Aston University
Birmingham

SquashT

The Squash Tracking Android Application

Android - Arduino - Bluetooth 4.0 - Data Mining

Computer Science

Author:
Sam Robert Hendriksen (119074026)

Supervisor:
Dr George Vogiatzis

28th April 2015

[PROJECT DEFINITION FORM FRONT GOES HERE]

[PROJECT DEFINITION FORM BACK GOES HERE]

Contents

1 Abstract

The purpose of this project is to explore the feasibility of analysing a Squash player's match using an Arduino device with an accompanying Android device to display the match data. The feedback provided to users demonstrates the weighting of shots played, whilst extracting details on their most and least successful shot types to help with targeted training. To understand the types of swings being performed, data mining tools are used on the Android device to help extract swing features communicated over Bluetooth 4.0 (Low Energy). To ensure that the end product rivals its competitors, cost and size of the Arduino device have been taken into consideration to make sure that the product can be used by casual players and adapt to any racket size.

2 Introduction

In recent years there have been many advances in sport analytics that help players from a variety of sports record and analyse their skills based on feedback provided by assisting devices. Through the use of these devices players can be made aware of their subconscious motions and improve the way they play, which could potentially help reduce the time it takes to advance their technique/s.

Squash is a sport that has not fully developed accompanying feedback devices. The current devices on the market (see Background section) are aimed at higher calibre players, which makes these devices less applicable to a large portion of casual Squash players. The devices are also integrated into expensive rackets or have standalone variations that cost more than what the average player is willing to pay, as demonstrated by a survey created for this project that will be explained further in the following section. By introducing a device into the Squash market that it aimed at casual players, has the flexibility to attach to any racket, and has a relatively low cost, a large percentage of individuals will have the option to analyse and improve their games. SquashT is a "Squash Tracking" Android application that aims to demonstrate the feasibility of helping players understand the weighting of shots that they play during their games. This will help players understand the shots that they are failing to play or in some cases the shots that they are being forced to play by their opponents. In a typical Squash match multiple games are played, between three and five, in which players take a short break between each one. This is where SquashT fits in; when the player comes off the court they can read an overview of their game and recommendations suggesting improvements for any upcoming games. This will take into account the shots that won them the most points in their last game and the shots that they should avoid because of their low success rate.

The project as a whole explores multiple very interesting technological areas of Computer Science. The project looks at portable devices - a popular area of computing in the past decade, hardware - the building blocks of every computer device, Bluetooth 4.0 communications - a new and often overlooked method of communication and finally data mining - the ability for technology to understand and predict patterns in large amounts of information. Combining these elements provide a very interesting glance at the possibilities and limitations of the technologies and their appliances.

This report will begin with background research aimed at understanding the overall price and functionality needed by project. Following this will be a section on the initial project preparation, which will look into the process of designing the Android application and testing some potential Bluetooth modules against the end price requirements. Using the preparation data the four technologies mentioned above will be explained in the 'Deliverable' section describing the challenges faced and how they were overcome. Each iteration of the gesture-tracking device will then be explained with the final costs broken down based on the required components. The report will then evaluate the end product against the initial requirements to help demonstrate the feasibility and success of the project. Finally the work performed will be concluded providing future aspirations and ideas to move the project forwards.

3 Background

To ensure that SquashT was implemented correctly, research was conducted for this project in order to find similar applications that deal with activity detection and research papers that provide a more detailed analysis of the task at hand. Currently there are no applications for Squash that could be used to directly base the project on, which meant that applications for other sports and activities were chosen to help understand how to implement the project. Some of these activities included tennis, golf and everyday activity detection.

The first relevant application, a Tennis application (App Store, 2014), allows a bystander to enter the shots being performed by a player during a match. After a game players are able to understand what shots they played frequently and how these shots affected the end score. I feel that this a major feature that could help a player improve their matches. The problem with this application is that an observer has to manually log the shots being played, which is quite cumbersome and cannot be done independently. I find this idea very interesting and would hugely benefit players if the data was recorded effortlessly. Finding a solution to this issue required research into applications that could automate and record part of a player's game. From the research, two tennis applications were discovered that implemented a device for swing analysis; Zepp (Zepp.com, 2014) and Smart Tennis Sensor (Smarttennissensor.sony.net, 2014). These products are currently the best tennis analysers on the market. However, I feel there are couple of problems with these devices. The first problem being the price of each device, Zepp - £130 and Sony - £120 (at the time of writing), which will discourage casual players from using the devices. In my opinion casual players are the ones that would benefit most from these applications because they do not have the same access to coaches and equipment as professional players do. The second problem is related to an individual's grip preference because some players tend to hold the racket towards the bottom of the grip, which means that the Zepp users knock the device located on the bottom of the racket when they swing causing discomfort and unreliable results. With these two problems in mind, the aim for SquashT was to produce a device that can easily attach to any location on a user's racket without being obstructive, and to also be affordable enough for all types of players to buy the device.

To understand how much casual players are willing to spend on a Squash analytical application a user survey was produced (see appendices 10.6). In this survey users were asked how much they are willing to spend on a device that is able to track their movements, shots and score during their games to help them improve. Out of the 17 users that participated, nobody voted that they were "Not Interested in the Concept", and the price that the majority (47%) were willing to spend was £25-£30 on this theorised device. Not only does this demonstrate that the target users are very interested but it also sets a benchmark for the end price of the device, because existing applications are too expensive compared to what the average player is willing to pay (based on this survey).

The next and major part of the research was to determine how to detect an individual's unique set of swings correctly. This required multiple research papers that described common steps and resources to use when detecting activities. The first useful paper described the detection of multiple user activities ranging from lying down to walking up stairs (Cleland et al., 2013). The key information that was taken away from this paper involved converting the raw accelerometer data into key features, for use in a library called Weka (Cs.waikato.ac.nz, 2015a). Not only did this paper rely on the use of Weka but multiple other activity based research papers also used this library to detect the correct movements for their research. The main differences between the papers were the types of features they extracted and the algorithms they used (Bayat, Pomplun and Tran, 2014; Catal et al., 2015; Kwapisz, Weiss and Moore, 2011). This method of detection is more flexible than hardcoding logic because it allows different users to tailor the application to their swing styles. For this reason SquashT was developed using a dynamic method of detection (Weka) to ensure that the application can be used by players of all levels and skills, whilst still being able to detect each individual's unique swing styles.

In Squash, a coach's role is to observe a player's game and inform them of their strengths and weaknesses so that they can improve. Without this feedback individual players will struggle to improve their game at a frequent pace. As mentioned by Alamar (2013) sports analytics can greatly help and assist coaches to determine areas in a player's game that they should improve. SquashT aims to reduce this feedback loop and provide higher-level details directly to players, which will reduce the need for a coach and enable players to analyse their games themselves at any point.

Based on the above research it is possible to conclude that the SquashT application is definitely feasible and there is currently a place in the market for this product. The main factors to ensure the product's success can be measured by the device's end cost (£25-£30), weight, flexible positioning and functionality.

3.1 Squash Research

Before implementing the SquashT application the Squash specific details, such as swings, needed to be understood to ensure that the requirements were correctly defined. The first and most important part of the research was to understand the total number of shots possible in Squash and which types the application should record. A useful article by Talabi (2006) described the mean percentage of shots during an average Squash game and concluded that the following shots were played the most; “Drives”, “Volleys”, “Drops”, “Boasts” and “Lobs”. Because SquashT is aimed at casual players these defined shots will provide enough precision without needing to record the less frequent shot types. However, the “Boast” shot type was omitted from the SquashT application due to the fact that it is played using the same style as a “Drive” with the only difference being the direction of the player when the shot is performed, which would prove difficult to differentiate.

4 Preparation

4.1 User Interface Sketches

To understand the types of screens that the application required a program called “JustInMind” (Justinmind.com, 2015) was used, which helps design graphical user interfaces. Below are the screens that were designed for the SquashT Android application.

Profile Selection Screen

The profile selection screen (figure 1) is first shown to users when they open the SquashT application. This allows different users to load the application with their own profiles, which helps correctly detect their style of swings. New users of the application can select the ’add user’ button at the top right of the screen, which will present them with a simple text dialog for them to enter their profile name. To remove a profile, users will simply long hold an item in the list, which will present them with a deletion dialog.

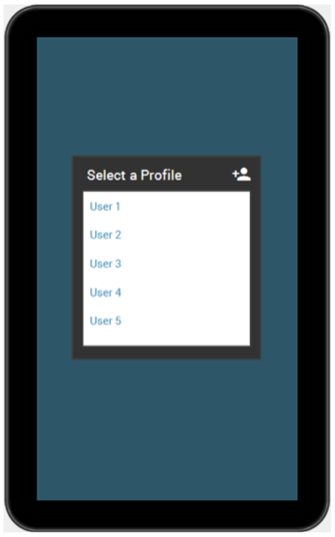


Figure 1: Profile selection.

Navigation Panel

The navigation panel (figure 2) uses the “Hub and Spoke” design pattern (Tidwell, 2005), which allows users to focus on one task at a time. Each menu item consists of a title and an icon, which represents the screen’s actions so that quick associations can be made by frequent users. Each icon also incorporates the similarity principle (Tidwell, 2005, p.139) to help users understand the relationship between each item and their importance.

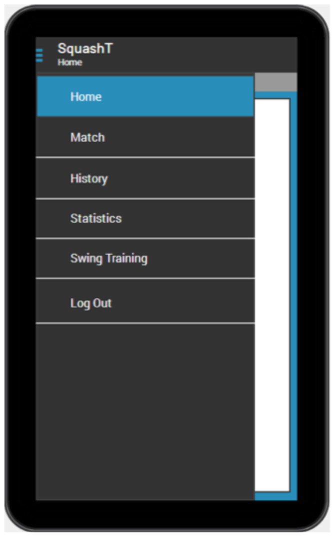


Figure 2: Navigation panel.

Home Screen

The home screen is the first screen that users will be taken to and is used to introduce the application. It will also provide a helpful image about the functions of the SquashT device, such as powering it on, to help users understand how to correctly use the device.

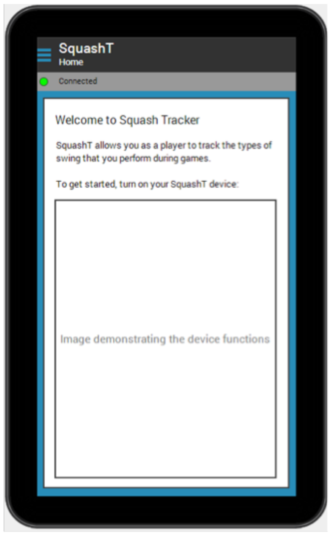
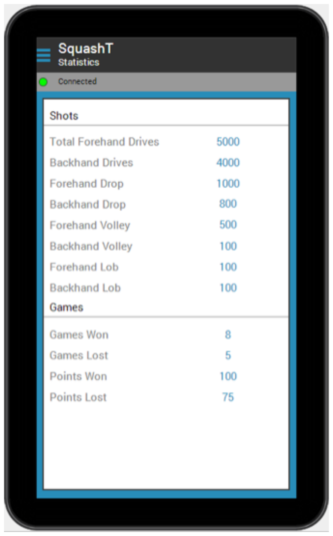

SquashT
Home
Connected
Welcome to Squash Tracker
SquashT allows you as a player to track the types of swing that you perform during games.
To get started, turn on your SquashT device:
Image demonstrating the device functions

Figure 3: Home screen.

Statistics Screen

The statistics screen is used to show global match information to users. Each match that has been saved to the application's database will be used to form the statistics and help players understand their overall game style and habits.


SquashT
Statistics
Connected

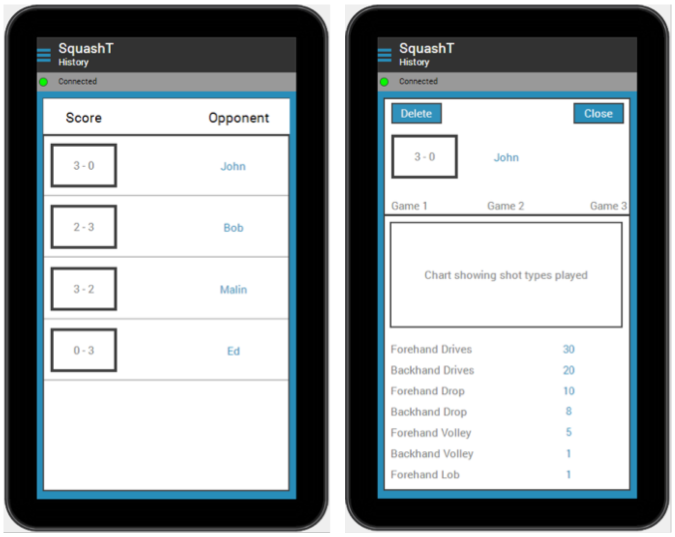
Shots	
Total Forehand Drives	5000
Backhand Drives	4000
Forehand Drop	1000
Backhand Drop	800
Forehand Volley	500
Backhand Volley	100
Forehand Lob	100
Backhand Lob	100

Games	
Games Won	8
Games Lost	5
Points Won	100
Points Lost	75

Figure 4: Statistics screen.

Match History Screens

The match history screen (figure 5a) allows players to revisit and analyse their previous matches and games. As a brief overview the list will display the match scores and the opponents name to help find a specific record. To access the in depth details users can select an item in the list, which will bring up the full match detail page (figure 5b). The match details screen provides users with details about their past games, which includes the scores, count and types of shots that were played in each game.


SquashT
History
Connected

Score	Opponent
3 - 0	John
2 - 3	Bob
3 - 2	Malin
0 - 3	Ed

SquashT
History
Connected

Forehand Drives	30
Backhand Drives	20
Forehand Drop	10
Backhand Drop	8
Forehand Volley	5
Backhand Volley	1
Forehand Lob	1

(a) Match history list.
(b) Match history details.

Figure 5: Match history.

Initial Training Screen

The initial swing training screen (figure 6) allows new players to train the application with their style of swings. The application will ask players to take a fixed amount of each swing (for example five) to help build up a swing classifier (base for future swings to be detected). When a swing is taken the application will be updated reflecting the amount of shots required for that swing type. Once all shots for each swing type have been taken the application will save the results to the user's profile.

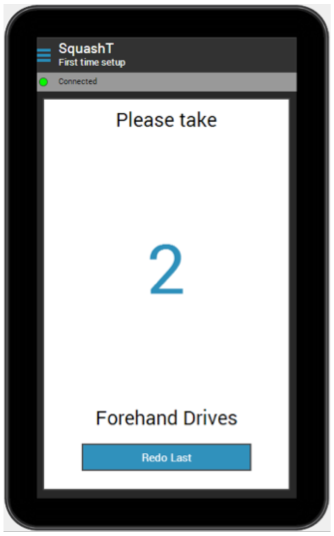
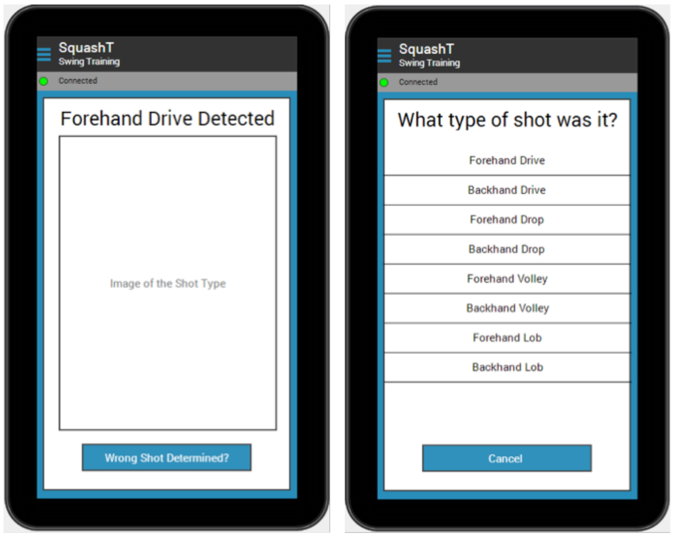


Figure 6: Initial swing training.

Training Screen

The training screen (figure 7a) allows users to test the accuracy of their profile and correct any incorrectly classified swings. When a user performs a swing the determined swing type will be shown on screen, which if incorrect can be corrected using the swing correction screen (figure 7b) accessible via the button at the bottom of the screen. The swing correction screen allows users to select the correct swing type that was performed so that the application can correct its data and retrain. The training screen has been designed to help make the swing detection clear to users by incorporating the “Centre Stage” design pattern (Tidwell, 2005, p145). This ensures that the swing type image retains the majority of the screen so that it can be seen further away, which will benefit users that do not hold the device when testing their swing accuracy.

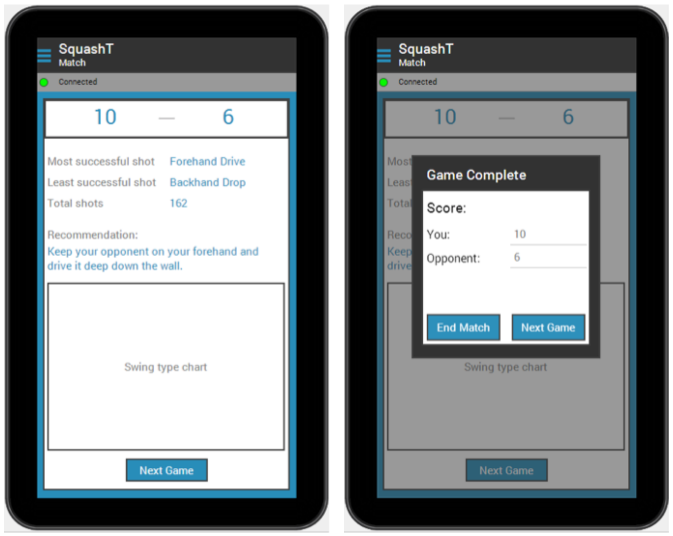


(a) Swing detection screen. (b) Swing correction screen.

Figure 7: Training screens

Match Screen

The match screen (figure 8a) is used to provide players with helpful information after they come off the court. This will include game-play recommendations and a chart depicting all of the shots that were played in the game. Once a user is finished with their game they can then save the results (figure 8b) and move onto the next game in the match. They can also end the match through the same dialog if they have finished.


SquashT
Match
Connected
10 — 6
Most successful shot Forehand Drive
Least successful shot Backhand Drop
Total shots 162
Recommendation:
Keep your opponent on your forehand and drive it deep down the wall.
Swing type chart
Next Game

SquashT
Match
Connected
10 — 6
Game Complete
Score:
You: 10
Opponent: 6
End Match
Next Game
Swing type chart
Next Game

(a) In-progress game. (b) Complete game.

Figure 8: Match screen.

4.2 Bluetooth Feasibility Tests

To allow the Android and Arduino devices to communicate, a Bluetooth module was needed that met specific requirements to ensure the final device's success. The prominent modules were tested against cost, size, power usage and reliability.

4.2.1 HM-10 Module

The first module to be tested against these requirements was the HM-10 (Jnhuamao.cn, 2015). The module is well documented and has multiple online guides explaining how to connect and use the module. The main guide that was used (Letsmakerobots.com, 2015) explains how to create a breakout board that allows easy access to the ports on the module and also provides sample code to test the connection.

After following the guide and creating a HM-10 breakout I found that this was a viable method of communication as it used a low amount of energy and performed well on some basic reliability tests. The only problem with this module and breakout board (figure 9) was the fact that it would be too large to have on the Squash racket with the rest of the equipment; it also has many unneeded features and components that would add to the price of the board. For example this board has extra components to show when data is being sent or received from the module, which was very useful for debugging the communications but would not be needed by the final users.

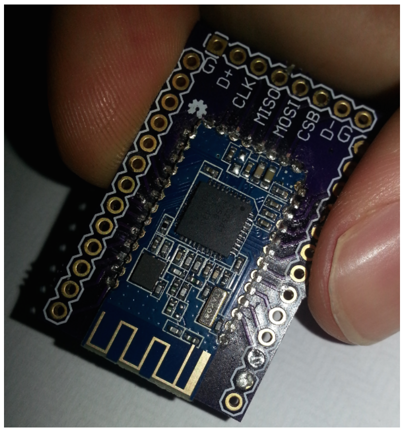
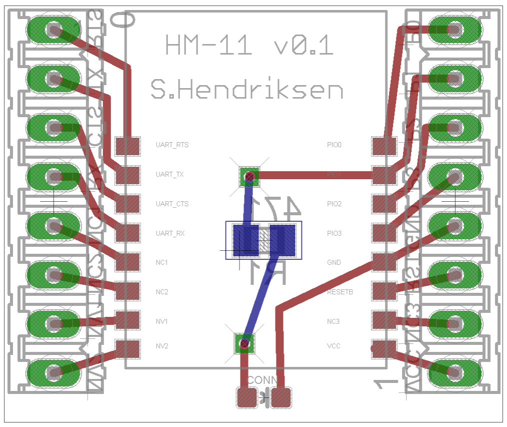


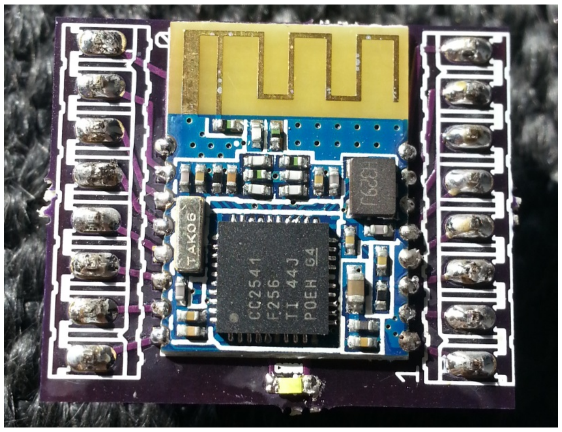
Figure 9: HM_10 Module Specifications - Width: 24mm, Length: 36mm, Price total: £10.50.

4.2.2 HM-11 Module

After seeing the HM-10 module working successfully the next step was to try and find a way of making the device smaller and cheaper. Because the module performed well under the requirements the best decision was to look for a variation of the same module. After looking through the manufacturers other products a HM-11 module was found (Jnhuamao.cn, 2015), which at the time of viewing was the smallest module they offered whilst maintaining the same functionality as the HM-10. Compared to the HM-10, the only feature that needed to be maintained for the new breakout was the connection status LED, as to alert the user of the current connection state. Therefore to understand the parts that would be needed to help reduce the cost, the module's data-sheet was used (JNHuaMao Technology Company, 2014), which explains the components needed for the functionality. This meant that the parts list for the new breakout included a HM-11 module, 0603 LED, 470ohm resistor and a board to combine the elements. Once the new parts list had been decided, research was needed to understand how to design a breakout board so that it could be printed. The highest rated software to design schematics was "Eagle PCB" (Cadsoftusa.com, 2015), which an online guide explained how to create schematic components (Instructables.com, 2015). After following the guide, and HM-11 data-sheet for the specifications, the breakout schematic was completed (figure 10a) and then uploaded to "OSH Park" (Oshpark.com, 2015) who printed and shipped the boards for £2.69. The components were then finally soldered onto the breakout board, as seen in figure 10b. The HM-11 module has surpassed the HM-10 by nearly half the price, whilst maintaining a smaller form factor, which means that it is a viable module for the project.



(a) HM-11 Eagle schematic.

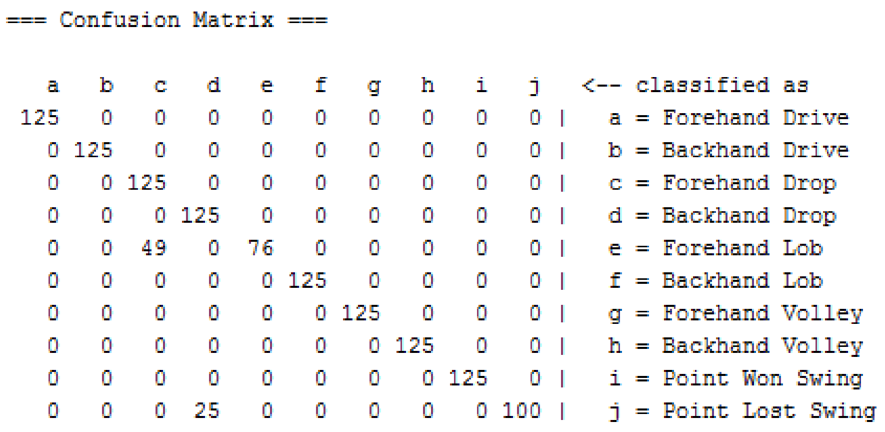


(b) Printed breakout board.

Figure 10: Hm_11 Module Specifications - Width: 26mm, Length: 21mm, Price total: £5.70.

4.3 Weka Basics

Before implementing Weka it was important to understand and identify the key terms and correct use of the library. With the help of the Weka resources (Weka.wikispaces.com, 2015a) the following terms can be understood; "Classifier" is an abstract name for the list of machine learning algorithms which predict the results, "Attributes" are the features that define the data being recorded e.g. accelerometer_x, "Nominal Attributes" are a predefined set of attributes which the data complies to e.g. "Forehand Drive", and "Instances" are individual data points of the attribute e.g. one x value in a series of x accelerometer values. Nominal attributes are the values that the classifier tries to predict using the set of instances of each attribute provided. When the classifier predicts the results it will display a "Confusion Matrix", which assigns instances to what it believes are the correct nominal types. As you can see in figure 11 the confusion matrix shows that 49 instances of the type "Forehand Lob" were incorrectly classified as a "Forehand Drop". This is where the classifier determines that the set of attributes provided closely resemble the other nominal type, which in this case was incorrect. To improve the accuracy, distinctive attributes need to be used that are only associated with a specific nominal type; this is one of the most difficult part of the data mining process, understanding the best features (attributes) to use.



```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   <-- classified as
 125   0   0   0   0   0   0   0   0   0 |   a = Forehand Drive
   0 125   0   0   0   0   0   0   0   0 |   b = Backhand Drive
   0   0 125   0   0   0   0   0   0   0 |   c = Forehand Drop
   0   0   0 125   0   0   0   0   0   0 |   d = Backhand Drop
   0   0  49   0  76   0   0   0   0   0 |   e = Forehand Lob
   0   0   0   0   0 125   0   0   0   0 |   f = Backhand Lob
   0   0   0   0   0   0 125   0   0   0 |   g = Forehand Volley
   0   0   0   0   0   0   0 125   0   0 |   h = Backhand Volley
   0   0   0   0   0   0   0   0 125   0 |   i = Point Won Swing
   0   0   0  25   0   0   0   0   0 100 |   j = Point Lost Swing
```

Figure 11: Weka Confusion Matrix

To understand how to use Weka in an Android environment, the site Weka.wikispaces.com (2015a) demonstrates how to call the Weka GUI functions from the code itself. This allows the SquashT application to do the data mining on the mobile device rather than having a computer counterpart calculate the predictions for it. The main items of interest for SquashT are the file output format and the building and running of classifiers programmatically, which the Weka documentation explains well (Weka.wikispaces.com, 2015b; Weka.wikispaces.com, 2015c). With the help of the resources the Weka library can be implemented into the Android application using a stripped down Weka library for Android (GitHub, 2011), which removes the unneeded features such as the GUI.

5 Deliverable

This section describes the final Android, Arduino and data mining capabilities. All artefacts and sources have been maintained and versioned using the versioning control system "Github" (GitHub, 2015a).

5.1 Android Application

The Android application is responsible for processing and collating a user's match and game data from a connected Arduino device. Using the stored data players can analyse past games and use the information to train and improve.

5.1.1 Development Environment and Libraries

To help deliver the final product, specific versions and libraries have been used by the Android application. The application has been developed for version 18 of the Android SDK, which means it is limited to devices running the Android OS 4.3 or later. This is because Bluetooth low energy is only supported on versions 18 or higher (Android, 2015).

The first external library being used is the "AndroidPlot" graphing library (AndroidPlot, 2015). This library helps display information such as the breakdown of swings in a bar chart. The next library being used is the "Apache Commons Mathematics Library" (Commons.apache.org, 2015), which helps perform some of the complex maths to calculate the features of each swing. The final and most important library is the "Weka" data mining library (GitHub, 2011), which helps determine what type of swing is being performed by analysing the swing features.

5.1.2 Software Architecture

From a higher-level overview the Android application was structured so that it contains one "Activity" (Developer.android.com, 2015a) and several "Fragments" (Developer.android.com, 2015b). The Activities' role is to control and display the fragments when needed, which has been learnt from prior experience and is recommended by Phillips and Hardy (2013). The application also contains an SQLite database, which was integrated into the Android framework and stores the match, game and swing data. The database is accessible via a singleton instance to ensure that there is only one instance during the application's life cycle.

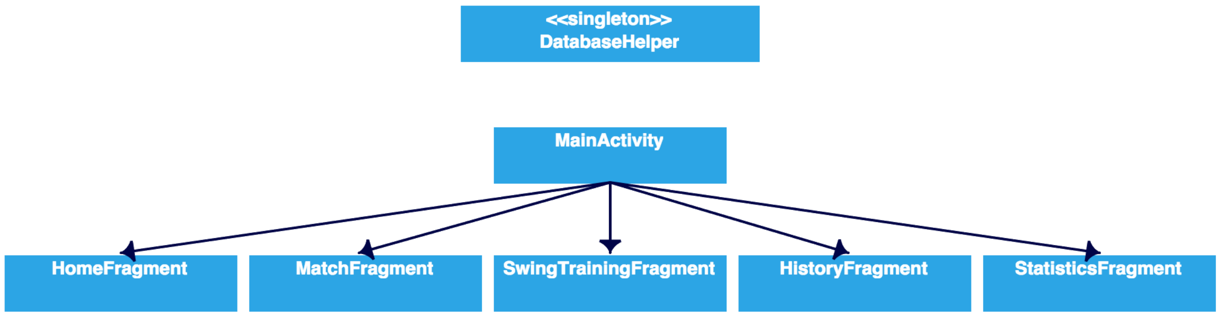


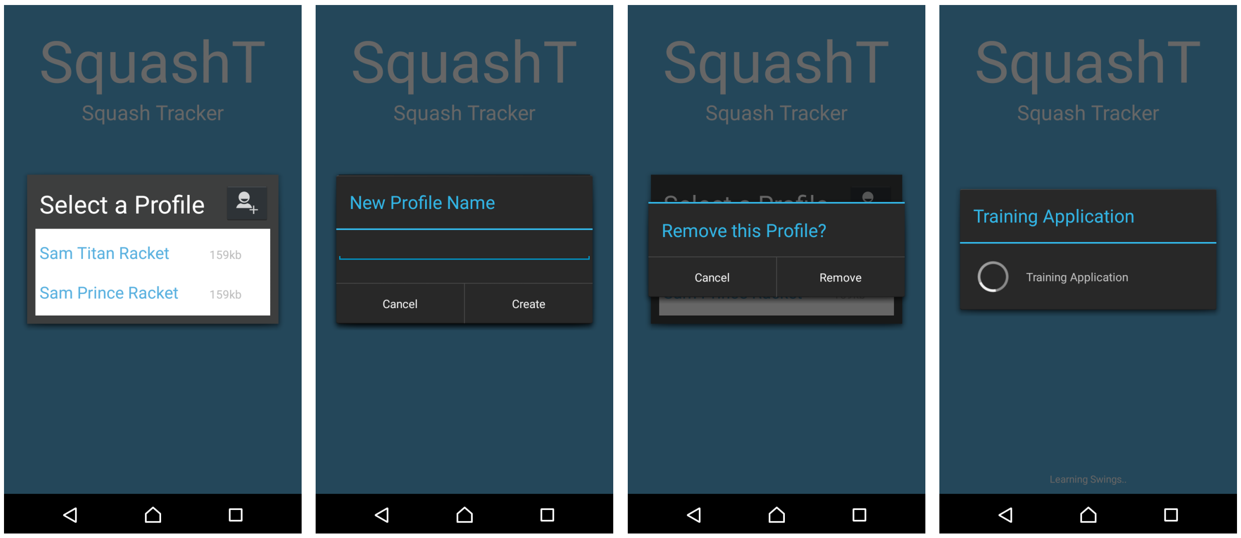
Figure 12: Android Basic Hierarchy.

5.1.3 User Interface

This section explains the rationale behind the final user interface designs and how each screen is accessed.

Profile Selection Screen

When a user first enters the SquashT application they will be presented with the profile selection dialog (figure 13a). The dialog enables users to create and load different swing presets depending on the player's swing style or racket type. When a new profile needs to be created a user simply presses the create user button at the top right of the profile selection dialog (figure 13a) to bring up the profile creation dialog (figure 13b). This dialog simply asks for a user's name and then loads the newly created profile on creation. If a user wants to remove a profile they simply have to long hold the profile to bring up the remove dialog (figure 13c). When a profile is being loaded the profile selection dialog is dismissed and a loading dialog is shown (figure 13d), which approximately takes three seconds to load a standard profile. Once a profile is loaded the user will be taken to the home screen (if it is an existing profile) or the introductory training screen (if it is a new profile).



(a) Profile selection. (b) New profile. (c) Remove profile. (d) Loading Profile.

Figure 13: User Profile Selection

Navigation Panel

The navigation panel (figure 14) is accessible on all screens of the application and can be made visible either by 'pulling' the drawer out from the left edge of the screen or by pressing the title at the top left of the screen. This method of navigation has been chosen so that users always have the option to navigate away from their current screen without becoming trapped or lost. Compared to the original UI sketches the menu now uses icons to help users quickly associate and find their desired action. The menu icons are designed and developed by Flaticon.com (2015), which allows the same design of images to be applied, enabling users to gain a sense of familiarity in the application.

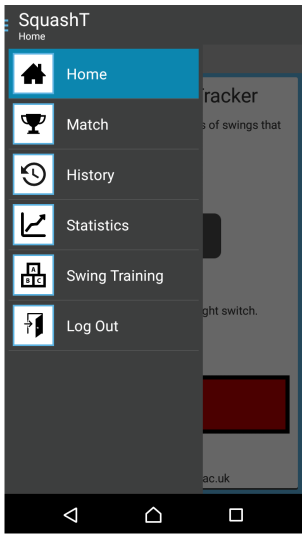
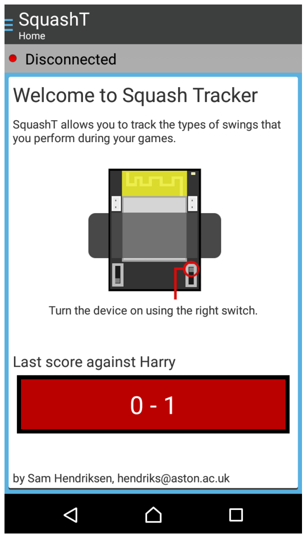


Figure 14: Navigation drawer.

Home Screen

The home screen (figure 15) is the first screen that users see when entering the application. The home screen consists of introductory text, a mock up Arduino device - to help demonstrate how to turn the device on and off, and finally the score from the last match, which can be clicked to view the match details. Compared to the original design the home screen now shows users their last match score, which has been chosen to help players quickly access their last match so that they can learn and improve before their next.


SquashT
Home

Disconnected

Welcome to Squash Tracker

SquashT allows you to track the types of swings that you perform during your games.

Turn the device on using the right switch.

Last score against Harry

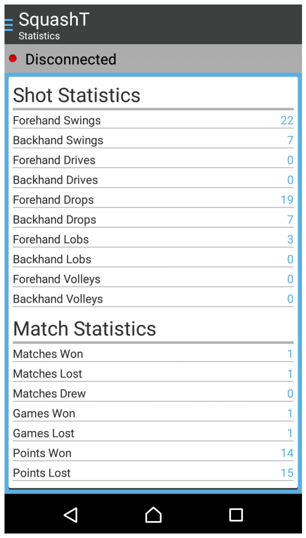
0 - 1

by Sam Hendriksen, hendriks@aston.ac.uk

Figure 15: Home screen.

Statistics Screen

The statistics screen is a brief and easy to understand overview of a user's matches and games. From the statistics users can understand what types of shots that they play and their breakdown of wins and losses. This information is provided to motivate users to try and increase their number of wins and increase their competitiveness.


SquashT
Statistics

Disconnected

Shot Statistics

Forehand Swings	22
Backhand Swings	7
Forehand Drives	0
Backhand Drives	0
Forehand Drops	19
Backhand Drops	7
Forehand Lobs	3
Backhand Lobs	0
Forehand Volleys	0
Backhand Volleys	0

Match Statistics

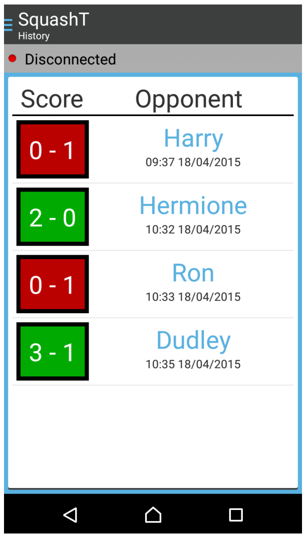
Matches Won	1
Matches Lost	1
Matches Drew	0
Games Won	1
Games Lost	1
Points Won	14
Points Lost	15

Figure 16: Shot and Match statistics.

Match History

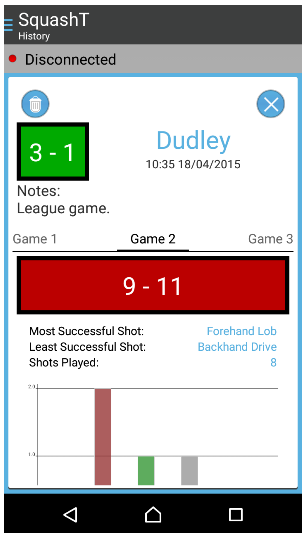
The match history screen allows users to view all of their previous saved games in an easy to view list (figure 17a). The list shows the score of the game, the name of the opponent and the date that the game was played. When a match in the list is selected, a full detailed page will be shown that provides details on the games and shots played (figure 17b). The games are tabular and can be scrolled through from left to the right, whilst the game information is in a list and can be scrolled up and down for more details (figure 17c). The shot type bar chart can be clicked to show a full screen view, which makes it easier for players to view the details.

Compared to the original designs, the game information was changed to show more relevant information such as the most and least successful shot type. This type of information proves more useful at a glance to players than the full swing type breakdown, which can be seen in the graph.


SquashT
History
● Disconnected

Score	Opponent
0 - 1	Harry 09:37 18/04/2015
2 - 0	Hermione 10:32 18/04/2015
0 - 1	Ron 10:33 18/04/2015
3 - 1	Dudley 10:35 18/04/2015

(a) Match history list.


SquashT
History
● Disconnected

3 - 1 Dudley 10:35 18/04/2015

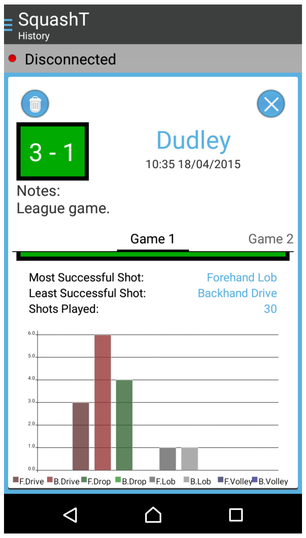
Notes:
League game.

Game 1 Game 2 Game 3

9 - 11

Most Successful Shot:	Forehand Lob
Least Successful Shot:	Backhand Drive
Shots Played:	8

(b) Second game overview.


SquashT
History
● Disconnected

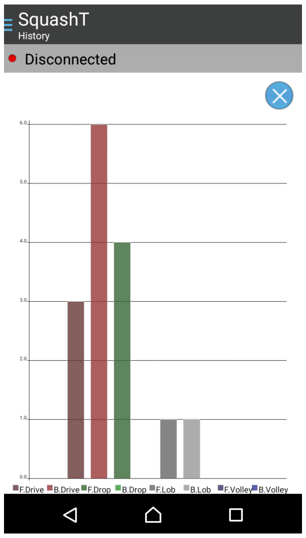
3 - 1 Dudley 10:35 18/04/2015

Notes:
League game.

Game 1 Game 2

Most Successful Shot:	Forehand Lob
Least Successful Shot:	Backhand Drive
Shots Played:	30

(c) First game overview.

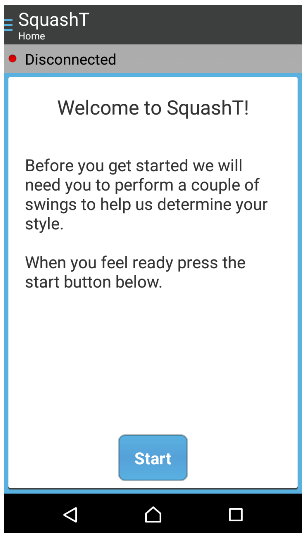

SquashT
History
● Disconnected

(d) Full screen chart.

Figure 17: Match history.

Initial Training Screen

When a new profile is created in the profile selection screen (figure 13), users will be taken to the initial training screen, which enables them to train up a new user profile (figure 18). The first screen that is shown introduces the initial training sequence and allows users to start when they are ready by pressing the "Start" button (figure 18a). Once the start button has been selected the swing training screen will be shown (figure 18b), which asks users to take three shots of each type of swing. When all of the swings have been taken, a save dialog will be shown (figure 18c) that allows users to save their swing data, which then redirects them to the home screen (figure 15).

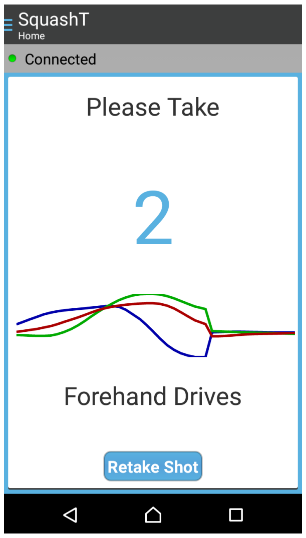

SquashT
Home
● Disconnected

Welcome to SquashT!

Before you get started we will need you to perform a couple of swings to help us determine your style.

When you feel ready press the start button below.

(a) Training introduction.

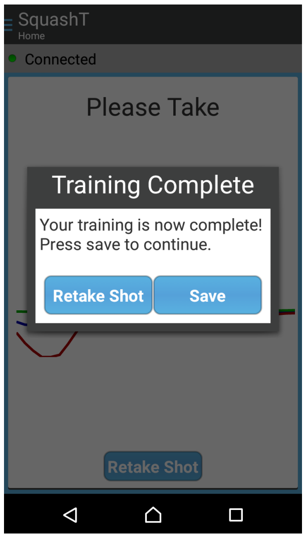

SquashT
Home
● Connected

Please Take

2

Forehand Drives

(b) Swing training.


SquashT
Home
● Connected

Please Take

Training Complete

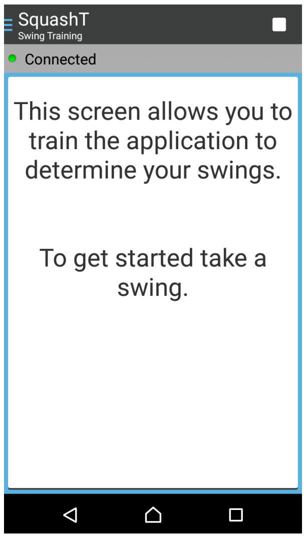
Your training is now complete! Press save to continue.

(c) Training complete.

Figure 18: Initial profile training.

Training Screens

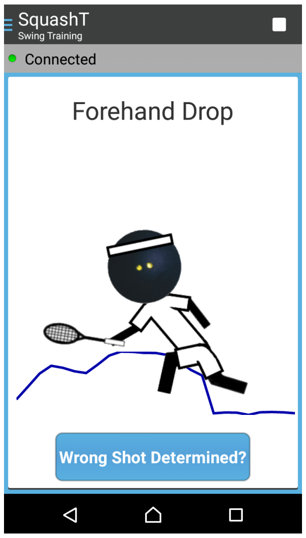
After the application has been trained by the initial training screens (figure 18) a user may still have problems with the detection of certain swing types. The training screens (figure 19) enable users to adjust the incorrectly detected swings by telling the application what the swing type should be detected as. When a user first enters the screen they will be presented with an introductory message that states that the swing detection can be started by making a swing (figure 19a). Once a swing is made, the swing type that the application detected will be displayed on screen (figure 19b), which users can view to see if the correct type was detected. In this view users are also able to see the magnitude of their swing in an axis-less graph, which shows them a glimpse of the lower level system. If a swing type is incorrectly detected then they can correct it by pressing the "Wrong Shot Determined?" button, which will take them to the correction screen (figure 19c). The correction screen provides users with a list of shot types that they can choose from. Selecting an item in the list updates the application to associate the swing made with the shot selected. This helps the application learn from the inputs and become more accurate as more data is provided. To aid debugging and for use during the project demonstration, the confusion matrix seen in figure 19d can be shown by long holding the swing type image in figure 19b.



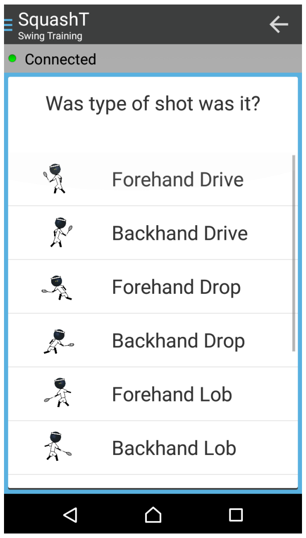
This screen allows you to train the application to determine your swings.

To get started take a swing.

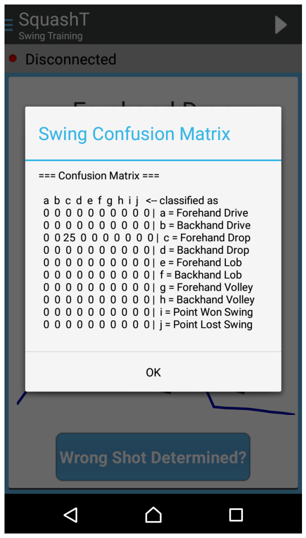
(a) Training introduction.



(b) Swing type detection.



(c) Swing type correction.



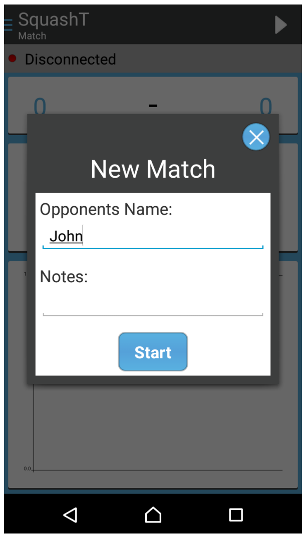
```
=== Confusion Matrix ===

 a  b  c  d  e  f  g  h  i  j   <-- classified as
 0  0  0  0  0  0  0  0  0  0 |  a = Forehand Drive
 0  0  0  0  0  0  0  0  0  0 |  b = Backhand Drive
 0  0 25  0  0  0  0  0  0  0 |  c = Forehand Drop
 0  0  0  0  0  0  0  0  0  0 |  d = Backhand Drop
 0  0  0  0  0  0  0  0  0  0 |  e = Forehand Lob
 0  0  0  0  0  0  0  0  0  0 |  f = Backhand Lob
 0  0  0  0  0  0  0  0  0  0 |  g = Forehand Volley
 0  0  0  0  0  0  0  0  0  0 |  h = Backhand Volley
 0  0  0  0  0  0  0  0  0  0 |  i = Point Won Swing
 0  0  0  0  0  0  0  0  0  0 |  j = Point Lost Swing
```

(d) Confusion matrix dialog.

Figure 19: Swing type training.

Match Screen

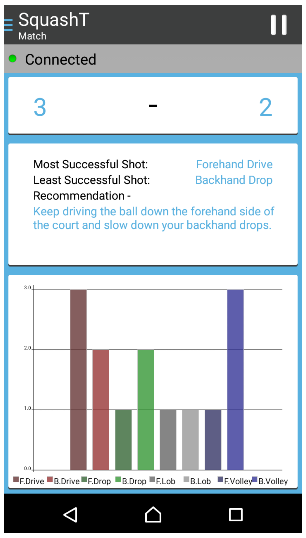
The match screen enables users to record and receive feedback on their current match and games. When a user first enters the screen they will be provided with a new match dialog (figure 20a), which asks for the opponents name ("Unknown" is used when the field is left empty) and any notes. Once the start button is pressed and an Arduino connection is made the main match screen is shown (figure 20b), which tracks every shot the player makes. The match screen keeps track of the current score, the most and least successful shot types, recommendations on how to progress and finally a bar chart breaking down the shots played. If the bar chart is selected then a full screen live updating bar chart will be shown much like the match history screen. At the top of the screen users can pause the current game using the pause icon or they can move to the next game in the match by pressing "next game" button, which will bring up the game complete dialog (figure 20c). The 'Game Complete' dialog allows corrections to be made to the score and allows users to move to the next game or end the match entirely. If the match has ended then the match complete dialog is shown (figure 20d) with the final game scores and the option to save or discard the match. Compared to the original designs the match screen now implements three new buttons; play, stop and pause. When the play button is pressed only the pause button is shown. When the pause button is pressed the start recording, stop recording and next game buttons are shown. Finally when the stop button is pressed the menu only shows the play button and the end game dialog is shown.



Opponents Name: John

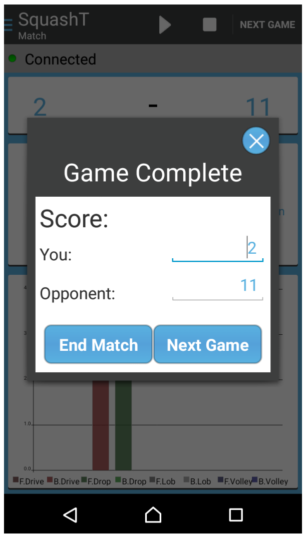
Notes:

(a) New match dialog.



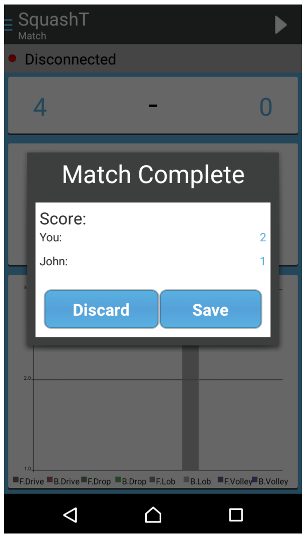
Most Successful Shot: Forehand Drive
Least Successful Shot: Backhand Drop
Recommendation -
Keep driving the ball down the forehand side of the court and slow down your backhand drops.

(b) Recording match.



Score:
You: 2
Opponent: 11

(c) Game complete dialog.



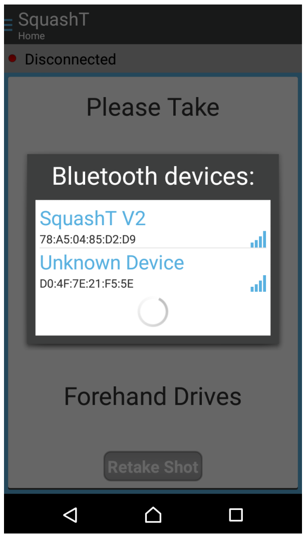
Score:
You: 2
John: 1

(d) Match complete dialog.

Figure 20: Match screen.

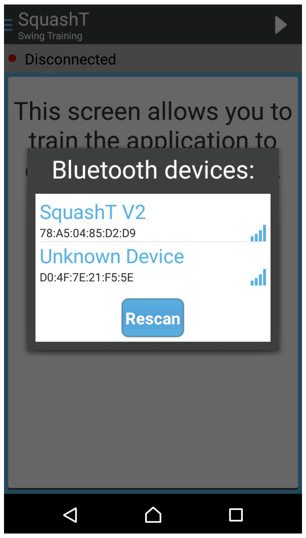
Connecting to the Arduino Device

When users enter a screen that requires a connection with the Arduino device (if they are not already connected), such as the match screen, they are presented with a connection dialog. The Bluetooth connection dialog enables users to simply select the device that they want to connect to, which activates the connection and dismisses the dialog. When a connection state change is made the sub action bar is updated to reflect the change, such as "connected" or "disconnected". The dialog scans for ten seconds and provides users with an option to rescan for their device again if it was not found (figure 21b). The selection dialog only enables users to select a Bluetooth device that corresponds to the specific HM-11 module requests. If a user selects a device that does not respond with the correct signature then they will be presented with an unsupported error and the item will be greyed out (figure 21c).



SquashT V2
78:A5:04:85:D2:D9
Unknown Device
D0:4F:7E:21:F5:5E

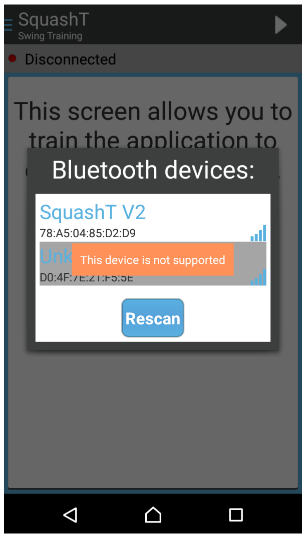
(a) Scanning for devices.



This screen allows you to train the application to

SquashT V2
78:A5:04:85:D2:D9
Unknown Device
D0:4F:7E:21:F5:5E

(b) Rescan for devices.



This screen allows you to train the application to

SquashT V2
78:A5:04:85:D2:D9
This device is not supported
D0:4F:7E:21:F5:5E

(c) Unsupported device.

Figure 21: Arduino Bluetooth connection dialog.

5.1.3.1 Navigation Hierarchy

The following figure shows the navigational structure between screens in the application (figure 22). As you can see, the profile selection screen can have two possible outcomes depending on the action chosen. If a user creates a new profile, they are taken to the initial training screen or if they select an existing profile they are taken to the home screen (default navigation screen) via the navigation panel . The figure below demonstrates the creation of screens (root to node), helping show the screen flow.

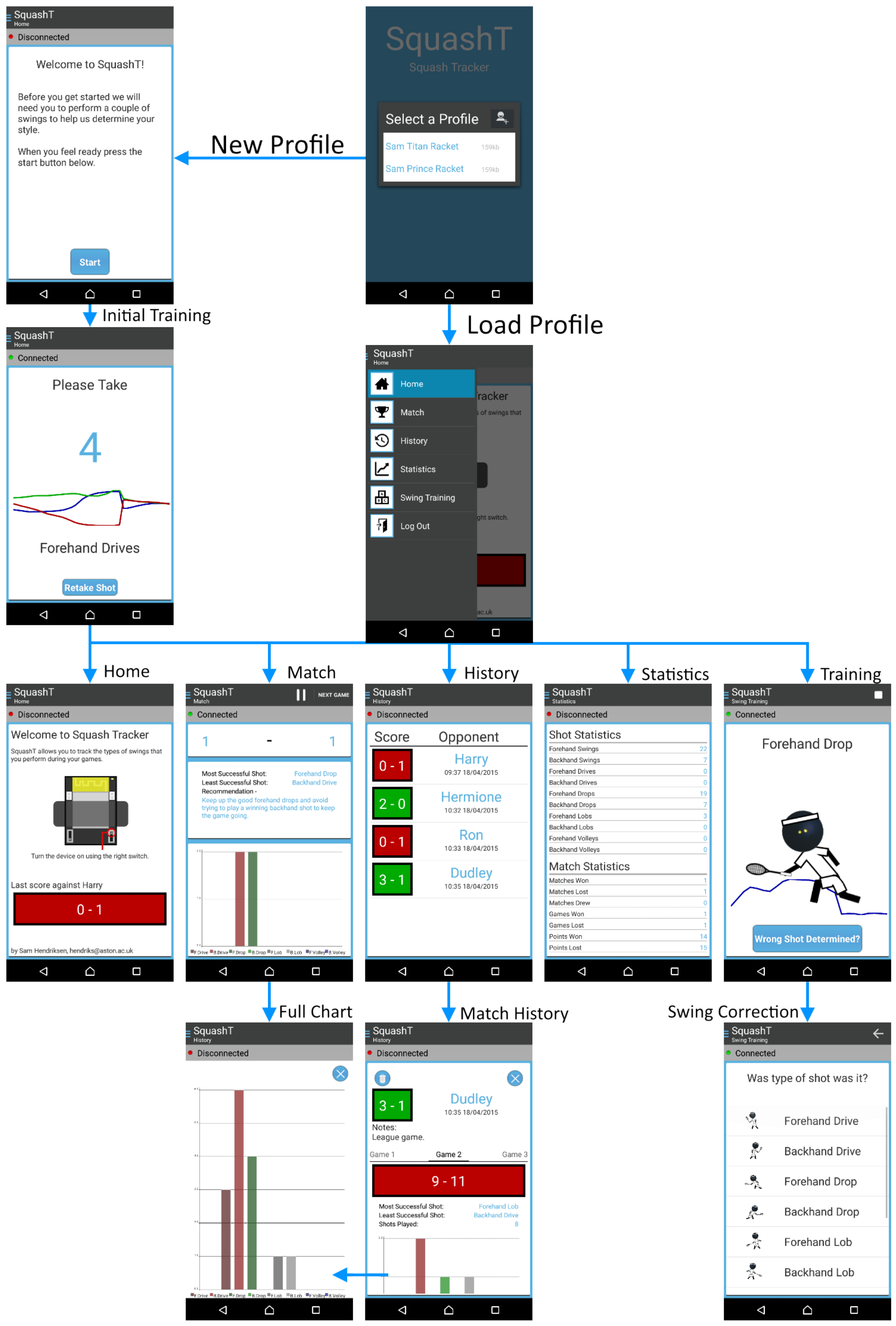


Figure 22: Screen navigation hierarchy.

5.1.4 Bluetooth Data Processing

The following section explains how swings are packaged and decoded after transmission. From a basic overview, the Android application breaks processing down into three key tasks with the data flow shown in the below figure.

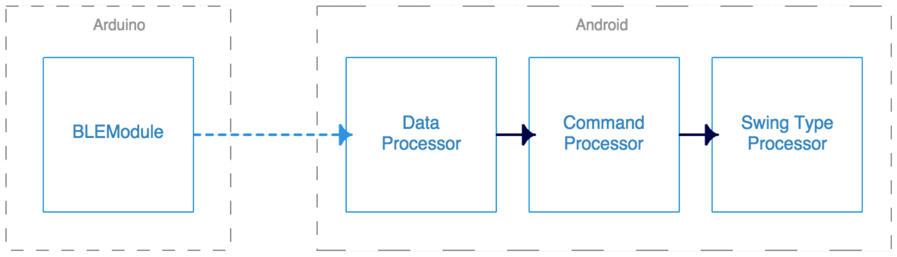


Figure 23: Bluetooth data flow diagram.

The “DataProcessor” parcels the raw packet information into swing objects based on the start and end swing commands received. The processor uses a shared command structure for communications to parse the string input into valid commands and data values. Each value extracted from a packet is either discarded - if the structure is invalid due to poor transmission, or sent for further processing - if the structure is valid. If the structure is valid and found to be a command then it is stored in a command array for the command processor to handle; if it is found to be swing data then it is assigned to a swing object. The full communications command structure shared by the Arduino and Android application are as follows:

Command	Name	Meaning	Usage
$	START_COMMAND	Starts each packet.	-
;	END_COMMAND	Ends each packet.	-
,	VALUE_DELIMITER	Separates each value in a packet.	$value,value;

Table 1: Command structure.

Command	Name	Meaning	Usage
!	END_SWING	Alerts about the detection of an ended swing.	$!;
>	START_ACK	Confirms the start command has been received.	$>;
<	STOP_ACK	Confirms the stop command has been received.	$<;
|	PAUSE_ACK	Confirms the pause command has been received.	$|;

Table 2: Commands sent by the Arduino.

Command	Name	Meaning	Usage
+	COMMAND_START	Activates the swing broadcasting session.	$+;
-	COMMAND_STOP	Disables the swing broadcasting session.	$-;
*	COMMAND_PAUSE	Pauses the current broadcasting session.	$*;
!	END_SWING_ACK	Acknowledges the end swing command.	$!;

Table 3: Commands sent by the Android application.

The second task, “CommandProcessor”, handles the state of the connection based on the commands sent or received. This includes starting and stopping the data and broadcasting any complete swings to the listening activity. The command processor processes each command in the array filled by the data processor, when it finds data it determines whether it should be broadcast or discarded depending on the application’s state e.g. if the application is paused then any swing data will be discarded. The command processor is also responsible for managing the data processing thread by stopping and starting it when required.

The final task, “SwingTypeProcessor”, takes a complete swing object and determines the type of shot that was played. The processor uses the Weka library (GitHub, 2011) to instantiate a classifier and determines the type of swing by analysing the instance data. The predicted class type is then sent to the listening class e.g. the swing training fragment. The below sequence diagram shows a new session that communicates one swing and then ends the session using the command structure.

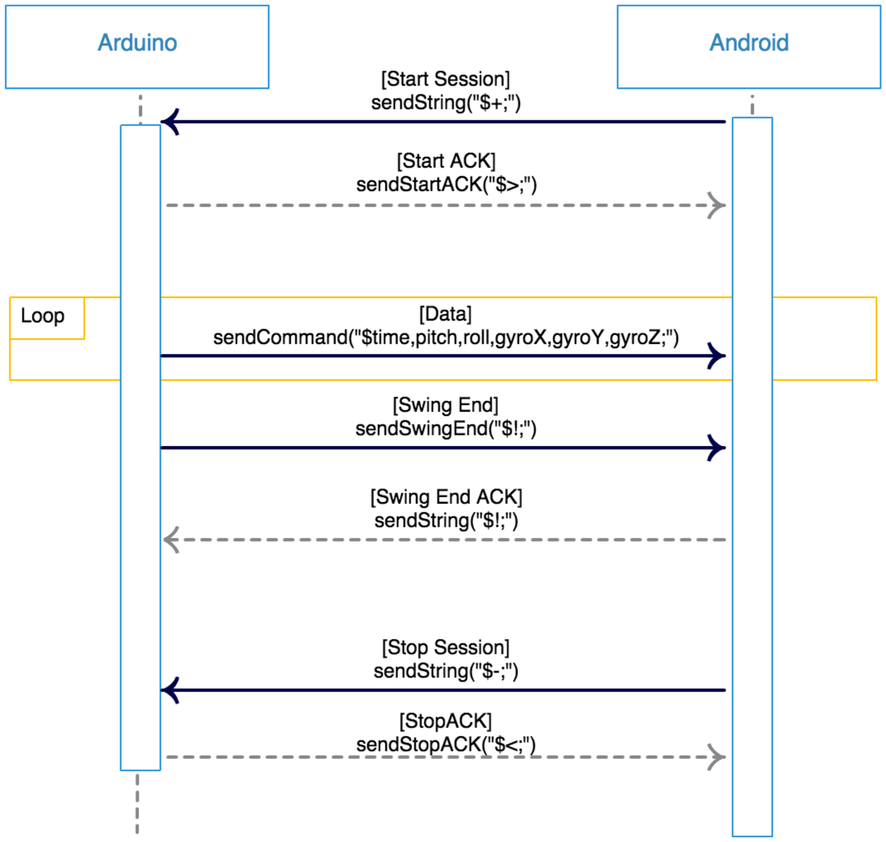


Figure 24: Sequence diagram showing one swing being sent in a single session.

5.2 Arduino Device

The Arduino device was built using multiple components. The main requirements when building this device were to keep it small - so that it fits on a racket, low powered - so that it will last the full length of a game and finally low cost (£25-£30) - so that it can be used by casual players.

5.2.1 Development Environment

The Arduino device was developed using the Arduino IDE (Arduino.cc, 2015c) (1.0.6), which includes all of the required libraries and examples to develop the basic functions and initiate communications. The only external library being used is “i2cdevlib” (GitHub, 2015b), which is needed to help initialise, configure and receive data from the accelerometer and gyroscope of the MPU6050.

5.2.2 Software Architecture

The Arduino source code consists of a “Main” class and two hardware wrapper classes, which expose the functions of the hardware devices. The main class controls these hardware devices by calling the functions when instructed to by the Android application. The “BLEModule” class exposes a function to process incoming serial data and a function to send data to the paired Bluetooth device. The “MPU” class extracts only the required functions from the “MPU6050” helper library, which removes the complexity of the low level library commands. Using these two libraries the device is able to read the MPU data and broadcast it over Bluetooth when it is instructed to do so.

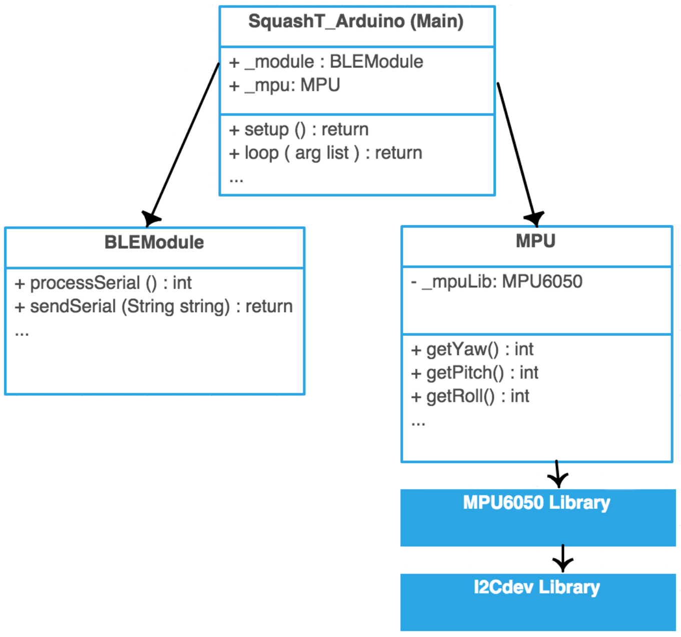


Figure 25: Arduino Hierarchy.

5.2.3 Device Evolution

During the development of the project, multiple devices were created to help implement different areas of functionality. Each device is explained below with the reasoning behind the new version and what it achieves.

Prototype

The first device that was created for the project was the initial prototype (figure 26), which ensured all of the components could communicate successfully. This prototype allowed the basic communications code to be written and tested to see if the Arduino was capable of communicating with Android. The device itself was not capable of swing testing but did allow for basic gyroscope values to be returned.

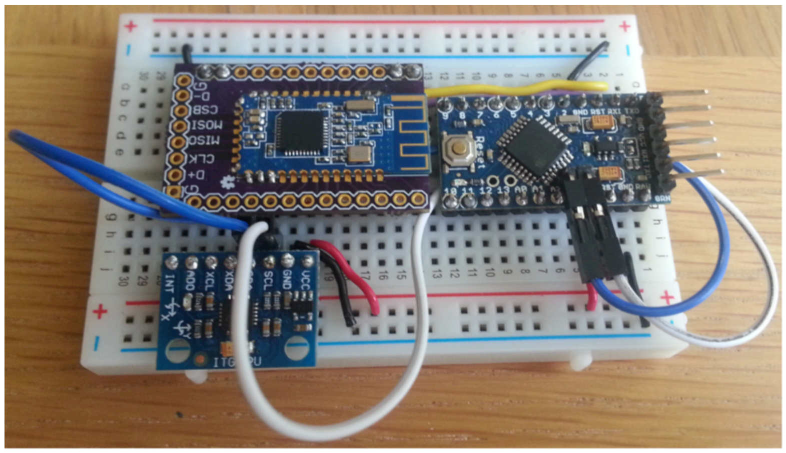


Figure 26: Prototype

Mark 1

The Mark 1 device was developed after the initial communications code was complete and the hardware was finalised (figure 27). The need for this device was to test any basic gestures made using the device without the worry of components falling off as swings were made. The device consists of a prototyping board, which houses the components and solders them as a more permanent solution. The board also leaves the components exposed so that any later additions or alterations could be made easily without having to assemble a new device.

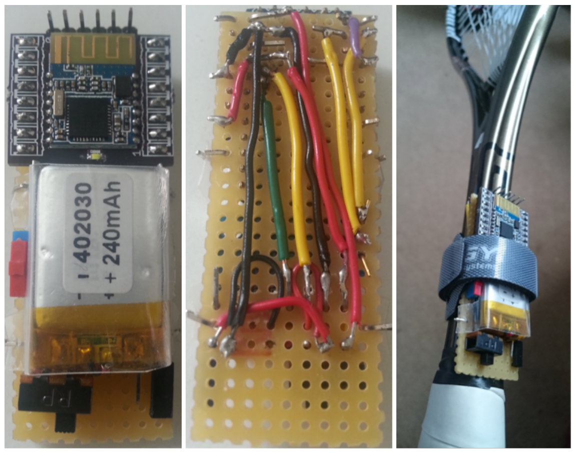


Figure 27: Mark 1

Mark 2

The Mark 2 device, the final device, was developed to be more robust and useable for the demo. It was created to resolve the issues present with Mark 1, where the wires on the back of the board came loose and cracked from wear against the racket. Mark 2 remedies this by having printed wires on the board, hence there are no wires to break free, which also reduces the size, weight and cost of the device. To ensure that the device does not break upon impact and to increase the device's grip on the racket a mouldable rubber called 'Sugru' (sugru.com, 2015) has been applied to the final device, which covers all of the components (figure 28). Theoretically with the components being used on the final device, the battery when idling could last up to 6 hours and when the device is transmitting the battery could last up to 4 hours. (Rated from a 100mAh LiPo battery).

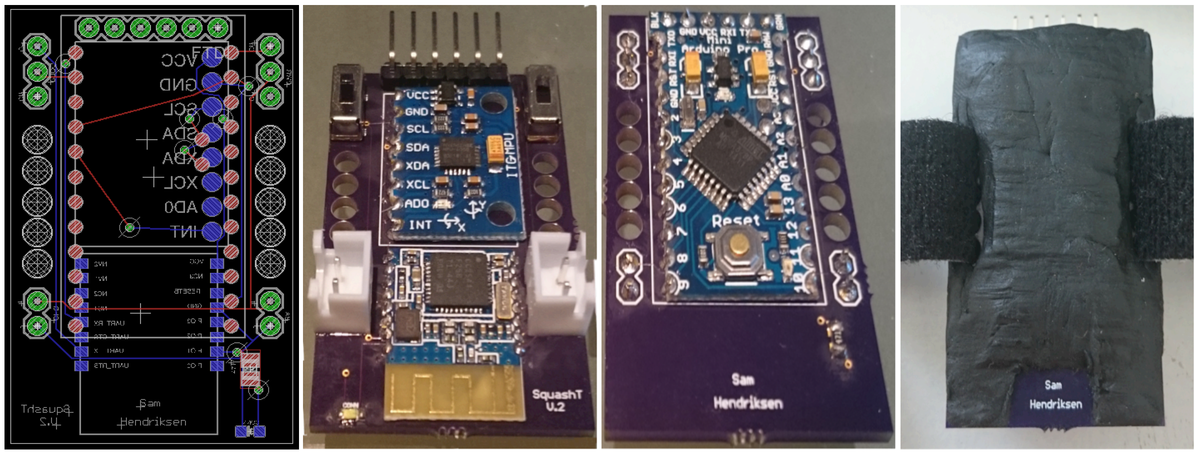


Figure 28: Mark 2

5.2.4 Final Parts List

This section describes the components required to build the final Mark 2 device, with a final cost of £16.

Arduino Pro Mini - £1.99

The first and most important item of the device is the Arduino (Arduino.cc, 2015a), which is from the pro mini series (Arduino.cc, 2015b). This device was chosen for its small size and low power usage with the added benefit that it allows for "Serial" and "Wire" communications. This means that the device is able to connect to the Bluetooth module over a Serial connection and the MPU over the Wire transfer line.

MPU6050 - £1.60

The module chosen to measure the acceleration of the device is an MPU 6050 (Playground.arduino.cc, 2015). The module has been chosen due to its ability to record measurements up to 16g, a sensitivity of up to 2000°/s and a very low cost per module. The module itself is very well documented and has an active community, which makes it easier to understand.

Bluetooth Low Energy Module, HM11 - £4.60

The communications protocol being used on the device is a Bluetooth 4.0. Bluetooth 4.0 use a very small amount of power, which is ideal as the device will be running off a battery. The module selected for the final device is a HM-11 module due to its very small size and power consumption; the specific power consumption being: Transmit current - 15mA, Receive current - 8.5mA, Deep sleep current - 600uA (Rlx.sk, 2015).

Other

To combine and make use of the above parts a further set of a components were required, these are briefly explained below.

- 100mAh LiPo battery, used to power the components (£2.45).
- 0603 White SMD LED, to show the connection status (0.8p).
- 47Ohm Resistor (10p).
- Socket header (5p).
- Printed circuit board (£2)
- 2 x Switches (2p).
- 2 x JST Connector, to connect the battery and charger (60p).
- Sugru, to protect the components (£1.50).

5.2.5 Communicating with the HM-11 Module

Setting up the HM modules requires a series of "AT Commands" to be sent over the serial connection before Bluetooth can be initialised. Commands such as baud rate, slave mode and broadcast name are programmed on new devices when they are detected; all subsequent times, a shortened and quicker setup is used which simply initialises the module as soon as it is ready. After the module has been initialised it is possible to send data over the 'Serial' connection, which is received on the paired device.

5.2.6 Bluetooth Low Energy Reliability Problems

During the development of the Arduino device there were multiple setbacks due to the reliability of Bluetooth 4.0, which were not apparent during the initial feasibility tests. The main problem was the irregular data loss and corruption when transmitting to the Android device. To understand why this was happening multiple scenarios were devised and run to see what was causing the data corruption. Each of the following scenarios use the same packet structure that helps Android parse the data. Each packet starts with "$", has values separated by ",", and ends with ";". The first scenario was to send a static stream of data to the Android application and keep track of the successful and unsuccessful packets. The lower baud rate of “9600” was chosen to ensure that the transmission of data would be more reliable. The following packet “$0,1,2,3,4,5;” was sent to the Android application at a continuous rate which as you can see from the following image (figure 29) was very successful in the sense that there was no data corruption.

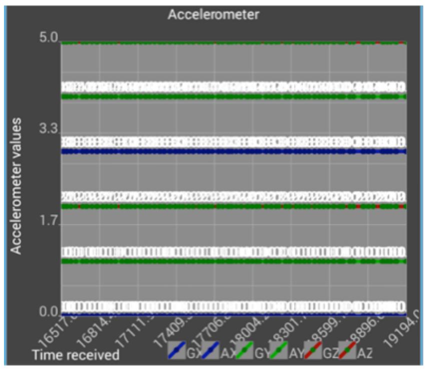


Figure 29

The next scenario was to try and send a higher precision number rather than single digit integers. For this test I sent the same values but as doubles. This meant that each single digit integer in the packet was now 3 characters long (“$0.0,1.0,2.0,3.0,4.0,5.0;”). In this test I found that the data was now very unreliable, the variables in the packets were either swapping or missing (figure 30).

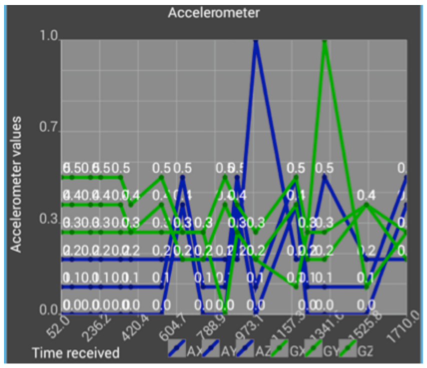


Figure 30

To understand if this was due to the Bluetooth module, Arduino device or Android code, another communications method was used. For this test USB communications were implemented connecting the Arduino directly to the Android device through the use of an FTDI serial cable. The test was run again using the same baud rate and proved 100% reliable. This meant that there was a problem with the Bluetooth module. Whilst the communications were reliable, real MPU values were sent to test that the MPU was performing as expected before moving back to fix the Bluetooth problems. During the tests there were spiked values being returned by the MPU which were easily fixed using a ‘Median Filter’.

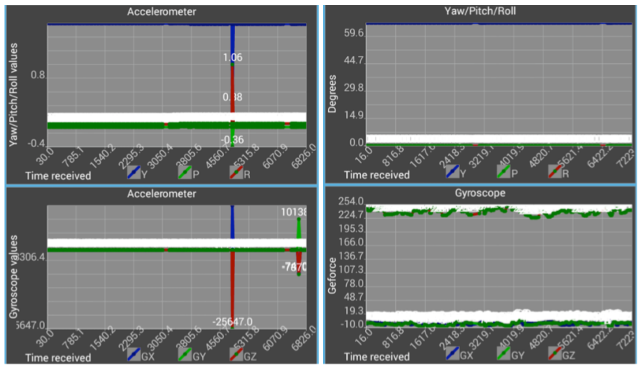
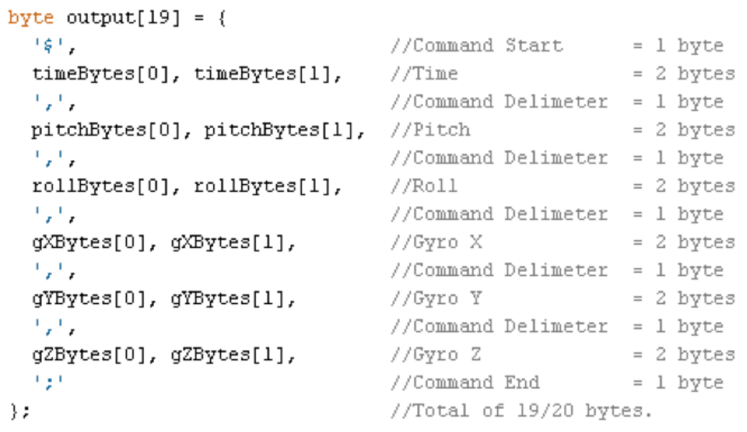


Figure 31: MPU spikes (left), Median filter (right)

Now that the data was reliable using a serial connection, Bluetooth needed to be debugged to understand why it was slicing/losing packets. To understand this the results from the initial two scenarios were compared with the only significant difference being the length of data being sent. This provided a final test scenario which involved sending varying lengths of data. The first successful packet, which was tested in the previous scenario, consisted of 13 bytes of data ("$0,1,2,3,4,5;"). This was used as the starting point for the test with a delay of 10 milliseconds between messages to increase the reliability to 100%. Now that the data was reliable the next test was to increase the data length byte by byte until the data became unreliable. After multiple revisions with varying lengths the data eventually and became severely unreliable at 21 bytes of data, with only 10% of the data passing the parsing on Android. Once one byte was removed and 20 byte packets were being sent, nearly 100% accuracy was achieved again. To understand why there was such a severe difference between 20 and 21 bytes of data, research was undertaken into Bluetooth 4.0 documentation. Eventually a useful article explained that the maximum data and rate each packet can be sent is 20 bytes every 7.5 milliseconds (Gomez, Oller and Paradells, 2012). The challenge now was to fit all of the accelerometer data into 20 byte packets or less.

After researching the best way to represent the values the data type 'short' was found that has a range of '−32767 - +32767'. This range is superfluous of what is needed but only translates to two bytes per a 'short' data value. This means that if all of the items are stored as 'shorts' then they will only take up a total of 12 bytes of information. Once the data was wrapped in the parsing characters the total bytes added up to 19 bytes of data, one short of the limit. See figure 32 for the final Arduino packet code.



```
byte output[19] = {
  '$',                              //Command Start      = 1 byte
  timeBytes[0], timeBytes[1],       //Time               = 2 bytes
  ',',                              //Command Delimeter  = 1 byte
  pitchBytes[0], pitchBytes[1],     //Pitch              = 2 bytes
  ',',                              //Command Delimeter  = 1 byte
  rollBytes[0], rollBytes[1],       //Roll               = 2 bytes
  ',',                              //Command Delimeter  = 1 byte
  gXBytes[0], gXBytes[1],           //Gyro X             = 2 bytes
  ',',                              //Command Delimeter  = 1 byte
  gYBytes[0], gYBytes[1],           //Gyro Y             = 2 bytes
  ',',                              //Command Delimeter  = 1 byte
  gZBytes[0], gZBytes[1],           //Gyro Z             = 2 bytes
  ';'                               //Command End        = 1 byte
};                                  //Total of 19/20 bytes.
```

Figure 32: Packet Structure

Sending the data using the new structure showed that it was now reliable enough to use with just the odd occurrence of 5% data loss, which is handled on Android. The throughput was also increased by raising the baud-rate from "9600' to "38400", which now showed the data was reliable enough for swing analysis.

5.2.7 Broadcasting Swings

To ensure power conservation of the device, so that it lasts the length of a game, special broadcasting methods are used where swings are only broadcast from the Arduino when they pass a specific criteria. Swings are broadcast when at least one axis surpasses 4g-force, which causes a continuous stream of data to be broadcast until all axis drop below 4g-force for greater than 250 milliseconds. The contingency period of 250 milliseconds has been chosen to ensure that the data has a chance to swap into negative g-force once a ball is impacted rather than stopping as soon as the axis starts to dip. Once the g-force has dipped for 250ms an end swing byte is sent to inform the Android device that a full swing has been taken and can be completed. By sending data in chunks, less processing power is needed on the Android device as it receives the important swing information, rather than having to parse a continuous stream of data. The battery of the Arduino device is able to last for two more hours when idling as it uses 8mA, compared to the 15mA when transmitting.

5.3 Data Mining

5.3.1 Algorithm Selection

To ensure that the correct swings are detected multiple factors need to be considered. The accuracy of the application breaks down into two key components, the algorithm and the features.

To understand which algorithm was best for the application three requirements were tested; the accuracy, the initial load time (build time) and the time it takes for the algorithm to classify a swing type. To test the accuracy, each algorithm was given the same training and test set of data to determine how closely they could match the data. The table below breaks down the most promising algorithms based the three requirements, where each value was calculated based on the average of five runs.

Algorithm	Accuracy	Build Time	Classification Time
SMO	98%	7.424 seconds	15.4ms
IBK	98%	2.918seconds	329ms
FT	96%	30.947 seconds	14.6ms
Dagging	97.6%	39.072 seconds	75.6ms

Table 4: Algorithms rated by their abilities.

The above table shows that during the testing exercises the algorithm types “IBK” (Aha and Kibler, 1991) and “SMO” (Platt, 1998) both proved to be very accurate and fast at either booting or calculating. “SMO” was chosen because, in my opinion, it is more desirable to make users wait longer to open their profile once than each time they take a swing.

5.3.2 Feature Selection

To understand which features would help correctly identify the types of swings performed, multiple research papers were explored that looked at activity detection (Bayat, Pomplun and Tran, 2014; Catal et al., 2015; Maguire and Frisby, 2009; Kwapisz, Weiss and Moore, 2011). From each paper a consistent set of features are present when detecting the correct activities; these features are “Mean”, “Standard Deviation”, “FastFourier Transform”(FFT), “Correlation”, “Magnitude”, “Skewness” and “Kurtosis”. Once applied to the pitch, roll and gyroscope values the following 24 features were defined:

- Raw[4]: raw values for the time stamp, pitch and roll.
- Mean[5]: the mean for pitch, roll, gyro x, gyro y and gyro z.
- Standard Deviation[5]: calculated for pitch, roll, gyro x, gyro y and gyro z.
- FFT[5]: FastFourier Transform calculations for pitch, roll, gyro x, gyro y, gyro z.

- Correlation[1]: calculated using pitch and roll.
- Magnitude[1]: calculated using all axis of the gyroscope.
- Skewness[2]: calculated for pitch and roll.
- Kurtosis[2]: calculated for pitch and roll.

Using the potential features, the next step was to find out which features were useful in helping to detect swings and which features were redundant or confusing the classifier. The Weka Explorer (Cs.waikato.ac.nz, 2015b) was used to understand and evaluate each individual feature and determine its usefulness. As can be seen in figure 33, the FFT values are too closely related and do not leave many unique features for the classifier to help determine the correct swing types.

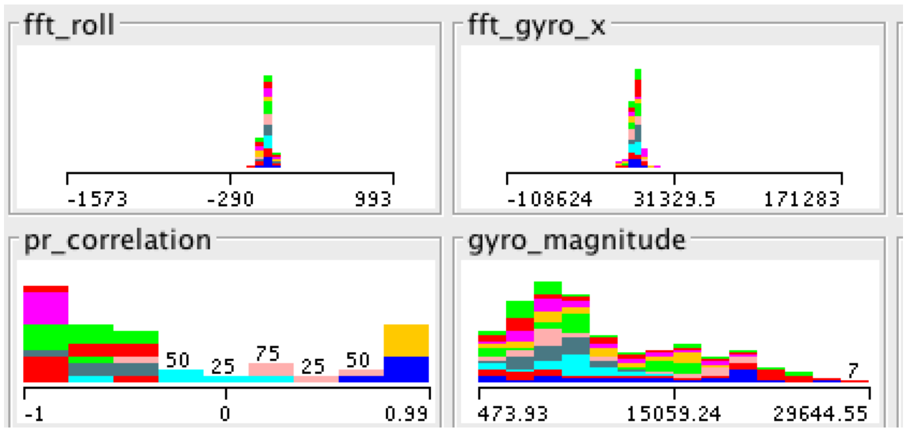


Figure 33: Weka Explorer showing the distinction between swing types (each colour being a type).

Using the results from the Weka Explorer, the table below (table 5) shows the effects on the accuracy when the redundant features were removed. By removing these features the accuracy of the application was increased by roughly 5% and the boot time of the application was reduced by 1480ms, which is because less features were being trained on the classifier.

Feature	Accuracy Change Once Removed
Pitch FFT	0%
Roll FFT	0%
Gyro X FFT	0%
Gyro Y FFT	0%
Gyro Z FFT	0%
Raw time series	+5%
Roll kurtosis	0%
Pitch Skewness	0%

Table 5: Accuracy when removing redundant swing features.

5.3.3 Final Swing Type Nominal Classes

Originally the plan was to detect four forehand and four backhand swing types; “drives”, “volleys”, “drops” and “lobs”. After the algorithm proved to be very accurate at detecting these eight swing types an additional two new swing types were added, which aid the scoring and shot recommendations. The new swing types are “shot won” and “shot lost”, which allow the application to keep track of the score and record a players successful and unsuccessful shots. This is done by marking the shot type before a point is won as a winning type and the shot type before a point is lost as a losing type. By maintaining a record of the successful and unsuccessful shot types, users can understand which shots they need to improve during training exercises or avoid in a game.

5.4 Product Testing and Quality Assurance

During the development of the SquashT project, multiple testing techniques were deployed to ensure that the software still functions as defined. During each revision a set of automated tests were run, which checked that the core functionality of the application still functioned and had not been broken by the latest changes.

To help test the Arduino components, the Arduino IDE code "Examples" (Arduino.cc, 2015d) were used, which helped test that each individual component worked and was able to communicate. Because the Arduino device is not debuggable it makes it harder to test the actual software, which means that the testing has to be done from the point it is received by the Android application.

To ensure that the Android application functions as required, a testing strategy was devised (see appendices 10.2). This helped test the four key quadrants of a project, as defined by Crispen and Gregory (2009). The testing strategy explains the testing techniques chosen and describes the tests performed and their outcomes.

To test the first quadrant of the project, Android automated test cases were written to test that the full process of swing creation and swing type prediction worked without error (see appendices 10.3.1). The tests start by creating two new swings objects that represent a forehand and backhand drive, which are then sampled and analysed to determine the correct features. Using these two swing objects, the data-mining algorithm is then trained to recognise these two types of swings before it is given the same two swings to classify. The test case ensures that the swings are correctly sampled, the features are chosen and finally that the classifier can differentiate two types of swings.

The second quadrant tests use robust boundary value analysis to check that swings are correctly sampled (see appendices 10.3.2). The tests were written to check that the sampled swing data returns the correct length of packets and that any errors are caught. Because these test cases are automatic they can be run at each development revision to understand if the core functionality of the application still works, which is very useful as a completing test before each commit.

As part of the third quadrant, exploratory testing was chosen which looks into the "user profile" (figure 13) functionality. A test plan was written (see appendices 10.4), which tests that the creation, deletion and loading of user profiles work as expected from a users perspective. This includes handling any errors, such as corrupt data, and displaying appropriate messages. This area was chosen for exploratory testing as it is one of the key items of functionality when it comes to usability. Without a successful user profile page users may not be able to access the application or load their previous data.

To check that the current data-mining algorithm used by SquashT is the most appropriate, performance tests were produced as part of the final quadrant tests. These tests check each algorithms performance based on accuracy, build time and classification time (see appendices 10.5). The algorithms are rated based on their merit from each testing category which helps conclude that "SMO" is most appropriate for SquashT.

6 Evaluation

The main goal for this project was to prove the feasibility of tracking a Squash player's game using a custom created device; which is cheaper, smaller and as efficient as its competitors. Now that the development has been completed, I feel the end product has proven that the methods and technologies used are feasible and very capable of breaking down a player's activities, meeting the initial goal. The device itself has surpassed the initial requirements with the end cost coming in at £16, which is 1/6 of the price of existing applications (see Background) and 1/3 of the price that a majority of casual users are willing to pay (£25-£30) (see appendices 10.6). The device is also small enough to be strapped to a user's racket and capable of providing the data needed for gameplay analytics. The data mining techniques have also proven to be more accurate than originally expected providing detections for more swing types than originally planned, which has allowed the application to be extended further providing a win and loss gesture.

Although the end product proves the project is feasible, the main limiting usability problem is the detection of false positive swings. Currently when a player brings their racket up too fast in preparation for a swing the application will detect this as a swing, which in theory is correct. This makes it hard to determine how to cancel out these false "pre-swings" as they could potentially be valid swings. However, when the application classifies a valid swing it is 98% accurate (as determined by Weka) the vast majority of the time, which means that once the false detections are corrected the application will be very usable and reliable. For the end product to be more reliable further sensors will need to be incorporated into the device to understand when an impact has been made with the ball. By implementing this check the false positive swings can be eradicated by only retaining those that have made an impact with the ball.

The main risk of this project has been using multiple technologies that I had no previous experience with. By incorporating Weka and Bluetooth low energy, I have had to spend a considerable amount of time understanding what these technologies can and cannot do. I feel that this has been useful for the project as the right technologies have been chosen but it has also made it harder to get the project closer to being usable on a Squash court. However, it has been very interesting learning about these technologies and they are definitely areas that I will look into incorporating into future projects now that I have the experience.

The most advantageous technology being used in this project is Weka. I feel that without this library the method of manually determining swings would be a near impossible task, especially to the degree of accuracy that Weka has provided. The ability to visualise the data using a desktop application made the process of deciding and testing algorithms and features a very structured and clear task. For this reason I highly recommend Weka because of its simple and incredibly useful set of features.

One of the most challenging aspects of this project has been debugging the Bluetooth communications between the Arduino and Android device. The problem stemmed from the Arduino device being unable to send a large amount of packet data to the receiving Android device. The main reason why this problem was hard to narrow down was because it only occurred at seemingly random points in time. As the accelerometer was being turned, the values increased from 0-360 degrees which meant more data was being taken up in each packet than when the device was flat on the table. Understanding that packets greater than 20 bytes became scrambled, each variable in a packet was reduced to a 'short' data type, which helped keep each packet a fixed length of 19 bytes. With this limitation understood, the Bluetooth communications became reliable enough for the project to proceed. This area took a lot of time to understand but I feel that Bluetooth is the only viable communications method that can ensure the project's success, mainly due to the fact that the power usage on the Arduino is minimal and most smartphones come equipped with a Bluetooth module.

As part of the initial background research I looked into the feasibility of Bluetooth as a communications method through some basic testing. The problem with the tests was that they did not pick up on the transmission problems described above. I feel that with a more thorough set of tests, the problems with the Bluetooth modules would have picked up before the development started, which may have persuaded my decision towards another form of communications. This could have freed up more time, which would have been spent on the resolution of false positives swings rather than debugging the communications.

As a whole I feel that the final application has more potential than originally planned. The application is capable of recording a user's activities and able to recognise when the same activity is performed again. If a more flexible user interface was implemented, the project could target activity detection rather than just Squash. This would allow users to apply the device to a sport or activity of their choice, which would greatly increase the user base and capabilities of the product. One idea would be to allow users to record gestures and enter actions for the application to perform when it sees those gestures again. This would allow each user to tailor the application to their unique needs, rather than having to use it a set way.

7 Conclusion

At the start of this project I aimed to create a device that was not only capable of recording a player's game but also of a low enough cost for a large number of casual players to see the benefits. I feel that the project has proven that it is definitely possible and feasible to record and analyse a player's game using the techniques defined and that the end cost of the product has been achieved by iteratively developing the device. Once mass produced the device could not only be making a profit but also opening up the option for casual players to perform analytics on their games as a cheaper alternative to coaching.

During the development of the project multiple setbacks and problems have been faced with the Bluetooth communications. Although it has taken time to learn and understand the limitations of the technology, I feel that now that I have overcome the problems, Bluetooth is a very viable and usable technology. The major benefit of Bluetooth compared to other RF devices is that it is highly supported in today's technology and is included in nearly all smartphones, which makes the end product accessible to more users.

Although the project has shown that it is possible to record a player's swings efficiently, the project still requires more work towards understanding how to remove the false positive detections generated by pre-swings. More research and recordings need to be made with these pre-swings to help identify a pattern that helps remove and reduce the amount of false detections made, which could be done by only keeping swings that have had an impact with the ball. Once these pre-swings have been eradicated, the next approach would be to enable the device to be usable on court, which would require solving the next problem of removing a player's motion as they move around the court so that it is not detected as a swing. A possible solution would be to add additional sensors to the player, which would record their movement speed to help cancel out the movement speed detected by the racket. The new device would open up the ability to track more about a player's game such as their average movement speed, calories and heart rate. This would extend the SquashT application to record and analyse the physical efforts that players perform during certain games and to keep track of their current fitness. Unfortunately the mobile device used for the application cannot be used to record a player's movement speed, as it does not offer the precision needed for the acceleration produced in Squash.

In summary I feel that the project has been both technically challenging and rewarding. I have learnt a lot about useful technologies and hardware that I had no prior experience with and I feel that from developing this project I can successfully inform others of the advantages and disadvantages of the chosen technologies.

8 References

Alamar, B. (2013). *Sports analytics*. New York: Columbia University Press.

Android, (2015). Bluetooth Low Energy | Android Developers. [online] Developer.android.com. Available at: https://developer.android.com/guide/topics/connectivity/bluetooth-le.html [Accessed 18 Mar. 2015].

AndroidPlot. (2015). [online] Available at: http://androidplot.com/ [Accessed 22 Mar. 2015].

App Store, (2014). Tennis Trakker Pro. [online] Available at: https://itunes.apple.com/gb/app/tennis-trakker-pro/id302285722?mt=8 [Accessed 3 Dec. 2014].

Arduino.cc, (2015a). Arduino - Home. [online] Available at: http://www.arduino.cc/ [Accessed 28 Feb. 2015].

Arduino.cc, (2015b). Arduino - ArduinoBoardProMini. [online] Available at: http://arduino.cc/en/Main/ArduinoBoardProMini [Accessed 28 Feb. 2015].

Arduino.cc, (2015c). Arduino - Software. [online] Available at: http://arduino.cc/en/main/software [Accessed 22 Mar. 2015].

Arduino.cc, (2015d). Arduino - Learn the basics. [online] Available at: http://www.arduino.cc/en/Tutorial/HomePage [Accessed 16 Apr. 2015].

Bayat, A., Pomplun, M. and Tran, D. (2014). "A Study on Human Activity Recognition Using Accelerometer Data from Smartphones". *Procedia Computer Science*. Volume 34, Pp. 450–457

Cadsoftusa.com, (2015). CadSoft EAGLE PCB Design Software|Support, Tutorials, Shop. [online] Available at: http://www.cadsoftusa.com/ [Accessed 28 Feb. 2015].

Catal, C., Tufekci, S., Pirmit, E. and Kocabag, G. (2015). "On the use of ensemble of classifiers for accelerometer-based activity recognition". *Applied Soft Computing*.

Cleland, I., Kikhia, B., Nugent, C., Boytsov, A., Hallberg, J., Synnes, K., McClean, S. and Finlay, D. (2013). "Optimal Placement of Accelerometers for the Detection of Everyday Activities", *Sensors 2013, 13(7)*, Pp. 9183-9200

Commons.apache.org, (2015). Apache Commons Math. [online] Available at: https://commons.apache.org/proper/commons-math/ [Accessed 22 Mar. 2015].

Cs.waikato.ac.nz, (2015a). Weka 3 - Data Mining with Open Source Machine Learning Software in Java. [online] Available at: http://www.cs.waikato.ac.nz/ml/weka/ [Accessed 21 Jan. 2015].

Cs.waikato.ac.nz, (2015b). Weka 3 - Data Mining with Open Source Machine Learning Software in Java. [online] Available at: http://www.cs.waikato.ac.nz/~ml/weka/gui_explorer.html [Accessed 10 Mar. 2015].

D. Aha, D. Kibler (1991). "Instance-based learning algorithms", *Machine Learning*, 6:37-66.

Developer.android.com, (2015a). Activity | Android Developers. [online] Available at: http://developer.android.com/reference/android/app/Activity.html [Accessed 23 Apr. 2015].

Developer.android.com, (2015b). Fragment | Android Developers. [online] Available at: http://developer.android.com/reference/android/app/Fragment.html [Accessed 23 Apr. 2015].

Flaticon.com, (2015). Free vector icons - SVG, PSD, PNG, EPS & Icon Font - Thousands of Free Icons. [online] Available at: http://www.flaticon.com/ [Accessed 18 Apr. 2015].

GitHub, (2011). rjmarsan/Weka-for-Android. [online] Available at: https://github.com/rjmarsan/Weka-for-Android [Accessed 18 Mar. 2015].

GitHub, (2015a). Build software better, together. [online] Available at: https://github.com/ [Accessed 2 Apr. 2015].

GitHub, (2015b). jrowberg/i2cdevlib. [online] Available at: https://github.com/jrowberg/i2cdevlib [Accessed 22 Mar. 2015].

Gomez, C., Oller, J. and Paradells, J. (2012). Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. Sensors, 12(12), pp.11734-11753.

Jnhuamao.cn, (2015). Huamao technology Co,. Ltd.. [online] Available at: http://www.jnhuamao.cn/bluetooth.asp?ID=1 [Accessed 28 Feb. 2015].

JNHuaMao Technology Company, (2014). Datasheet for Bluetooth 4.0 BLE module. [pdf] Available at: http://www.huamaosoft.com/bluetooth40_en.zip [Accessed 1 Jan. 2015]

J. Platt (1998). "Fast Training of Support Vector Machines using Sequential Minimal Optimization", *Advances in Kernel Methods - Support Vector Learning*

Justinmind.com, (2015). Justinmind: Interactive wireframes for web and mobile. [online] Available at: http://www.justinmind.com/ [Accessed 18 Mar. 2015].

Kwapisz, J., Weiss, G. and Moore, S. (2011). "Activity recognition using cell phone accelerometers", *ACM SIGKDD Explorations Newsletter archive, Vol. 12 , No 2,* Pp.74-82

Letsmakerobots.com, (2015). Bluetooth 4.0 for Arduino | Let's Make Robots!. [online] Available at: http://letsmakerobots.com/node/38009 [Accessed 28 Feb. 2015].

Maguire, D., Frisby, R. (2009) "Comparison of feature classification algorithm for activity recognition based on accelerometer and heart rate data", *Dublin Institute of Technology, IT&T Conference*

Oshpark.com, (2015). OSH Park ~ Welcome. [online] Available at: https://oshpark.com/ [Accessed 28 Feb. 2015].

Phillips, B. and Hardy, B. (2013). *Android programming: The Big Nerd Ranch Guide,* The Big Nerd Ranch, Inc.: United States of America

Playground.arduino.cc, (2015). Arduino Playground - MPU-6050. [online] Available at: http://playground.arduino.cc/Main/MPU-6050 [Accessed 28 Feb. 2015].

Rlx.sk, (2015). Bluetooth V4.0 HM-11 BLE Module (Seeed 210005001) | RLX COMPONENTS s.r.o. Electronic Components Distributor. [online] Available at: http://www.rlx.sk/en/bluetooth/2781-bluetooth-v40-hm-11-ble-module-seeed-210005001.html [Accessed 28 Feb. 2015].

Smarttennissensor.sony.net, (2014). Sony | Smart Tennis Sensor. [online] Available at: http://www.smarttennissensor.sony.net/NA/ [Accessed 21 Jan. 2015].

Sugru.com, (2015). FIX THAT THING. [online] Available at: http://sugru.com/ [Accessed 1 Mar. 2015].

Talabi, A. (2006). Optimal Strategies for the game of Squash. *Journal of Education*, Vol, 25, p.79.

Tidwell, J. (2005), *Designing Interfaces*, O'Reilly Media Inc.: United States of America.

Weka.sourceforge.net, (2015). SMO. [online] Available at:
http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/SMO.html [Accessed 25 Mar. 2015].

Weka.wikispaces.com, (2015a). weka - home. [online] Available at: http://weka.wikispaces.com/ [Accessed 18 Mar. 2015].

Weka.wikispaces.com, (2015b). weka - Programmatic Use. [online] Available at: ht-tps://weka.wikispaces.com/Programmatic+Use [Accessed 18 Mar. 2015].

Weka.wikispaces.com, (2015c). weka - Creating an ARFF file. [online] Available at: http://weka.wikispaces.com/Creating+an+ARFF+file [Accessed 18 Mar. 2015].

Zepp.com, (2014). Zepp Tennis | Analyze & Improve Your Game. [online] Available at: http://www.zepp.com/tennis [Accessed 3 Dec. 2014].

9 Bibliography

Dario G. Liebermann , Larry Katz , Mike D. Hughes , Roger M. Bartlett , Jim McClements & Ian M. Franks (2002) “Advances in the application of information technology to sport performance”, *Journal of Sports Sciences, 20:10,* 755-769, DOI: 10.1080/026404102320675611

Mike D. Hughes & Roger M. Bartlett (2002) “The use of performance indicators in performance analysis”, *Journal of Sports Sciences, 20:10*, 739-754, DOI: 10.1080/026404102320675602

10 Appendices

10.1 Project Diary

Title	Initial Market Research	01/10/2014
Discussion	To ensure that my project is feasible and original I have started to look at other devices and applications present on the market. This aimed to help me understand how others have implemented their applications and to help me find a unique aspect for my project. During my research I have found different applications that focus on specific analysis areas, some of these areas are of interest to my project. Here are some of the applications found that contain functionality of interest: App Store, (2014). Tennis Trakker Pro. [online] Available at: https://itunes.apple.com/gb/app/tennis-trakker-pro/id302285722?mt=8 [Accessed 3 Dec. 2014]. For the first application, a bystander had to log the shots that a player performed so that after the player was done they could evaluate how they played. I found this to a very interesting idea, although relatively useless for Squash if you did not have a bystander present. This made me think that the process could be very useful if automated. After researching into applications that use devices to record data rather than users, I found an application called Zepp. This application provides swing statistics such as shot speed and ball impact placement. The problem with this application is that it is only for tennis and costs a considerable amount of money (£130). This rules out the average player being able to improve their game. Zepp.com, (2014). Zepp Tennis | Analyze & Improve Your Game. [online] Available at: http://www.zepp.com/tennis [Accessed 3 Dec. 2014].	
Conclusion	It would be very useful to be able to record any shots that you make during a game to help you improve your shot selection. The current devices are quite expensive which means that less players can see the advantages of the technology. I will look into creating a low cost device that can help users record their game.	

Title	Building a Prototype Device	02/10/2014
Discussion	From my research I have come up with a list of parts that are needed to record a player's swings. The parts are: • Arduino Pro Mini (due to its small form factor). • Bluetooth Low Energy (BLE) HM-11 Module (due to its ability to use a low amount of power). • MPU-6050, because it is able to provide up to 16g detection with accelerometer data also. From this list of parts a basic breadboard prototype has been created which can enable communications testing between the Arduino and Android device.	
Conclusion	Prototype built from the researched parts list.	

Title	Survey Performed to Understand the Project's Marketability	05/10/2014
Discussion	I have created a survey to gather information from casual Squash players at Aston University. The survey asks the players if they would be willing to buy a device to help them track and analyse their games. If they are interested, the survey asks what price they would be willing to pay for this range of functionality. The options were "Not Interested", "£10-£15", "£15-£20", "£20-£25", "£25-£30" and "£30+".	
Conclusion	Survey sent to Aston University casual Squash players.	

Title	Project Definition - Started	12/10/2014
Discussion	I have started to implement the first draft of my project definition form. This includes planning out my project using Microsoft's project planner to create a Gantt chart. I have also started to form the key points of my project in the definition form.	
Conclusion	Started a Gantt chart project plan and started to form my key project points.	

Title	Supervisor Meeting - 1	17/10/2014
Discussion	During my first meeting with George I described my current progress and the direction I would like to take with my project. This included describing how I would like the application to log a player's swings for them without needing a bystander, which I found to be the case in my market research. George thought it was a good idea but could potentially be quite hard to create the device, which would provide game analysis. He therefore suggested that I make a fast start and try to prove it can be done so that if it turns out to fail then I will have time to work on a contingency plan. The main concern was over the accuracy of the accelerometer data. George also suggested that if it did fail then I could look at recording a player's game using video and detect swings from the video after the game. This would be a good alternative, although I would like players to be able to review their game at any point and not after the game which is too late.	
Conclusion	Prove the device can be built and capable or real time recording.	

Title	Testing the live data reliability	17/10/2014
Discussion	To prove the feasibility of the prototype device, each component needs to be tested to ensure that it can provide the precision of data needed for swing analysis. The first component that has been tested is the MPU, which uses a library to help me configure it to provide Yaw/Pitch/Roll data. As for the BLE device I have written some basic test code to send the MPU data over Bluetooth, and some basic receiving code on Android to test that it is received. After some testing the data appears to get scrambled slightly in the sense that some packet data is getting merged. I will need to look into this further to try and understand why. Currently the BLE data is running fast enough for swing detection but the fact that it gets scrambled means that it is not a viable option yet.	
Conclusion	The Arduino is capable of sending data fast enough for swing detection although it is getting scrambled for some reason.	

Title	Creating the Mark 1 Device	18/10/2014

Discussion	The Mark 1 prototype device has now been created, which has proved successful with each part communicating to the Arduino successfully. The Mark 1 device has been created to house the components more permanently so that it can be used for testing without the worry of components coming loose.
Conclusion	Mark 1 device working successfully.

Title	Implemented Main Screens – Android	19/10/2014

Discussion	Now that the Mark 1 device has been created I have developed the main application screens. These include 'Home', 'Match', 'History' and 'Swing Training' screens. These screens will give me a basis to work on whilst developing the Arduino device along side. Further work will be done on the individual screens when the Arduino development has stabilised.
Conclusion	Created the basic screens and application structure.

Title	Supervisor Meeting - 2	21/10/2014

Discussion	In this meeting I showed the BLE communications to George and demonstrated how the communications became unreliable when more data was being sent. We both agreed that I need to look into making BLE reliable and if all else fails to use an alternative communication method such as 'Zigbee'. George also mentioned that I should start looking into how I will detect a player's swings by looking at existing technologies. I had currently looked a little at hardcoding pattern recognition into the Arduino device but George mentioned that I would be better off with a more dynamic method such as a 'neural network'. I will therefore look into methods that enable me to detect patterns in the data.
Conclusion	Look into other technologies rather than BLE. Look into activity and pattern detection methods.

Title	Producing Reliable Communications - Arduino & Android	21/10/2014

Discussion	To identify the problem with the communications I have implemented a graph which plots the data received over Bluetooth. The graph should draw six straight lines for each fixed 'int' value in the packet. As the data is being sent the graph shows the data is becoming unreliable and plots scrambled data. The next stage is to understand why the data is becoming unreliable. I have ordered XRF devices in case BLE cannot be used.
Conclusion	Data transfer gives different results under the same tests.

Title	Project Definition Form - Submitted	23/10/2014

Discussion	I have now completed my project definition form which includes the reasoning behind my project, the usefulness and why it is original. I have also provided a detailed breakdown of how I expect my project to progress over the coming months. Both of these documents have been handed in and submitted.
Conclusion	Submission of the Project Definition Form and Project Plan.

Title	Producing Reliable Communications - Arduino & Android	25/10/2014

Discussion	To understand if the Bluetooth device is the problem I have implemented USB communications between the Android device and the Arduino. This sends the same information but this time over USB and not Bluetooth. The results show me that the USB communications are 100% reliable which means both the Android and Arduino code are reliable and that the BLE module is the problem. My next step is to determine why the BLE device is causing these issues and see if it is still a useable device for my project.
Conclusion	Implemented USB communications which are 100% reliable. Need to find why BLE is not.

Title	Supervisor Meeting - 3	28/10/2014

Discussion	In my third meeting with George we discussed possible reasons why the data is being unreliable. George suggested that I try another Arduino device just to ensure that the Arduino Pro Mini is not the cause. George lent me an Arduino Uno to test this theory. Other theories we came up with that might help are to increase the baud rate, send the data twice and apply a checksum. I will look into trying each one of these suggestions if I cannot find the BLE problem.
Conclusion	Try another Arduino device, increase the baud rate, send the data twice and apply a checksum – potential fixed to the poor communications.

Title	Producing Reliable Communications - Arduino	03/11/2014

Discussion	After much research and testing I have found out why the BLE communications are unreliable. To get to this point though I have tested other Arduino devices, applied checksums, tried sending the data twice and then filtering out bad packets determined by the checksum. It appears that none of these methods helped reduce the poor transmission. In the end I finally came across and post that explains why I am having trouble sending data. In short Bluetooth Low Energy has a 20byte packet limit and each packet can only be sent every 7.5 milliseconds. With this key information I changed my Arduino code to only send packets less than 20 bytes with 10 millisecond intervals and found the data was reliable! To test the 20byte limit I gradually increased the bytes sent in a packet until I reached 21, which is when the data became incredibly unreliable again. This is a huge breakthrough and means the project can proceed!
Conclusion	Found BLE limitation causing the BLE unreliability.

Title	Supervisor Meeting - 4	04/11/2014

Discussion	In this meeting I described the BLE limitations to George. I proposed two solutions to this limitation. The first potential solution was to shorten the data I sent to just 20bytes, this would require removing the checksum and other key parts of the information to ensure I could send all of the important information. The second potential solutions was to write my own BLE firmware that would support multiple characteristics, this would enable me to send up to four 20byte packets per a 7.5 second window. The problem with this though is that it would take a considerable amount of time to do. Because of the amount of work still needing to be done George suggested that I look into shortening my data and to send it as bytes. George also recommended that I should start writing my report, find research papers to do with swing detection and log all of the research in my main report.
Conclusion	Need to shorten my BLE data and start writing my report.

Title	Producing Reliable Communications - Arduino	04/11/2014
Discussion	My final challenge was to create a packet structure that would fit all of the data I needed for successful communications (20 bytes). For each packet I decided that I would need a time stamp, pitch, roll, gyro x, gyro y and gyro z data. To ensure that I could fit all of this data in I needed to convert the current short values into bytes. I managed to convert each short value into a simple 2 byte chunk. Along with the data I needed a command structure to help me parse the information on the Android side. For this I use “$” to depict the start of a packet and “;” to depict the end of a packet and “,” as a value separator. This packet structure successfully allows reliable communications.	
Conclusion	BLE is now reliable. Need to progress more on the Android development.	

Title	Supervisor Meeting - 5	11/11/2014
Discussion	I have been to my fifth supervisor meeting with George today. He mentioned that my project was coming along nicely now that the BLE communication problem was understood and more reliable. George mentioned that the next step for me was to sample my existing dataset. The data when received on Android can range from 10packets to 60packets on an average swing. George now wants me to convert this to a set of data around 20 packets long so that it will be easier to process at a later period. George has also mentioned that I should aim to get a working model ready by the end of the year so that I have time to tweak and perfect the application after Christmas.	
Conclusion	Project progressing well. Need a working model by the end of the year.	

Title	Swing Detection Research	22/11/2014
Discussion	I have currently been reading multiple research papers about activity recognition. This is so that I can see how others have been detecting activities such as running or walking. Most of the research papers have a common library that they use to detect activities, this is ‘Weka’. Weka is a multipurpose multi-algorithm data mining tool. It allows you to look for patterns in data. Therefore I feel that this is the correct tool to use to help me detect swings. Weka works by ‘predicting’ the missing data and adding the prediction to a confusion matrix.	
Conclusion	Found a method of predicting swings using Weka.	

Title	Implemented Two Swing Related Screens	24/11/2014
Discussion	I have now implemented the ‘Swing Training’ screen which allows users to perform swings and correct the application if it incorrectly guesses their swing. It does this by updating the Weka classifier with the new instance data. This allows users to perfect their new swing profiles. I have also implemented the ‘Match’ screen that allows users to play a series of games and record their swings during these games. Future work will involve implementing a history screen to view this saved data. The data is currently saved in an SQLite database.	
Conclusion	Two screens implemented; ‘Swing Training’ and ‘Match’.	

Title	Supervisor Meeting - 6	02/12/2014

Discussion	In this meeting with my supervisor I discussed my findings and progress. George was impressed by my use of Weka and had multiple suggestions as to improve and proceed with my project. Firstly I will need to look into identifying the correct features of a swing. George also mentioned that along with identifying the correct features, I need to look at removing redundant features. This will help improve the time it takes to process the training data on app launch. The next suggestion by George was to pass a lot more data into the Weka explorer to help identify the correct algorithm to use. This will help me determine which algorithms are better at detecting swings. The next suggested change is to give the application a default training set of data when one is not provided by the user, this will allow users to use the application straight away. Finally, the report is suggested to be between 60-70 pages. I will look at starting some of the key areas that I have been making good progress on.
Conclusion	I need to look at the best features for swing detection and start making progress with my report.

Title	Designed the Mark 2 Device	09/01/2015

Discussion	With my current swing detection progress a new more robust Arduino device is needed for demonstration purposes. Currently I am able to make test swings without hitting a ball but if I was to use this on court then the device could potentially fall apart and break quite easily. Therefore I have redesigned the device using ‘Eagle PCB’. My design is currently being printed and I should expect it within the next two weeks, in which I will need to assemble it. I’m hoping that after this I will be able to make test shots on court and test the actual shot detection accuracy.
Conclusion	Redesigned the Arduino device.

Title	Started the Term 1 Progress Report	11/01/2015

Discussion	I have now started my term 1 progress report. I plan to break the structure down into three key areas; Android – the Android development progress, Arduino – the current development of the hardware device and finally Swing Detection – for my progress using Weka.
Conclusion	Term 1 report has three key sections.

Title	Created the Mark 2 Device	15/01/2015

Discussion	Using the new boards that have been printed, the Mark 2 version of the Arduino device has now been created. The device is more robust, lighter, smaller and cheaper to produce. This device can now be strapped to a Squash racket for further swing analysis.
Conclusion	Mark 2 device created successfully.

Title	Term 1 Progress Report Submitted	23/01/2015

Discussion	The Term 1 progress report has now been submitted.
Conclusion	Term 1 submission complete.

Title	Supervisor Meeting - 7	05/02/2015
Discussion	During this meeting I demonstrated my current progress to George with the actual device strapped to a racket. We both came to the conclusion that because the swing detection is looking good for its current stage that I should focus on implementing all of the screens so that they are at the same quality of implementation. After I have implemented all of the screens I should then look into improving and testing the accuracy of the swing detection. This will involve choosing the right set of features and algorithms to use whilst restricting features that do not benefit the detection. Along with implementing the rest of the screens I have been advised to get the report up to the same level as the development so that I do not fall behind on the report. The features that I have been told to pay attention to in the report are areas that have interested me during the development of the project, this includes showing the evolution of the prototype device with reasonings as to why the device have been changed. To further improve the device I have been told to look at how to better decide when a swing has started or stopped on the Arduino so that I do not cut out important swing information. Once I feel that the application is performing as desired I should then ask multiple users to take two sets of data so that I can understand the accuracy of swing detections in Weka. To do this I will need to implement a user profile screen which will allow users to easily create and choose their saved swing profiles. Finally, when I get the accuracy to a good level I will then need to look at implementing quirky elements such as a game won or lost gesture to help keep the score and suggest which shots users are good or bad at. This will help them target their training in the future.	
Conclusion	• Improve and test the device accuracy. • Implement all screens. • Create a user profile screen. • Work on the report.	

Title	Supervisor Meeting - 8	19/02/2015
Discussion	In this brief meeting we discussed how the project is moving in the right direction and that I needed to continue with the checklist of items as we discussed in the prior meeting. I also demonstrated the rubber that I have moulded around the device to protect it further.	
Conclusion	Continue with the checklist of items from the previous meeting.	

Title	Swing Detection Accuracy - Android	04/03/2015
Discussion	During the recent development I have been able to increase the swing detection accuracy on Android to 90%. This has been done by extending the swing contingency period during a swing recording and by removing redundant features. The main and only problem left with the application is the false positives being detected by the Weka library. When a device is raised fast before a swing the application can sometimes falsely classify this as a swing. This needs to be further explored.	
Conclusion	High level of swing detection accuracy although false positive problems.	

Title	Supervisor Meeting - 9	05/03/2015
Discussion	In this meeting we concluded that the project has reached a good point and that I need to look at getting the first draft of my report ready. I explained to George how the device was really accurate now (90%) but the main crippling element to the project is that it is incredibly hard to differentiate a pre-swing to a normal swing using the Weka library. George explained that I should look for methods to remove these false positives to open up the ability to get the participants during the demo session to try the device. Currently the device would only be best demonstrated by myself. We also talked about the possibility of the device being used on court by cancelling out the players movement with the phones sensor, however it has come to my attention that the sensor in most phones are not capable of the high level of precision. Ideas for the demonstration included playing videos demonstrating the devices use, this can be explored later.	
Conclusion	Write the first draft of the report and remove the false positives.	

Title	Finalised the Application Screens	15/03/2015
Discussion	The user interface of the Android application has now been finalised and all of the front end functionality has been completed.	
Conclusion	UI Complete.	

Title	Supervisor Meeting - 10	27/03/2015
Discussion	After meeting with George he recommended that I get the first draft of the project report done in a weeks time and sent to him for review. I asked the question of whether a class diagram would be needed for the project, George recommended to show the screen flow of the application rather than a full class hierarchy. Any time that is left at the end of the project will need to be spent on extras items like impact detection to improve the usability.	
Conclusion	Complete the first draft of the report for review.	

Title	First Draft of the Report Complete	03/04/2015
Discussion	The first draft of the project report has been written and sent to George for a review.	
Conclusion	First draft complete and sent for review.	

Title	Second Draft of the Report Complete	25/04/2015
Discussion	The second draft of the project report has been written and is now in the stage of fine turning the layout.	
Conclusion	Second draft complete.	

Title	Final Report and Project Submitted	29/04/2015
Discussion	My final year project has now been completed and submitted.	
Conclusion	Final year project complete and submitted.	

10.2 Testing Strategy

The purpose of this testing strategy is to define tests for the key areas of the SquashT application that impact the intended use and usability of the end product.

Based on the Agile Testing Quadrants (Crispin and Gregory, 2009) the following series of white and black box testing techniques will be used to help test SquashT.

Quadrant One

For quadrant one “Unit” tests have been chosen to help test the key architectural components and understand if the components still function as desired between releases. The unit tests will be run at each release of the project allowing the automated tests to confirm that any new changes made do not affect the existing functionality. The unit tests will be specifically targeted at the key events of the application to ensure that the main functionality works as intended. Tools that enable the help of unit testing for Android applications include Android “TestCase” and “ActivityTestCase”.

Quadrant Two

In quadrant two multiple types of tests have been chosen to ensure that the application functions as defined by its requirements. The first type of test to be implemented is a “User Story” which is a type of functional test. The user story will be written in the inception and construction phase of the project, which allows the initial requirements and tasks to be defined. Using the user story a “Functional” set of tests will be written that test specific items in the application and ensure that they handle the potential boundary inputs. The next testing technique that will be used is the use of “Prototypes” to help test the individual hardware components on the Arduino device. Because the Arduino device cannot be debugged the best solution is to load specific code scenarios onto the device so that the Android application can check that the responses received are correct and that the hardware works correctly. The prototyping phase will be run for each new iteration of the Arduino device to ensure that all of the components still successfully run as intended.

Quadrant Three

For this quadrant, three testing techniques have been chosen. “Acceptance” testing, which will be confirmed at each release, ensures that the implemented functionality works as defined by its accompanying specification and that the next item of the functionality can be started. “Scenario” testing helps ensure that the product works as desired by the end users by testing the system from a users perspective. This is an important area that will be tested at the end of each item of functionality to ensure that the user experience does not degrade as new items of functionality are implemented. Finally “Exploratory” testing will be undertaken to identify possible weaknesses of the system. The exploratory tests will be aimed at testing the new functionality implemented when it is marked as complete to ensure that any major bugs are documented.

Quadrant Four

During each development release of the project “Performance” tests will be run to ensure that the application is still capable of providing feedback to users without significant changes in latency. This will help understand early if a new implementation has been implemented incorrectly or has memory leaks causing the application to slow down and become unresponsive. To help test the performance code logs can be applied to record the starting and stopping times of various functions. These times can then be recorded and tracked through the development of the project to understand if the performance of the application is decreasing.

10.3 Quadrant One and Two Testing

10.3.1 Unit Test Cases

```
/**
* This test ensures that the swing prediction processor correctly identifies
   the correct swing types.
*/
public class SwingTypeTest extends ActivityUnitTestCase<MainActivity> {
    //The main activity.
    private MainActivity mainActivity;

    public SwingTypeTest() {
        super(MainActivity.class);
    }

    @Override
    protected void setUp() throws Exception {
        super.setUp();

        // Starts the MainActivity of the target application
        startActivity(new Intent(getInstrumentation().getTargetContext(),
            MainActivity.class), null, null);

        //Store the main activity.
        mainActivity = getActivity();

        //Ensure the tests are full-filled.
        getInstrumentation().waitForIdleSync();

        //Initialise the Helper class.
        Helper.getInstance().initialise(mainActivity);
    }

    //Check that the main activity has been instantiated.
    @SmallTest
    public void testMainActivityIsInstantiated() {
        assertNotNull(mainActivity);
    }

    //Check that the helper class has been instantiated.
    @SmallTest
    public void testHelperIsInitialised() {
        try {
            Helper.getInstance().showToast("", Toast.LENGTH_SHORT, false);
        } catch (Exception e) {
            fail("Helper␣is␣not␣initialised.");
        }
    }

    //Check that the helper class has been instantiated.
    @SmallTest
    public void testSwingTypeProcessorIsInitialised() {
        assertTrue(SwingTypeProcessor.getInstance().isInitialised());
    }

    //For two given swings, predict the swing types.
    @MediumTest
    public void testSwingPrediction() {
```

```
//Create the forehandSwing.
final Swing forehandSwing = new Swing();

//Create some mock forehandSwing data to be determined.
forehandSwing.timeValues = new ArrayList<Short>(Arrays.asList((short)
    1, (short) 2, (short) 3, (short) 4, (short) 5,
        (short) 6, (short) 7, (short) 8, (short) 9, (short) 10));
forehandSwing.pitchValues = new ArrayList<Short>(Arrays.asList((short)
     1, (short) 2, (short) 3, (short) 4, (short) 5,
        (short) 6, (short) 7, (short) 8, (short) 9, (short) 10));
forehandSwing.rollValues = new ArrayList<Short>(Arrays.asList((short)
    1, (short) 2, (short) 3, (short) 4, (short) 5,
        (short) 6, (short) 7, (short) 8, (short) 9, (short) 10));
forehandSwing.gyroXValues = new ArrayList<Short>(Arrays.asList((short)
     1, (short) 2, (short) 3, (short) 4, (short) 5,
       (short) 6, (short) 7, (short) 8, (short) 9, (short) 10));
forehandSwing.gyroYValues = new ArrayList<Short>(Arrays.asList((short)
     1, (short) 2, (short) 3, (short) 4, (short) 5,
        (short) 6, (short) 7, (short) 8, (short) 9, (short) 10));
forehandSwing.gyroZValues = new ArrayList<Short>(Arrays.asList((short)
     1, (short) 2, (short) 3, (short) 4, (short) 5,
        (short) 6, (short) 7, (short) 8, (short) 9, (short) 10));

assertTrue(forehandSwing.timeValues.size() == 10);

//Set the sample size of the forehandSwing.
forehandSwing.SAMPLE_SIZE = 5;

//Sample the data down to 5 packets.
forehandSwing.sampleData();

//Determine the features of the forehandSwing.
forehandSwing.determineFeatures();

//Check that the data has been correctly sampled.
assertTrue(forehandSwing.timeValuesSampled.length == 5);

//Create the forehandSwing.
final Swing backhandSwing = new Swing();

//Create some mock forehandSwing data to be determined.
backhandSwing.timeValues = new ArrayList<Short>(Arrays.asList((short)
    10, (short) 20, (short) 30, (short) 40, (short) 50,
        (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
backhandSwing.pitchValues = new ArrayList<Short>(Arrays.asList((short)
     10, (short) 20, (short) 30, (short) 40, (short) 50,
       (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
backhandSwing.rollValues = new ArrayList<Short>(Arrays.asList((short)
    10, (short) 20, (short) 30, (short) 40, (short) 50,
        (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
backhandSwing.gyroXValues = new ArrayList<Short>(Arrays.asList((short)
     10, (short) 20, (short) 30, (short) 40, (short) 50,
       (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
backhandSwing.gyroYValues = new ArrayList<Short>(Arrays.asList((short)
     10, (short) 20, (short) 30, (short) 40, (short) 50,
       (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
backhandSwing.gyroZValues = new ArrayList<Short>(Arrays.asList((short)
   10, (short) 20, (short) 30, (short) 40, (short) 50,
        (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
        assertTrue(backhandSwing.timeValues.size() == 10);

        //Set the sample size of the forehandSwing.
        backhandSwing.SAMPLE_SIZE = 5;

        //Sample the data down to 5 packets.
        backhandSwing.sampleData();

        //Determine the features of the forehandSwing.
        backhandSwing.determineFeatures();

        //Check that the data has been correctly sampled.
        assertTrue(backhandSwing.timeValuesSampled.length == 5);

        //Generate the Weka Attributes.
        FastVector fvWekaAttributes = Helper.getInstance().getWekaAttributes()
            ;

        //Create an empty training set.
    Instances swingInstanceData = new Instances("swing_set", fvWekaAttributes,(
       forehandSwing.timeValuesSampled.length + backhandSwing.timeValuesSampled
       .length));
        swingInstanceData.setClassIndex(fvWekaAttributes.size() - 1);

        //Add data to the training set.
        Instance tempInstance;
        for (int i = 0; i < forehandSwing.timeValuesSampled.length; i++) {
            // Create and add the instance and set the type to a Backhand
                Drive.
            tempInstance = Helper.getInstance().createWekaInstance(
                forehandSwing, i, "Forehand␣Drive");
            //Add the instance to our new instance set.
            swingInstanceData.add(tempInstance);
        }
        for (int i = 0; i < backhandSwing.timeValuesSampled.length; i++) {
            // Create and add the instance and set the type to a Backhand
                Drive.
            tempInstance = Helper.getInstance().createWekaInstance(
                backhandSwing, i, "Backhand␣Drive");
            //Add the instance to our new instance set.
            swingInstanceData.add(tempInstance);
        }
        //Check that the total number of instances present is 10 (Forehand +
            Backhand instances).
       assertTrue(swingInstanceData.numInstances() == 10);

        // Retrain the classifier with the new data.
        SwingTypeProcessor.getInstance().retrain(swingInstanceData, new
            BuildClassifierListener() {
            @Override
            public void onBuildSuccess() {
                //Passed.
            }
            @Override
            public void onBuildFail() {
                fail("Failed␣to␣build␣the␣classifier.");
            }
        });
        //Check that the forehand swing equals a forehand-drive. A forehand
            drive is returned as index 0.
        assertEquals(Swing.FOREHAND_DRIVE, SwingTypeProcessor.getInstance().
            determineSwingType(forehandSwing));

        //Check that the backhand swing equals a backhand-drive. A backhand
            drive is returned as index 1.
        assertEquals(Swing.BACKHAND_DRIVE, SwingTypeProcessor.getInstance().
            determineSwingType(backhandSwing));
    }
}
```

10.3.2 Functional Test Cases - Robust Boundary Value Analysis

BVA Sampling Test Values
-1
0
1
5
9
10
11

Table 6: Testing the Sampling of Data with BVA on a 10 packet dataset.

```
/**
 * This test ensures that the swing data samples as desired.
 */
public class DataSamplingTest extends ActivityUnitTestCase<MainActivity> {
    //Swing to test the BVA of sampling data.
    private Swing bvaSwing;

    public DataSamplingTest() {
        super(MainActivity.class);
    }

    @Override
    protected void setUp() throws Exception {
        super.setUp();

        // Starts the MainActivity of the target application
        startActivity(new Intent(getInstrumentation().getTargetContext(),
            MainActivity.class), null, null);

        //BVA Swing setup.
        bvaSwing = new Swing();

        //Create some mock forehandSwing data to be determined.
        bvaSwing.timeValues = new ArrayList<Short>(Arrays.asList((short) 10, (
            short) 20, (short) 30, (short) 40, (short) 50,
                (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
        bvaSwing.pitchValues = new ArrayList<Short>(Arrays.asList((short) 10,
            (short) 20, (short) 30, (short) 40, (short) 50,
                (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
        bvaSwing.rollValues = new ArrayList<Short>(Arrays.asList((short) 10, (
            short) 20, (short) 30, (short) 40, (short) 50,
                (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
    bvaSwing.gyroXValues = new ArrayList<Short>(Arrays.asList((short) 10,
        (short) 20, (short) 30, (short) 40, (short) 50,
            (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
    bvaSwing.gyroYValues = new ArrayList<Short>(Arrays.asList((short) 10,
        (short) 20, (short) 30, (short) 40, (short) 50,
            (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));
    bvaSwing.gyroZValues = new ArrayList<Short>(Arrays.asList((short) 10,
        (short) 20, (short) 30, (short) 40, (short) 50,
            (short) 60, (short) 70, (short) 80, (short) 90, (short) 100));

}

//Sampling Robust BVA Test #1
@SmallTest
public void testBVASample1() {
    //Check that the bvaSwing values are equal to 10.
    assertTrue(bvaSwing.timeValues.size() == 10);

    //Set the sample size of the swing.
    bvaSwing.SAMPLE_SIZE = -1;

    //Sampling the data should fail.
    assertFalse(bvaSwing.sampleData());
}

//Sampling Robust BVA Test #2
@SmallTest
public void testBVASample2() {
    //Check that the bvaSwing values are equal to 10.
    assertTrue(bvaSwing.timeValues.size() == 10);

    //Set the sample size of the swing.
    bvaSwing.SAMPLE_SIZE = 0;

    //Sampling the data should fail.
    assertFalse(bvaSwing.sampleData());
}

//Sampling Robust BVA Test #3
@SmallTest
public void testBVASample3() {
    //Check that the bvaSwing values are equal to 10.
    assertTrue(bvaSwing.timeValues.size() == 10);

    //Set the sample size of the swing.
    bvaSwing.SAMPLE_SIZE = 1;

    //Sampling the data should pass.
    assertTrue(bvaSwing.sampleData());

    //Check that the data has been correctly sampled.
    assertTrue(bvaSwing.timeValuesSampled.length == 1);
}

//Sampling Robust BVA Test #4
@SmallTest
public void testBVASample4() {
    //Check that the bvaSwing values are equal to 10.
    assertTrue(bvaSwing.timeValues.size() == 10);
```

```
        //Set the sample size of the swing.
        bvaSwing.SAMPLE_SIZE = 5;

        //Sampling the data should pass.
        assertTrue(bvaSwing.sampleData());

        //Check that the data has been correctly sampled.
        assertTrue(bvaSwing.timeValuesSampled.length == 5);
    }

    //Sampling Robust BVA Test #5
    @SmallTest
    public void testBVASample5() {
        //Check that the bvaSwing values are equal to 10.
        assertTrue(bvaSwing.timeValues.size() == 10);

        //Set the sample size of the swing.
        bvaSwing.SAMPLE_SIZE = 9;

        //Sampling the data should pass.
        assertTrue(bvaSwing.sampleData());

        //Check that the data has been correctly sampled.
        assertTrue(bvaSwing.timeValuesSampled.length == 9);
    }

    //Sampling Robust BVA Test #6
    @SmallTest
    public void testBVASample6() {
        //Check that the bvaSwing values are equal to 10.
        assertTrue(bvaSwing.timeValues.size() == 10);

        //Set the sample size of the swing.
        bvaSwing.SAMPLE_SIZE = 10;

        //Sampling the data should pass.
        assertTrue(bvaSwing.sampleData());

        //Check that the data has been correctly sampled.
        assertTrue(bvaSwing.timeValuesSampled.length == 10);
    }

    //Sampling Robust BVA Test #7
    @SmallTest
    public void testBVASample7() {
        //Check that the bvaSwing values are equal to 10.
        assertTrue(bvaSwing.timeValues.size() == 10);

        //Set the sample size of the swing.
        bvaSwing.SAMPLE_SIZE = 11;

        //Sampling the data should pass.
        assertFalse(bvaSwing.sampleData());
    }
}
```

10.4 Quadrant Three Test Plan

Exploratory Test Plan

SquashT User Profile Testing	15/04/15
	Software Revision 250

Risks and Tasks

Risks	Tasks
Creating a new profile with no name might cause an invalid file to be created on disk.	Create a zero length file name.
Logging out of a new user profile may cause errors with the profile not being saved.	Check new profiles are saved on creation.
Deleting a new user profile may cause problems if it has not been written to disk.	Attempt to delete a new user profile.
User profiles when deleted may still remain on disk.	Check that profiles are removed from disk when deleted in the application.
User might be able to use the phones return button to go back as a profile is loading. This could cause a thread completing in the wrong view.	Check that when a profile is loading no hardware buttons can interfere apart from the home button.
If a user places a file that is not of type "ARFF" in the SquashT directory it could potentially be displayed in the user profiles list.	Create a new file in the SquashT directory, which is not of type ".arff" and check that it does not appear in the user profiles list.
Creating a user profile with the same name as an existing profile might overwrite the previous data.	Create a new user profile with the same name as an existing file to check that data is not overwritten.
Opening a corrupt file may cause the application to crash.	Edit a file so that it does not match the standard convention and check that the application catches the error and displays a message to the user.

Functions and Characteristics Covered

Areas Under Test	Test Equipment
User Profile Loading.	SquashT Application.
User Profile Delete.	SquashT Application & Android Stock File Manager.
User Profile Create.	SquashT Application & Android Stock File Manager.

Functions and Characteristics Not Covered

- Testing user profile swing data in the swing training screens.
- Checking for errors in valid swing profiles.
- Maximum number of profiles possible in the Android “ListView”.

Issues Found During Testing

- User Profiles are not removed from the list when deleted until navigation is made away from the user profile screen.
- The user profile list shows files of all types instead of just showing “.arff” file types.
- The application does not show an error to the user when a corrupt profile is attempted to be loaded.

Exploratory Test Cases

ID	Purpose	Observation	Pass/Fail
1	Check that a zero length file name is not accepted.	Functionality catches this event and alerts the user to enter a valid name.	Pass
2	Check that new profiles are saved to disk before they are used.	File shows instantly in the file manager.	Pass
3	Check that deleted profiles are removed from disk.	Files are removed from disk but the current user list is not updated until the user navigates to the profile selection again. The list needs to be updated immediately.	Fail
4	Check that hardware buttons on the phone do not cancel a profile that is loading, which prevents a bad application state.	As desired the back button does not interfere with a profile that is loading.	Pass
5	Create a file that is not the expected “.arff” file in the SquashT directory.	The file shows in the list and can be selected. Once it is selected it returns the user back to the list as it cannot open the file. A filter needs to be used when loading the files.	Fail

6	Create a new user profile with the same name as an existing file to check that the existing profile is not overwritten.	The application loads the existing profile instead of creating a new one.	Pass
7	Create a file that does not conform to the required document type and check that the application does not open the file and reports a corruption error to the user.	The application currently attempts to open the file and returns the user back to the user profile list without an error message. Need to implement a corruption error message.	Fail

10.5 Quadrant Four Test Plan

Performance Test Plan

SquashT Data Mining Algorithm Testing	16/04/15
	Software Revision 250

Functions and Characteristics Covered

Areas Under Test	**Test Equipment**
Algorithm Accuracy.	SquashT Application and Weka Desktop GUI.
Algorithm Build Time.	SquashT Application.
Algorithm Classification Time.	SquashT Application.

Functions and Characteristics Not Covered

- Algorithms that do not meet a standard accuracy of 95+ percent.
- Algorithm feature selection.

Algorithms Tested

- Dagging.
- FT.
- IBK.
- SMO.

Highest Performing Algorithms Based on the Criteria

1. SMO.
2. IBK.

3. FT.
4. Dagging.

Performance Test Cases

Test Case 1: Accuracy Test Case

Description: This test case looks at the accuracy of each algorithm correctly predicting the swing types. For each algorithm two sets of swing data are provided, one to train the algorithm and one to be classified. The algorithms accuracy is then recorded from the Weka GUI output.

ID	Algorithm	Accuracy (%)
1	Dagging.	97.6
2	FT.	96
3	IBK.	98
4	SMO.	98

Test Case 2: Build Time Test Case

Description: This test case looks at the time each algorithm takes to load a full swing data set. This is done each time a user profile is selected in the application. Each Algorithm has been tested five times and the average from these results have been used. Code level timers have been used with the time values being printed to console when complete.

ID	Algorithm	Build Time (Seconds)
1	Dagging.	39.072
2	FT.	30.947
3	IBK.	2.918
4	SMO.	7.424

Test Case 3: Classification Time Test Case

Description: This test case looks at the time each algorithm takes to classify a swing type for the given swing instance. Each algorithm has been tested five times and the average from these results have been used. Code level timers have been used with the time values being printed to console when complete.

ID	Algorithm	Classification Time (Milliseconds)
1	Dagging.	75.6
2	FT.	14.6
3	IBK.	329
4	SMO.	15.4

10.6 Marketability Survey

Deployment	Facebook poll targeting Aston Universities casual Squash players.
Question	If there was a device that could track your movements, shots and score around the court to help you improve, how much would you realistically be willing to pay for it (that is if you would buy it). The device would recommend game play advice as you come off the court, such as what shots to play and avoid in the next game.

Results	
Option	**Votes**
Not Interested in the Concept	0
£10-£15	3
£15-£20	4
£25-£30	8
£30+	2

10.7 User Instructions and Installation

Android - Running the Source Code

To run the Android source code you will need an Android IDE, such as Android Studio.

1. In Android studio, select 'File' -> 'Import Project' and select the project path.
2. Follow the wizard, which will configure the libraries.
3. Once you have imported the project, plug in an Android device that has developer mode enabled.
4. Select the run tab in Android Studio and run the 'BootActivity'.

Android - Running the APK Build

To run the Android APK build, you will need to place it onto a desired Android device running 4.3 or later.

1. On the Android device go into settings and enable 'Unknown Source' installations.
2. Copy the APK onto the Android device from a PC.
3. Using a file manager on the Android device, locate the APK and click it to install the application.
4. The application should now be installed and located in the devices 'Application Drawer'.

Arduino

The Arduino source code can be compiled and built using Arduino IDE 1.0.6.

1. Open the Source file by double clicking it.
2. The Arduino IDE should open and you should have access to the 'Tools section'.
3. Select 'Tools' -> 'Board' -> 'Arduino Pro Mini (3.3v, 8 Mhz) w/ ATmega328'.
4. With the device connected using an FTDI cable, press 'Upload'.