development should see the development of the Device's software in a more efficient IDE and language, such as SES. This would reduce the size of the application and provide a more flexible platform for its development.

Connectivity Improvements

Several improvements can be made to the Device's wireless protocol implementation. An important feature to add is connection security between the Device and smart phone. Especially with professional athletes, the privacy of their training and performance data is imperative. BLE is easily intercept-able but there are well-documented methods in the SDK for including security procedures, such as using the BLE Security Manager with pre-shared AES (Advanced Encryption Standard) key or implementing application-layer encryption.

A feature not covered in this project is the combining of multiple Devices in a sensor network. This is also an important feature to add as the use of more than one sensor in technique analysis is often required. The nRF5 SDK includes support for the Bluetooth mesh profile, which allows for a many-to-many network topology. An alternative that could be investigated is the integration of the ANT^{TM}protocol between devices.

An investigation into the efficiency of data transfer methods using BLE could be undertaken. This involves determining the optimal number of characteristics and services set up for the Device versus the structure and frequency of the data packets being sent.

IMU Improvements

Further investigations into the types of filters, eg. the Madgewick/Mahony filters in comparison to..* and processing of the motion readings should be conducted.

Additional Features

Onboard storage should be included into the Device design as it is highly likely there will be cases where the athlete may not wish to use their phone while training. The Device can then store the data and transfer it to a smart phone at a later stage. This would also assist with the prevention of data loss due to a loss of Wireless connection between the Device and smartphone while recording. The chip's internal memory may be able to act as a small buffer in such a scenario, but the inclusion of external memory will provide a more reliable system and reduce the risk of this buffer overflowing.
External housing for the device should be designed, based on factors of comfort (in the case of on-body use) or unnoticeable impact on the piece of equipment being used. This is highly dependent on the sport or application the device is being used for.

10.2 Application Development

Connectivity Development

The number of characteristics being read, and the most efficient method of doing so, from the Android application should be investigated so that a full complement of motion sensor readings can be accessed.

IMU Interface

The 9-Axis requires calibration for the magnetometer. A calibration guideline and procedure should be included in the application which will automatically start on connecting with the Sensing Device and be accessible for the user to initiate again when desired, as sometime multiple calibrations are required for long training sessions. This guideline would lead the user through the calibration process and implement the calibration function calls on the device.

Additional Features and Improvements

Useful additional features that could be added to the application are that of: user profile/log in functionality to increase data security and allow for easy cloud data backups; the integration of GPS location tracking; and the improvement and addition of the graphic representations of the motion data.

Appendix A

Schematic Diagrams

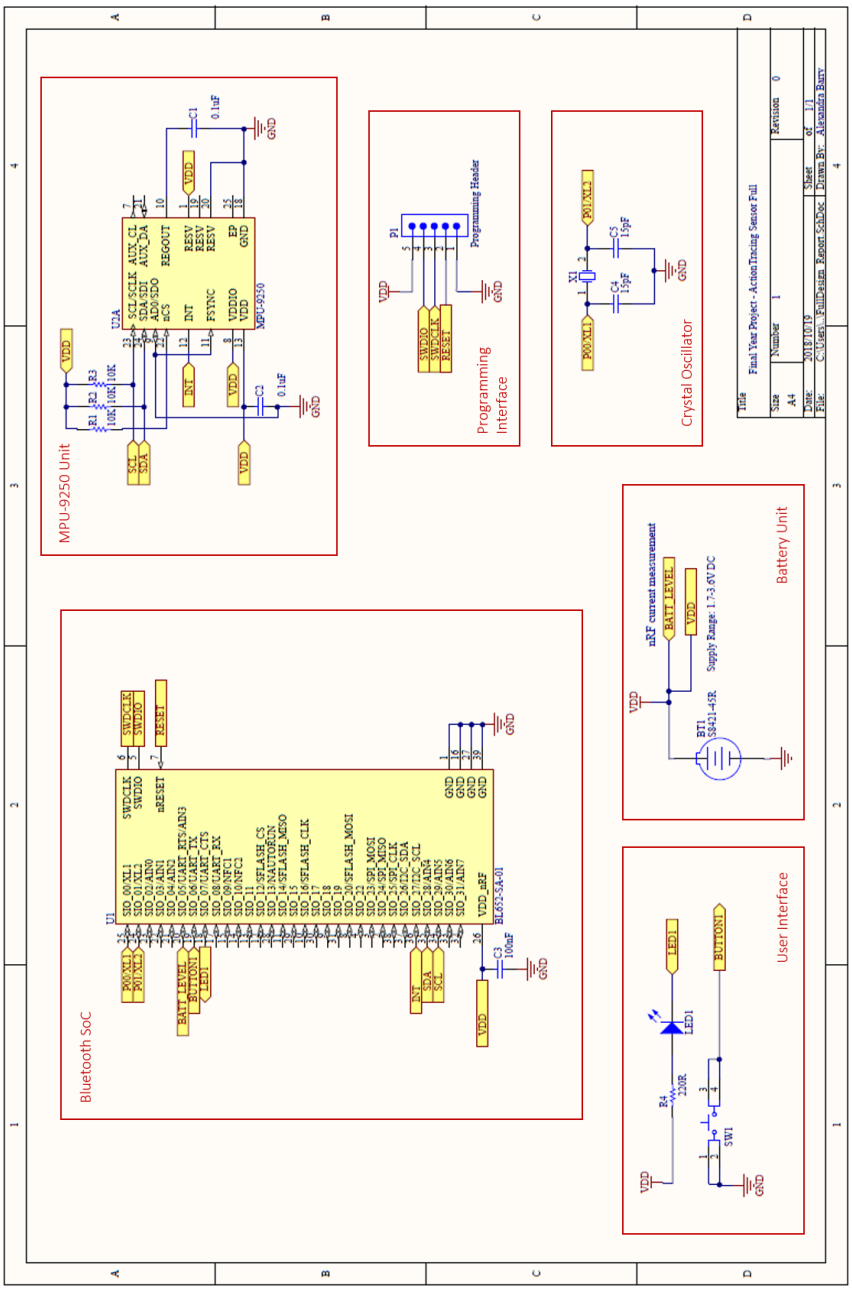


FIGURE A.1: Schematic of the Full Sensing Device Design

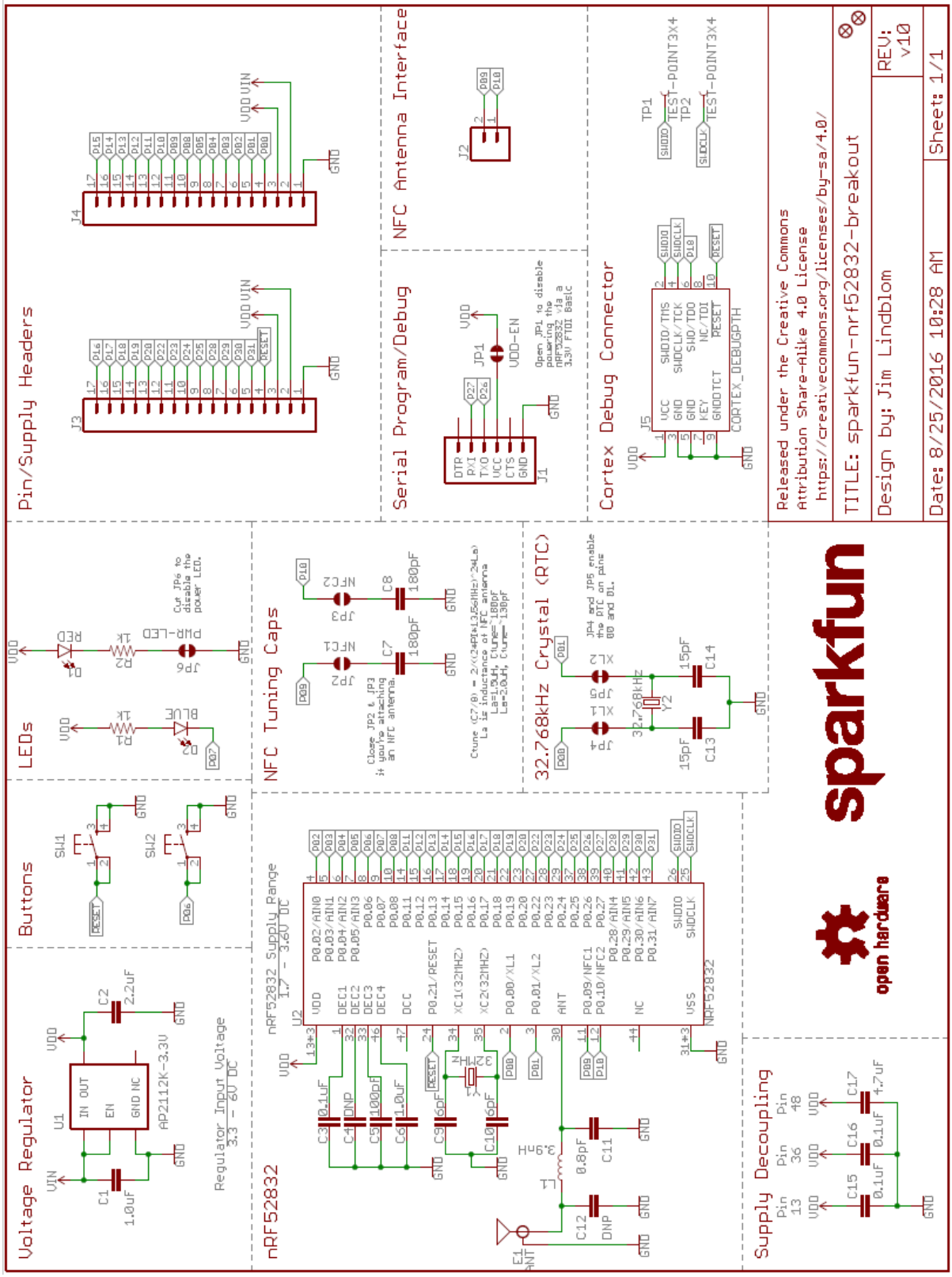


FIGURE A.2: SparkFun nrf52832 Breakout Board Schematic

Appendix B

Simulation and Testing Results

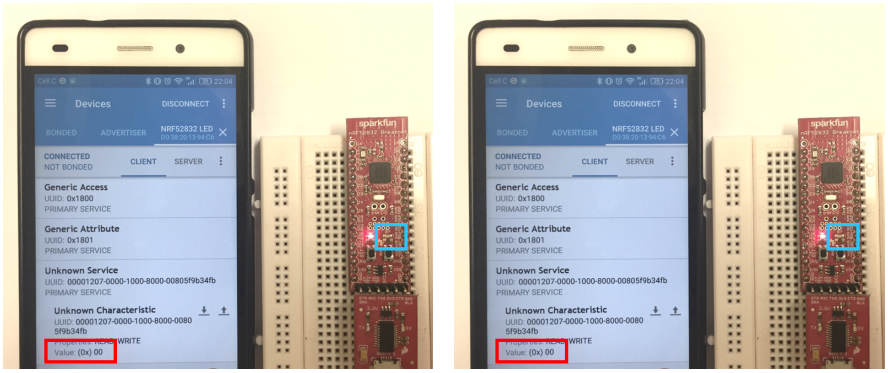


FIGURE B.1: Arduino BLE LED Control Examples

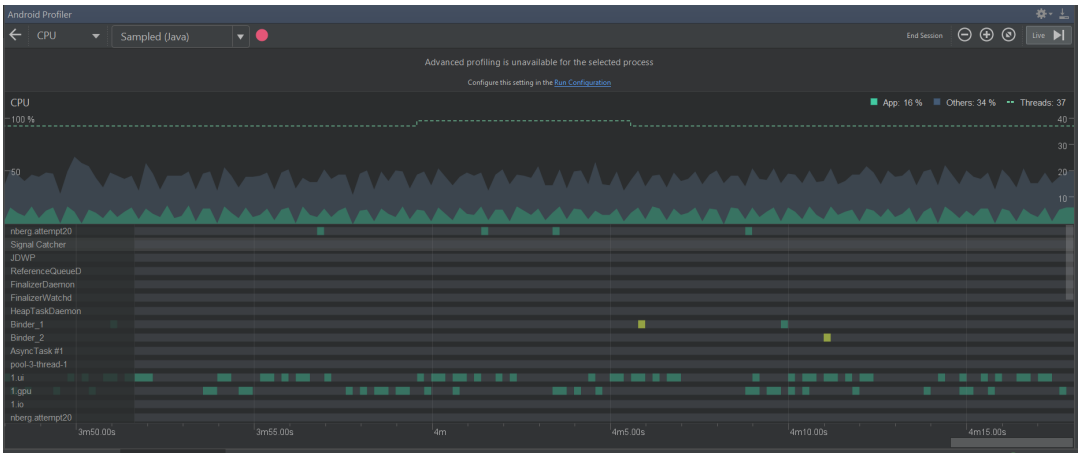


FIGURE B.2: CPU Application Profile Test 1 on Android Studio

References

[1] "Industry Statistics, Sports Industry Statistic and Market Size Overview", *Plunkett Research Ltd.*, 2018. [Online]. Available: https://www.plunkettresearch.com/statistics/Industry-Statistics-Sports-Industry-Statistic-and-Market-Size-Overview/. [Accessed: 11- Oct- 2018].

[2] S. Pritchard, "Marginal gains: the rise of data analytics in sport", *The Guardian*, 2015.

[3] "Sport Industry: The impact of technology on sport", *Leoisaac.com*, 2018. [Online]. Available: http://www.leoisaac.com/sportman/sportman06.htm. [Accessed: 11- Oct- 2018].

[4] D. Gouwanda and S. M. Senanayake, "Emerging trends of body-mounted sensors in sports and human gait analysis", *IFMBE Proceedings*, vol. 21 IFMBE, no. 1, pp. 715–718, 2008, ISSN: 16800737. DOI: 10.1007/978-3-540-69139-6-178.

[5] Talk Tennis, Crushing It. [pics]. 2018.

[6] TrackMyGolf, TrackMyGolf - GPS and Range Finder. 2018.

[7] Freepik, Smartphone Hand Vectors. 2018.

[8] "Étienne-Jules Marey | French physiologist", *Encyclopedia Britannica*, 2018. [Online]. Available: https://www.britannica.com/biography/Etienne-Jules-Marey. [Accessed: 11- Oct- 2018].

[9] Mérillon, D. (1901). *Concours Internationaux d'exercices physiques et de sports.* 1st ed. Paris: Impr. National. Paris, p.401.

[10] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi, "Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications", *International Journal of Signal Processing Systems*, vol. 1, no. 2, pp. 256–262, 2013, ISSN: 23154535. DOI: 10.12720/ijsps.1.2.256-262. [Online]. Available: http://www.ijsps.com/index.php?m=content{\&}c=index{\&}a=show{\&}catid=32{\&}id=65.

[11] *Zepp*, 2018. [Online]. Available: https://www.zepp.com/en-us, [Accessed: 15- Oct- 2018].

[12] *Trace*, 2018. [Online]. Available: http://www.traceup.com/, [Accessed: 15- Oct- 2018].

[13] "Notch: Motion Capture for Smartphones", *Notch*, 2018. [Online]. Available: https://wearnotch.com/. [Accessed: 20- Oct- 2018].

[14] "MbientLab – Smart Wireless Sensors and a Machine Learning Cloud for Motion Recognition", *Mbientlab.com*, 2018. [Online]. Available: https://mbientlab.com/. [Accessed: 20- Oct- 2018].

[15] "HEAD Tennis Sensor", Head.com, 2018. [Online]. Available: https://www.head.com/us-CA/sensor/. [Accessed: 20- Oct- 2018].

[16] R. P. Troiano, J. J. McClain, R. J. Brychta, and K. Y. Chen, "Evolution of accelerometer methods for physical activity research", *British Journal of Sports Medicine*, vol. 48, no. 13, pp. 1019–1023, 2014, ISSN: 14730480. DOI: 10.1136/bjsports-2014-093546. arXiv: NIHMS150003.

[17] G. Wetzstein, *"Inertial Measurement Units I"*, EE 267. Lecture Notes, Stanford University, 2017, [Accessed: 14- Oct- 2018].

[18] *Mems: Microelectromechanical systems*, CSE466. Lecture Notes, University of Washington, 2017, [Accessed: 15- Oct- 2018].

[19] A. Burg, A. Meruani, M. Wickmann and B. Sandheinrich, "MEMS GYROSCOPES AND THEIR APPLICATIONS". Evanston: Northwestern University, 2018.

[20] R. Verrinder and Y. A. Gaffar, *"Embedded Communication: Serial Peripheral Interface"*, EEE3017W. Class Lecture, University of Cape Town: Department of Electrical Engineering, 2017, [Accessed: 10- Oct- 2018].

[21] , P. Hindle, "History of Wireless Communications", Microwave Journal, 2015.

[22] , F.L. Lewis. "Wireless Sensor Networks." *Smart Environments: Technologies, Protocols, and Applications*, ed. D.J. Cook and S.K. Das, John Wiley, New York, 2004. Automation and robotics research institute. 26 Oct. 2013.

[23] (2018). "lmsc, lan/man standards committee (project 802)", Ieee802.org, [Online]. Available: http://www.ieee802.org/. [Accessed: 08- Oct- 2018].

[24] B. Ray, "A Bluetooth & ZigBee Comparison For IoT Applications", Link-labs.com, 2018. [Online]. Available: https://www.link-labs.com/blog/bluetooth-zigbee-comparison. [Accessed: 22- Oct- 2018].

[25] Nordic Semiconductor, 2018. [Online]. Available: https://www.nordicsemi.com. [Accessed: 21- Oct- 2018].

[26] SparkFun. (2018). "sparkfun nrf52832 breakout", [Online]. Available: https://www.sparkfun.com/products/13990. [Accessed: 10- Oct- 2018].

[27] *Nrf52832 breakout board hookup guide*, 2016. [Online]. Available: https://learn.sparkfun.com/tutorials/nrf52832-breakout-board-hookup-guide#ble-blink-example, [Accessed: 12- Oct- 2018].

[28] Polidea, *"flutterblelib"*, https://github.com/Polidea/FlutterBleLib, 2018, [Accessed: 10- Oct- 2018].

[29] nRF5 Series: Developing with SEGGER Embedded Studio, 1st ed. Nordic Semiconductor, 2018.

[30] D. Team, *"dart2_fix 1.0.6"*. [Online]. Available: https://pub.dartlang.org/packages/dart2_fix#-readme-tab-.

[31] ——, *"ByteData class"*. [Online]. Available: https://api.dartlang.org/stable/2.0.0/dart-typed_data/ByteData-class.html.

[32] S. Grider, *"Dart and Flutter: The Complete Developer's Guide"*, 2018. [Online]. Available: https://www.udemy.com/dart-and-flutter-the-complete-developers-guide/learn/v4/overview.

[33] M. Scwarzmuller, *"Learn Flutter & Dart to Build iOS & Android Apps"*, 2018. [Online]. Available: https://www.udemy.com/learn-flutter-dart-to-build-ios-android-apps/learn/v4/overview.