Gerard Brosnan – Computing With Multimedia

T00161582  REF: 2014-12-08-007

Android Motion Sensor CAPABILITIES for Tracking and Analysing Resistance-Training Workouts

# Abstract

Technology plays a big role in modern sports and fitness. Professional athletes and teams use technology to improve their performance. Mobile application have also become very popular among non-professional and enthusiasts. A wide range of applications provide functionality to track and monitor cardiovascular training session such as running and walking. However, there are few applications designed specifically for resistance-training where repetitions and sets need to be counted and analysed to help the user improve their performance.

One of the aims of this project was to test the capability of the motion sensors in an Android device, for detecting motion during the performance of an exercise. The hardware and software fusion for the micro sensors were researched and tested. A server was set up to collect the sensor data for analysis, interpretation and feedback for the user.

The approach taken for analysis and interpretation of the data was to use a neural network. This network would continuously take in data until it is has an exercise finally trained. This way, when a user performance the exercise, the network can recognize the movement and other factors related to us such as speed and reps.

Five exercises have been chosen for testing this technology.

Table of Contents

[Abstract 1](#_Toc416342269)

[Chapter 1: Introduction 4](#_Toc416342270)

[1.1 Overview 4](#_Toc416342271)

[1.2 Problem Statement 4](#_Toc416342272)

[1.3 Aims and Objectives 5](#_Toc416342273)

[Chapter 2: Research 6](#_Toc416342274)

[2.1 Resistance Training 6](#_Toc416342275)

[2.1.1 Principles 6](#_Toc416342276)

[2.1.2 Principle of Progressive Overload 6](#_Toc416342277)

[2.1.3 1RM 7](#_Toc416342278)

[2.1.4 Velocity-based training 8](#_Toc416342279)

[2.1.5 Velocity Measuring Devices 9](#_Toc416342280)

[2.1.6 Workout Planning 9](#_Toc416342281)

[2.1.7 Strength Training Conclusion 9](#_Toc416342282)

[2.2 Android Sensors 10](#_Toc416342283)

[2.2.1 Galaxy S3 Sensor List 10](#_Toc416342284)

[2.2.2 Accelerometer 10](#_Toc416342285)

[2.2.3 Gyroscope 11](#_Toc416342286)

[2.2.4 Barometer 11](#_Toc416342287)

[2.2.5 Sensor fusion and filtering 12](#_Toc416342288)

[2.2.6 Galaxy S3 Sensor Hardware 13](#_Toc416342289)

[2.2.7 ST iNemoEngine\_PAAP 15](#_Toc416342290)

[2.2.8 Android Sensor API 16](#_Toc416342291)

[2.2.9 The Physics of Movement 18](#_Toc416342292)

[2.3 Artificial Neural Networks 19](#_Toc416342293)

[2.3.1 Overview 19](#_Toc416342294)

[2.3.2 History 19](#_Toc416342295)

[2.3.3 Neural Networks vs Von Neumann 20](#_Toc416342296)

[2.3.4 Perceptron 21](#_Toc416342297)

[Chapter 3: Methodology 24](#_Toc416342298)

[3.1 Exercises 24](#_Toc416342299)

[3.2 Android 24](#_Toc416342300)

[3.3 The Server 25](#_Toc416342301)

[Chapter 4: Design 26](#_Toc416342302)

[4.1 Vision Document 26](#_Toc416342303)

[4.1.1 Scope 26](#_Toc416342304)

[4.1.2 Risk Analysis - Overview 27](#_Toc416342305)

[4.1.3. Risk Analysis - Full List 28](#_Toc416342306)

[4.1.2 High Level Feature List 29](#_Toc416342307)

[4.1.3 Prioritization of Features 30](#_Toc416342308)

[4.2 Functional Specification 31](#_Toc416342309)

[4.2.1 Use-Case Diagram 31](#_Toc416342310)

[4.2.2 User Stories 32](#_Toc416342311)

[4.2.3 Prototype 39](#_Toc416342312)

[4.3 System Design 40](#_Toc416342313)

[4.3.1 Architecture 40](#_Toc416342314)

[4.3.2 State Diagram – Gym Routine Plan 41](#_Toc416342315)

[4.3.6 Sprints 42](#_Toc416342316)

[Chapter 5: Implementation 45](#_Toc416342317)

[5.1 Sprint 1: Start-up/Prototype 45](#_Toc416342318)

[5.2 Sprint 2: Pre-Processing of Data 50](#_Toc416342319)

[5.3 Sprint 3: Artificial Neural Network 65](#_Toc416342320)

[Chapter 6: Findings and Analysis 65](#_Toc416342321)

[6.1 Introduction 65](#_Toc416342322)

[References 66](#_Toc416342323)

# Chapter 1: Introduction

## 1.1 Overview

In a culture where people have become more aware of their physical well-being, we see all around us how technology can aid in physical fitness. Fitness apps are used to track fitness data such as speed, foot-steps, time, calorie counts and heart rate. It’s common to see people with smart phones strapped onto their arms while doing cardio training, in both outdoors and indoor fitness sessions. Not only can smart phones track fitness data, they can also provide music and radio for motivation (or distraction from the challenging experience) while also organizing fitness-data spanning over several weeks.

One area of fitness where smart phones are not so common is in resistance training. A quick browse of the Google Play store may show many resistance training apps that can log data for reps and sets. However, none of these apps can dynamically track and analyse live data similar to the apps design for cardio workouts. During a resistance training workout, sets and reps need to be counted while other factors such as speed and breaks can be influential in a person’s workout session. Currently no phone applications provide features for tracking such data with the exception of a few that require other expensive wearable devices.

This study will assess the important factors of tracking a resistance training workout. The sensor hardware capabilities of an Android smartphone will be determined for the functionality that is required in reading resistance training exercises. Live data will be analysed and interpreted for feedback to the user to help them track their progress. This data analysis and interpretation will be the main area of focus for this project using a neural network.

## 1.2 Problem Statement

Can a smart phone be used, to sense a user’s gym exercise activities and then use the raw data to be dynamically analysed and reported live back to the user?

## 1.3 Aims and Objectives

**Aim:** To create an Android smart phone to track and analyse a resistance training work out and provide instant feedback for the user.

**Objectives:**

* Create a user-friendly app that can plan and organise a training work out
* Use the phone sensors to track movements and time
* Send data to a server for analysis
* Receive results and suggestions back from the server application

Sending sufficiently accurate data to the server application and interpreting this data for user feedback could have many benefits for the user. The important factors of gym plans and progress need to be researched for the writing up of a user requirements list. The capabilities and accuracy of smart phone sensors will have an influence on how exercise data can be analysed for the user. Extensive research into how this data can be can be analysed and interpreted will be carried out. This research will aim to come up with solutions to provide reliable feedback for the user.

# Chapter 2: Research

## 2.1 Resistance Training

### 2.1.1 Principles

Resistance Training involves doing any type of exercise that can make the muscles contract against a chosen external resistance (Weil, 2014). It has many health benefits, especially in today’s world where many people very rarely need get to work their muscles for what they were designed for.

In a report by the NSCA, there are 3 basic principles discussed which are overload, variation and specificity (Stone, 2000):

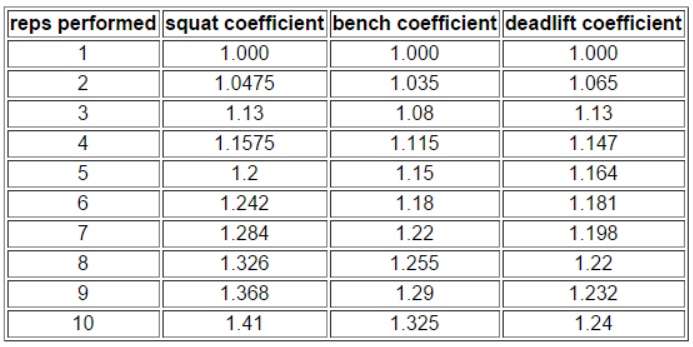
* Overload is for stimulating properly a specified physical adaptation. It involves exercising beyond normal levels of intensity. Intensity is measured by the amount of weight lifted where as volume is measured by the amount of repetitions and sets performed in each exercise. The best estimation for the amount completed is Volume load (load x volume).
* Variation refers to the speed of movement and volume and exercise selection. Specific changes to this area can result in superior enhancement in performance abilities.
* Specificity is described as the most important principle. It involves the mechanics of exercise movements including body regions, direction, muscle regimes, timing and velocity. The factor that is seen as the most important one is the rate of force production. This involves the timing and velocity of movements.

This information points out to a key part of the project proposal where sensors could help in the analysis and feedback of timing and velocity movement.

### 2.1.2 Principle of Progressive Overload

Principle of progressive overload is well known in the area of Strength Training that it is viewed as “universally accepted as the model that creates the greatest gains in strength” (Weil, 2014). The physiologist explains that to follow this principle, one has to lift weights which are heavy enough to cause fatigue in the muscle towards the end of a set. Once you are able to reach the end of a set, the weight can then be increased to keep the challenge progressive. More weight means fewer reps but increasingly strong muscles means one can accomplish more of these reps.

### 2.1.3 1RM



***Figure 2.1-1*** Rep co- efficiencies (Butt, 2001)

As with the Principle of Progressive Overload, one can accomplish more repetitions with a lighter weight compared to doing it with a heavier load. A weightrainer.net article (Butt, 2001) explains if one can perform a number of repetitions with a certain weight, a calculation can be made to determine how much weight the person can lift in just one rep. This weight is known as the 1RM. If a person’s 1RM is known, a personal trainer can then use this to determine the weight load for the amount of desired reps for any given workout plan. This article provides a table, ***figure2.1-1***, from the National Strength and Conditioning Association that provides co-efficiencies for every rep performed in three well known exercises.

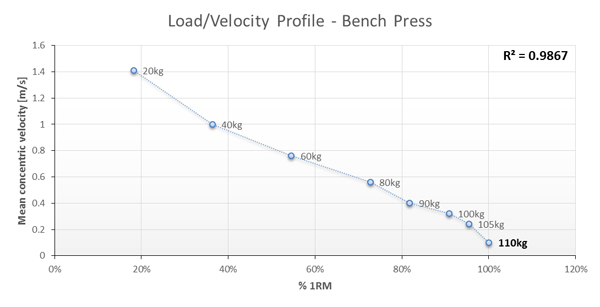
With these efficiencies, 3 areas of useful information is provided for the trainer. This includes determining muscle efficiency, judging progress and determining weak links in a person’s strength. These figures could be used to provide essential feedback to a user of a gym application.



***Figure 2.1-2*** The Brzycki Forumla(Butt, 2001)

A formula mentioned in the article for determining the 1RM is the Brzycki Formula, ***figure 2.1-2***. The Scientific Electronic Library Online (ScIELO) provides an article that investigates the accuracy of this formula (Materko, 2007). According to this article, the tests performed gave positive results. Acceptable reliability was proven and it concluded that this formula can be used as a tool for predicting a person’s 1RM weight load.

### 2.1.4 Velocity-based training



***Figure 2.1-3*** The Load/Velocity Relationship(Jovanović, 2014)

An article in Elifefts (Jovanović, 2014) explains the Load/Velocity Relationship. The load on each rep has an impact on how fast the user can complete it. The lighter the load, the faster the rep, while the heavier the load, the speed of the rep gets lower. It also explains how this relationship can be graphed as a simple linear as show below, show in ***figure 2.1-3***.

It points out as an example, that 80% of 1RM would always have a similar velocity regardless of changes to the user’s 1RM over time (if a person’s 1RM were to improve over a number of weeks). This means that by knowing the speed of a particular percentage of a 1RM, the user would always know they have reached a certain exertion level if that particular velocity could be detected for them.

Another interesting point is the Velocity/Exertion relationship. The velocity of the final rep (failure) is similar if not exact to the velocity of the 1RM. This is true regardless of load or the amount of repetitions. When the user keeps going until exertion is reached, the same velocity should always occur.

The relationships described here represents a concept that is used in velocity based strength training. It’s a concept that can predict, monitor and auto-regular a person’s strength training program.

### 2.1.5 Velocity Measuring Devices

Velocity and using velocity measuring devices is discussed in an NSCA article (Mann, 2014). It points out that tracking real-time velocity and power can provide information on how an athlete moves weight at a certain velocity and improve a specific goal. The use of velocity can determine the correct load for the day that is not based on previous sessions like traditional methods might do. It’s claimed that a velocity measuring device can provide live feedback which can determine the correct loads to lift for the user.

### 2.1.6 Workout Planning

There are many types of workouts plans that can be used weekly and an article in Muscle & Fitness describes four classic templates including the conjugate system, the linear periodization, undulating periodization and the 5x5. (Tuthill, 2014). All these templates involve doing different exercises depending on the day of the week. Sets and weights change weekly in accordance with the progressive overload as previously discussed.

With this information, it is obvious that a software application could provide benefits in keeping track of workout data and while providing feedback to the user, especially on the principle of progressive overload where weight progression can change after each workout.

### 2.1.7 Strength Training Conclusion

A brief investigation reveals how an application can play a big role in analysing, adjusting and providing feedback for a strength training plan. There are many scenarios depending on the training plan goal and progress, where a lot of different data will need to be tracked and adjusted.

Velocity can play a huge role in strength training and if the phone’s sensors can monitor and compare the different velocities involved in a users’ program, than the user can benefit greatly without ever the need for buying any extra hardware.

## 2.2 Android Sensors

This section will look into Android’s sensor capabilities. In particular the sensors for Samsung’s Galaxy S3 as this is the device that will be used in this project.

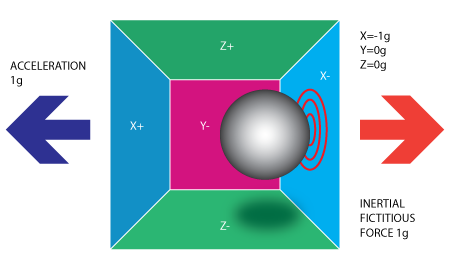
### 2.2.1 Galaxy S3 Sensor List

According to Samsung’s specification list (Samsung, 2014), the Galaxy S3 has six sensors;

* Accelerometer
* RGB light
* Digital compass
* Proximity
* Gyroscope
* Barometer.

### 2.2.2 Accelerometer

The accelerometer is explained as a device that can measure acceleration or vibration of a moving object (Omega, 2014). The explanation describes how the vibration or motion change causes force which in turn causes the squeeze in mass within the device. In some accelerometers this can cause electric charge which is proportional to the acceleration.



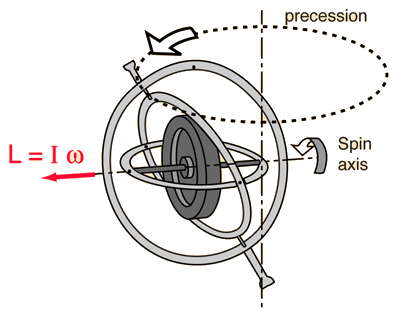
***Figure 2.2-1*** Acceleration and force (Accelerometer & Gyro Tutorial, 2014)

An Intractable tutorial (Accelerometer & Gyro Tutorial, 2014) provides a simple illustration, ***Figure 2.2-1***, where the acceleration of the box is in the direction of the left (X+). This forces the ball towards the right, the opposite direction, where the ball hits the x- wall. The result is a gravity force of -1g.

There are many areas where accelerometers are used (Electronics, 2014), such as vibration of vehicles, machines, in agriculture to detect behaviour of animals and they can also be found in health monitoring applications to report data. The accelerometer has become so widely used that Wired UK published an article titled “Accelerometers are the most under-appreciated technology innovation”. The article (Tatton-Brown, 2012), discusses the impact theses sensors have made on technologies we use how it has made certain gaming features possible such as titling on control devices.

### 2.2.3 Gyroscope

A Canadian science museum defines a Gyroscope as “any object mounted so that it turns very quickly around an axis of symmetry” (Museum, 2014).



***Figure 2.2-2*** A conventional gyroscope (HyperPhysics, 2014)

Gyroscopes respond to rotation and they can measure rotation in inertial space. The example in ***figure 2.2-2*** illustrates a conventional gyroscope where as in recent years solid state gyroscopes have become available. These newer devices measure vibration and the direction the vibration is applied from (Altheris, 2014).

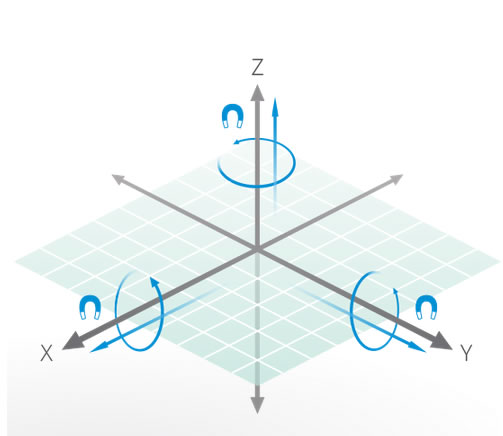
Gyroscopes can be used in many situations (Gyroscopes.org, 2014) including robotics, racing cars, science demonstration and computing pointing devices where movements and directions of movements are measured for various types of scenarios.

### 2.2.4 Barometer

Barometers measures atmospheric pressure and are used in phones to for weather forecasting and detecting altitude (Levi, 2011). It also is a passive sensor that is not heavy on battery use.

### 2.2.5 Sensor fusion and filtering

An EE article explains Inertial Measurement Units (IMUs) that combine a three-axis accelerometer and a three-axel gyroscope into one unit (Johnson, 2014). It focuses on an IMU unit by Bosch that uses different combinations of sensors such as an accelerometer-magnetometer for navigation, the accelerometer-gyroscope combination for gaming and recognising gesture. This unit also includes a barometer. The IMU device has features motion tracking, tremor cancelation, distortion detection and stability enhancement for indoor tracking.



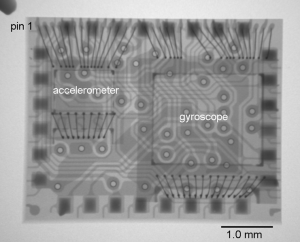
***Figure 2.2-3*** Acceleration (Sailing, 2014)

An illustration of a nine-axis sensor, ***figure 2.2-3***, shows the different movement that can be detected combining an accelerometer, a gyroscope and a magnetic sensor. Note the three axis of X, Y and Z. Movements can move along these axis, as well as movement of rotation. The magnetic sensor on each axis completes the nine axis. By taking the magnetic sensor out, it becomes a six-component force sensor described by Showa technology company (Showa, 2007). These six kinds of forces can detect movement in vehicle tests, man-shaped robot controlling and medical applications where human behaviours can be analysed when the human body is in standing upright.

In an introductory to sensor fusion (Elmenreich, 2014), it is explained how sensor fusion combines data from multiple sensors so it results in a more reliable end result. Motivations included in the paper for using sensor fusion include imprecision and uncertainty of the sensors. The expected advantages of using the fused data include reliability, extended spatial coverage and reduced ambiguity although the paper claims slight scepticism exists on the ability to perfect an end result. Fusion algorithms are discussed for smoothing, filtering and predicting data as well algorithms for situations when sensors may become temporarily blocked and or when data need to be reduced. It names the Kalman Filer and the Bayesian reasoning as the tools most commonly used in sensor fusion.

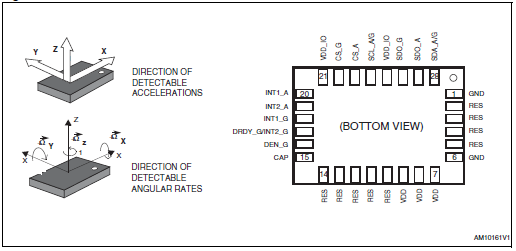
The Kalman Filter is claimed to be “one of the most important and common data fusion algorithms in used today” (Faragher, 2012). The British scientist points out that the Kalman Filter was in the Apollo computer for the Neil Armstrong moon landing and it is widely used to today in every satellite navigation device and smart phone as well as many computer games for the improvement of navigation and movement detection purposes.

### 2.2.6 Galaxy S3 Sensor Hardware



***Figure 2.2-4*** LSM330DLC device by ST Microelectronics (Dixon-Warren, 2014)

In the Galaxy S3, the accelerometer and gyroscope are combined as a 6 axis Inertial Sensor made by ST Microelectronics as discovered in a “teardown analyse” by Chipworks (Dixon-Warren, 2014). This analyses describes the LSM330DLC device, ***Figure 2.2-4***, as a three-axis accelerometer and a three-axis gyroscope combined functionality. The ST specification (Microelectronics, 2012) lists features which include a low power mode and 3 independent acceleration channels and 3 angular rate channels. The device also includes high-pass and low-past filter configuration.



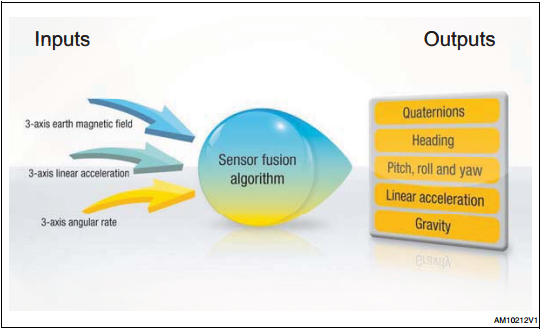
***Figure 2.2-5*** LSM330DLC Direction of Detection (Microelectronics, 2012)

The specification illustrates the direction of detection caused by the two sensors in the unit, ***Figure 2.2-5***.

The Electronic Compass in the Galaxy S3 is the AKM8975 Electronic Compass sensor in the Galaxy S3 (Chipworks, 2012). The AKM provides a datasheet that describes the senor as a 3-axis magnetometer device for use in GPS-equipped cell phones or to realize pedestrian navigation (Kasei, 2004).

The proximity sensor explained by Android (Android, Position Sensors, 2014), determines how close an object is to the front of device. There are many examples in the Android Play store where the sensor can sense hand movements being passed in front of it. The application can perform any action necessary when it detects these movements. This sensor is included in the list of sensors for the Galaxy S3.

### 2.2.7 ST iNemoEngine\_PAAP



***Figure 2.2-6*** Sensor Fusion input and output (ST, 2014)

The same company, ST Microelectronics, that provide the Accelerometer and Gyroscope device on the Galaxy S3, is also responsible for the iNemoEngine\_PAAP. It’s product page (ST, 2014), provides documents on this software library. The engine is described as advanced software that fuses data from the accelerometer, gyroscope and the magnetometer. This provides a 9-axis senor fusion as illustrated in, ***Figure 2.2-6***, which also illustrates the inputs and outputs involved. Some key features listed include motion tracking accuracy, immunity to magnetic interference, accurate direction and compatibility with Android 2.3.3+. This is all possible by using an advanced algorithm based on the Kalman filter. The engine allows correction of magnetic distortion from the magnetometer, dynamic distortion from the accelerometer and drift issues from the gyroscope. Trading between performance and power saving can be configured in the library that is easily integrated into smart consumer devices.

On researching reviews on this product, all sources have had very positive opinions. For example, an article on Digi-Key, (Mathas, 2012), reflects on how sensor fusion has gone from military to consumer use and it targets the iNemoEngine as an example of this. It confirms much of what is in the iNemoEngine documentation and describes it as a firmware software.

The EE Times (Buckley, 2011), reports on the iNEMOEngine after I was unveiled in 2011. It reports on much of the features discussed already. However, it must be said that its report is based on information from ST Microelectronics as opposed to an independent review. It quotes the company’s General Manager as saying “The iNEMOEngine delivers a quantum leap in the performance level required by next-generation smart mobile devices to enable a myriad of new exciting applications” (Buckley, 2011).

### 2.2.8 Android Sensor API

The Android Sensor API is well documented (Android, Sensors Overview, 2014). There is also a comprehensive list on the hardware with the methods and parameters that implement them under the Sensor class (Android, Sensor, 2014). To use the sensors, a class needs to implement the SensorEventListener interface. The SensorManager manages every sensor within the class while the Sensor class is used for every Sensor object created.

A table of all the Sensor types supported by Android are listed in the API guide. Some are these are hardware based, while others are both hardware and software based. These software based Sensors, or Virtual Sensors, take data from one or more hardware based Sensors.

The following list takes Sensors from the list that may be of use to this project. All of these are motion based sensors or at least facilitate in determine location and motion with other sensors.

TYPE\_ACCELEROMETER: This is a hardware-based sensor that measures acceleration force on three axes. It is commonly used for motion detection and returns values of m/s2 for x, y and z.

TYPE\_GRAVITY: This is a virtual sensor that measures force of gravity on three axes. It is commonly used for motion detection and returns values of m/s2 for x, y and z.

TYPE\_GYROSCOPE: This is a hardware-based sensors that measures rate or rotations on three axes. It is commonly used for rotation detection and returns values of rad/s for x, y and z.

TYPE\_LINEAR\_ACCELERATION: This is a virtual sensor that measures acceleration force on three axes but excludes the force of gravity. It is commonly used for monitoring acceleration on a single axis and returns values in m/s2 for x, y and z.

TYPE\_MAGNETIC\_FIELD: This is a hardware-based sensor that measures the ambient geomagnetic field for all three axes. It is commonly used for creating a compass and returns values for x, y and z.

All the above sensors are confirmed compatible for Android version 2.3+ (API Level 9).

One could assume that the virtual sensors in the Android API, for the Galaxy S3 implements the iNemoEngine software library discussed in the previous section. However, Google may have their own implementation for these software-based sensors. Further investigation will be needed to determine the implementation method of these virtual sensors to determine if any type of data filtering will be needed on the returned data. The API documentation provides advice on an implementation to test a device for which sensors, both hardware and software based, are using which particular hardware device and software firmware/library. This type of implementation may confirm if the iNemoEngine is used in the Galaxy S3.

Despite the fact that extra filtering may be needed or not, the documentation points out one issue that will need to be dealt with. This issue is offset. All values are based on the original point/position of the phone when the sensor started to detect data. According to the documentation, the offset will need to be taken from all data return to determine the true value.

### 2.2.9 The Physics of Movement

An understanding of the physics related to motion may be needed to calculate position, distance and displacement related acceleration and gravity values.

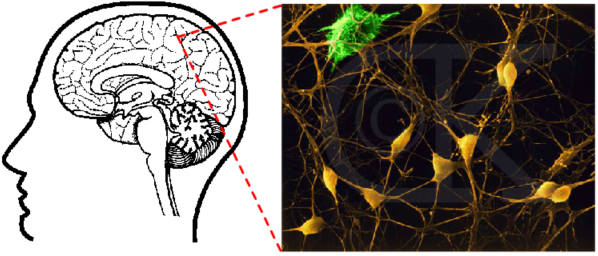
Newton’s law of Motion

Acceleration

Distance, displacement, velocity.

## 2.3 Artificial Neural Networks

### 2.3.1 Overview



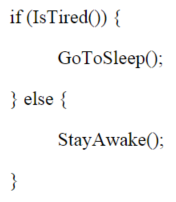
***Figure 2.3-1***A Biological Inspiration (Stanford, 2000)

The idea of an Artificial Neural Network (ANN) in computing is a biological inspiration based on how the human/animal brain processes and memorizes information (Stanford, 2000). The diagram taken from Stanford, ***Figure 2.3-1***, is used to illustrate the neural network in the brain. The yellow areas represent neurons which are connect by the lines known as the input and output channels. With about 10 billion neurons and each one connected to thousands of its neighbours, the brain can perform very complex tasks. Neurons exchange electrochemical inputs from each other and they only fire these signals if a cell body exceeds a certain level. Transmissions are formed as part of a large number of simple process unit, which fire a binary signal if the weighted sum of its inputs reaches a certain level. Though ANNs are based on this idea, they are very simplistic in comparison but they have shown to be good at problems such as image recognition and making predictions based on past knowledge.

### 2.3.2 History

ANNs dates back to the 1940s but it was the emergence of computers in the 1950s that researches came together create better networks (Bros, 2014). Physiologists, psychologists and computer engineers were among the types of researches that contributed. It was this time that the Perception neural network was formed which was based on the processing within the eye. It was from work on improving this network that the back-propagation network was created which became popular in the 1980s.

### 2.3.3 Neural Networks vs Von Neumann

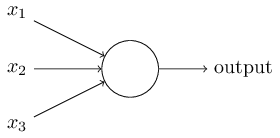


***Figure 2.3-2*** Top-down programming (Stanford, 2000)

The comparison between how the traditional von-Neumann computers and neural networks work are discussed by Neural Network Solutions (Bros, 2014) and Stanford (Stanford, 2000). The von-Neumann style is reliable in computing sequential data in defined steps such as an algorithm. The computer needs to be given in advance the details of the sequence of instructions and algorithms. Not only must the algorithm be known in advance, the data needs to be precise also. The variability of problems and data in the real world is too complex for this type of computing. Take the simple top-down approach of a simple If-Else statement, ***Figure 2.3-2***. What if the person is in an important meeting and can’t sleep? Or it is late at night but there is an assignment due in the morning? Or maybe the assignment is not that important? The number of If-Else conditions needed to satisfy these questions makes things very complex for a decision the human brain would see as simple. From this example, it is hard to see how top-down programming could ever meet the complex demands of every day decision making the human brain can deal with. In contrast, the bottom-up approach is about doing and learning by example and mistakes. The human brain is a mix of both the top-down and bottom-up approach. Some behaviours are hard-wired into the brain while other behaviours are learnt over time. Arthur Samuel’s chess machine is used as an example of how a machine cannot start off with a blank state. Certain knowledge of chess had to be given initially. Other comparisons include parallel processing compared to the limit of threads in von-Neumann computers. These computers also function in logic and rules while ANNs can function using images and concepts. Self-programming and speed (multiple processors) are other factors where ANNs can have advantages over the traditional computing methods.

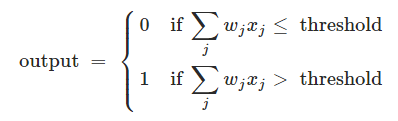
### 2.3.4 Perceptron

A perceptron is an artificial neuron that takes several binary inputs and produces a single binary output (Nielson, 2014). Nielson gives an example to explain the perceptron and how it uses weight.



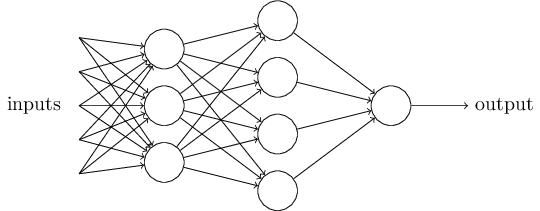
***Figure 2.3-3*** Several binary inputs, one binary output (Nielson, 2014)

A perceptron takes in several binary inputs and then computes a binary input using a weight with each input ***Figure 2.3-3***. The perceptron also has a threshold value. If the weighted sum is greater than the threshold, the output is 1 or if it’s below the threshold the output is 0. The algebraic term is as follows where x is an input and w is the weight:



***Figure 2.3-4*** Algebraic term of a perceptron (Nielson, 2014)

For a real-world example, a decision to go to a festival is used. Inputs represent factors that influences a decision to go or not. These factors include the weather, ease of public transport and availability of a friend for company. A person may be reluctantly go without the friend but the bad weather could be a complete turn off. This is where the weights come in. Here, the weather input would get a large weight while the friend input would get a small weight. The sum of the weights are measured against the threshold. This threshold can be adjust and the lower the threshold, the more willing the person is to go regardless the factors. As noted in the previous section, the model of human decision-making is far more complex. However this example shows how decisions can be made by weight up different sources of evidence.



***Figure 2.3-5*** A complex network of perceptrons (Nielson, 2014)

In ***Figure 2.3-5***, a network has different layers of perceptrons. The first layer could weigh up simple decisions and send the outputs to the next layer. The next layer could then make decisions on a more complex and abstract level compared to the first layer. In this way, having several layers of perceptrons means that more complex and sophisticated decisions are possible in a network.

Nielson gives other examples of how a perceptron network can be used such as using a bias instead of a threshold. Using weights and a threshold with a double digit binary number, a neural can act as NAND logic gate which gives opportunity for all types of functional computation in a network. However, the author points out that a neural network isn’t just another example of a circuit of NAND gates. Introducing a learning algorithm can independently tune the weights and biases within the network. These learning algorithms opens up ways of making decisions that are radically different to normal logic gates.

With these examples given by Nielson, it is clear that neural networks can make sophisticated decisions. Where it would be difficult, and even impossible, to foresee how certain problems and situations develop for a top-down programming solution, a neural network on the other hand can learn based on how a problem develops and then adjust weights and biases accordingly.

***FURTHER RESEARCH PLAN***

Sigmoid neurons

<http://www.cheshireeng.com/Neuralyst/nnbg.htm>

<http://www.princeton.edu/~stengel/MAE345Lecture19.pdf>

-Components

-Supervised (what it should have done) and unsupervised (yes or no) -- reinforced

-Applications + area that is relevant to this project

Pattern, time series, signal processing, soft sensors, anomaly detection

-Libraries and tools

Java – Implementations or implementations on motion/ sensor data.

Gesture recognition

-----

-3 layer = input – hidden - output

-feed forward

# Chapter 3: Methodology

## 3.1 Exercises

For the purpose of this project, five exercises will be used to test the ability of the Android sensors with a Neural Network.

* Lateral Raise
* Bicep Curl
* Shoulder Press
* Shrug
* Rear Fly

This exercises differ from each other in terms of motion and direction. This should allow for testing all the different ranges across the different axils. More may be added to test if similar movements can be distinguished from each other. The research provided information on the areas of strength training that are important for analgising progress. Important data that will be considered in the implementation include speed of reps, frequency of reps and break times. Data stamps could be used to aid in calculating this information. However, more complex feedback will include detecting which exercise is being executed without the user needing to tell the device.

## 3.2 Android

The Samsun Galaxy S3 will be used for this project. This device has Android Kit Kat version 4.3.1 which uses Android SDK API 18. Research has shown that this device can provide filtered data by fusing different sensors. Since this work is done in the firmware at a very low level using algorithms based on the Kalman filter, no additional filtering is necessary on the data retrieved. The TYPE\_GRAVITY virtual sensor will be used which takes data from the accelerometer and gyroscope. Data will be sent to the server over a network using a Wi-Fi connection. Data will be in the format of JSON object which can hold all the necessary data associated with the exercises executed including time, exercise name and user name. The device will also be receiving data from the server for the feedback results for each exercise.

## 3.3 The Server

The server application will be responsible for the main area of the project. The programming language of choice is Java 8 which will run on a Tomcat server. The framework Spring 4 will be used to cut down on boilerplate code in functionality including connecting to the databases. Spring Web will also be useful for providing REST calls to and from the Android application. A Mongo database will conveniently store the sensor data as JSON objects while an Oracle Database can be used to store information on the users and login information.

On test trials using the above technology, samples were sent from the phone to the Mongo database. For a 10-rep exercise, almost 2000 samples were collected. This amounted to about 250KB. This indicates that for a three-set exercise, data could accumulate up to 1MB but should not exceed it, unless very high reps are carried out.

For detecting the exercise motion, a neural network will be implemented. The sensor data will be fed to the network for supervised learning. Once the network has learned an exercise, it should detect future exercises by comparing the new data with the trained data. Different algorithms will be tested to determine which is the most suitable for this task. The sensor type of the Android may change during the course of the project to determine the best possible method of collecting and analysing data for feedback to the user.

# Chapter 4: Design

## 4.1 Vision Document

### 4.1.1 Scope

The aim of this project is to design and implement an Android application that will provide tracking and analysis for resistance training programs. It will provide feedback to the user during and after each session of training. Feedback will include an analysis of the user’s exercises, a comparison to their previous workout and suggestions on how to progress in future workouts.

The application will have a login system so it can track each user. When a user is logged in, previous and current plans can be viewed and new plans can be created. A user will also have the option to start a gym session from the current plan. The phone will use its sensors to track each exercise and send the data to a server. The application will also present the feedback it receives from the server to inform the user of their progress and suggested future progression. The application will also provide an option to add a new exercise where the user can record movements.

The server application will have the responsibility of processing the gym plan and exercise data. It will use a neural network to learn new exercise data and compare them to the current exercises being carried out by the user. It will need to determine exercise behaviours of the user such as which exercise was carried out and how many repetitions were completed. It will also analysis other data such as speed, form and time gaps between each exercise.

It is aimed, through the analysis and interpretation of the server application, that the Android can receive reliable feedback for presentation to the user.

### 4.1.2 Risk Analysis - Overview

As this study looks into implementing technologies in new ways based on research on relevant technologies and of other related implementations, certain risks will exist throughout its lifecycle. One of the main risks will be attempting to read the data from the phone sensors and interpreting it into an accurate result. The sensors will need a certain level of accuracy for this to be successful. Using a neural network to analyse, learn and compare data will need to be successful enough to allow for relevant feedback to the user. If the network cannot learn effectively, the feedback may return inaccurate data which could influence the user to make incorrect decisions in the progress of their gym plan. Other risks include the ability of sending large data over a wireless connection, comfort level of strapping a phone to the user’s arm and the ability of the phone to communicate effectively to the user while training.

### 4.1.3. Risk Analysis - Full List

The following table illustrates the list of risks involved in this study.

**Low:** 1-3, **Medium:** 4-7, **High:** 8-10 **C:** Cost, **S:** Schedule, **P:** Performance

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Risk Source** | **Probability** | | | **Impact** | | | **Impact Areas** | | | **Risk Response** |
| **L** | **M** | **H** | **L** | **M** | **H** | **C** | **S** | **P** |
| 01 | Application not to user requirements | 3 |  |  |  | 5 |  |  | X | X | Review Design |
| 02 | Gym-plan data incorrect | 3 |  |  |  | 5 |  |  | X | X | Review gym research |
| 03 | Knowledge of Android |  | 7 |  |  | 5 |  |  | X | X | Android research |
| 03 | Android Sensors Accuracy |  |  | 9 |  |  | 9 |  |  | X | Data filtering |
| 04 | Android processing performance |  | 7 |  |  |  | 8 |  |  | X | Transfer more tasks to server application & reduce data rate |
| 05 | Knowledge of Java application on server | 2 |  |  | 2 |  |  |  | X | X | More research |
| 06 | Size & complexity of server application | 3 |  |  |  | 4 |  |  | X | X | Implement a framework |
| 07 | Knowledge of neural networks |  |  | 8 |  |  | 8 |  | X | X | More research |
| 08 | Neural Network performance |  | 6 |  |  |  | 8 |  | X | X | Algorithm review |
| 09 | Software bugs |  | 6 |  |  | 6 |  |  | X | X | More testing |
| 10 | Time constraint |  | 7 |  |  | 7 |  |  | X |  | Reduce exercises |
| 11 | Database performance |  | 7 |  |  | 7 |  |  |  | X | Research NoSQL databases |
| 12 | User intuitive | 4 |  |  |  | 5 |  |  |  | X | Research GUI Designs |
| 13 | Communication method of feedback | 4 |  |  |  | 6 |  |  | X | X | Research Android applications |

### 4.1.2 High Level Feature List

|  |  |
| --- | --- |
| 001 | The application shall be for an Android device |
| 002 | A user must login into their account to use the device |
| 003 | The user can use and create new gym routines |
| 004 | The user can train the application to learn new exercises |
| 005 | The application shall have a session mode for when the user is in the gym |
| 006 | The application shall read live data while the user performs an exercise |
| 007 | The user shall receive feedback from the application on their performance which will include data for form, speed, time and suggestions on how to adjust their routine according to their performance |
| 008 | The user shall have the option to share their progress on Facebook |
| 009 | The application shall provide links to web resources that inform the user on areas that need improvement |
| 010 | A diagram showing a the line of movement to compare proper and actual form of exercise shall be viewable to the user |
| 011 | A voice option shall be used to communicate to the user so they don’t have to read the screen |
| 012 | The application shall create custom music playlists for the user’s session |
| 013 | The application shall calculate calorie and weight estimations according to the user’s progress |

### 4.1.3 Prioritization of Features

MoSCoW Method Prioritization

|  |  |
| --- | --- |
|  | **MUST HAVE** |
| 001 | The application shall be for an Android device |
| 005 | The application shall have a session mode for when the user is in the gym |
| 006 | The application shall read live data while the user performs an exercise |
| 007 | The user shall receive feedback from the application on their performance which will include data for form, speed, time and suggestions on how to adjust their routine according to their performance |

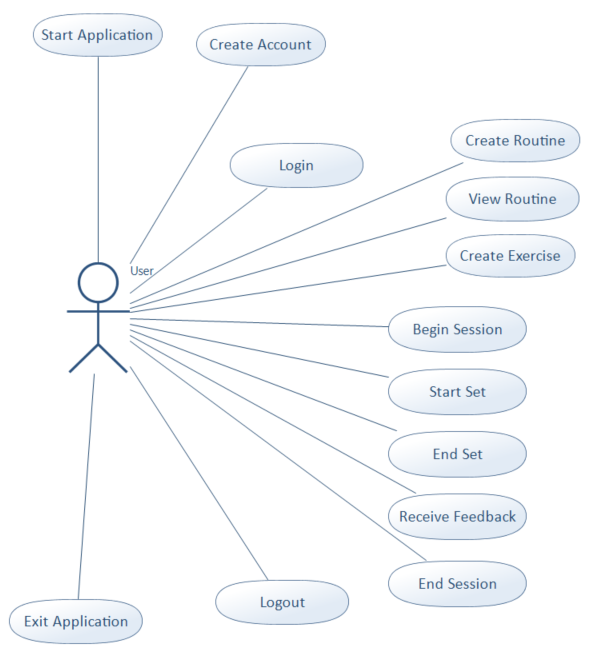
|  |  |
| --- | --- |
|  | **SHOULD HAVE** |
| 002 | A user must login into their account to use the device |
| 003 | The user can use and create new gym routines |
| 004 | The user can train the application to learn new exercises |

|  |  |
| --- | --- |
|  | **COULD HAVE** |
| 011 | A voice option shall be used to communicate to the user so they don’t have to read the screen |
| 010 | A diagram showing a the line of movement to compare proper and actual form of exercise shall be viewable to the user |
| 010 | A diagram showing a the line of movement to compare proper and actual form of exercise shall be viewable to the user |
| 013 | The application shall calculate calorie and weight estimations according to the user’s progress |

|  |  |
| --- | --- |
|  | **WON’T HAVE** |
| 009 | The application shall provide links to web resources that inform the user on areas that need improvement |
| 012 | The application shall create custom music playlists for the user’s session |

## 4.2 Functional Specification

### 4.2.1 Use-Case Diagram



***Figure 4.2-1*** Use-Case Diagram

### 4.2.2 User Stories

|  |  |
| --- | --- |
| **Use Case Name** | **Start Application** |
| Reference | 001-001 |
| Use Case Actor | User |
| Description | The application starts and presents to the user a login screen |
| Goal | As a user I want to start the application so I can use it for my gym routine |
| Acceptance | A successful launch and user is presented with a login screen |
| Flow of Events | The user press’s the application icon.  The application opens with a login screen |
| Assumptions | The application is being run on an Android smart phone |

|  |  |
| --- | --- |
| **Use Case Name** | **Create Account** |
| Reference | 002-001 |
| Use Case Actor | User |
| Description | The user provides a username and password to create an account |
| Goal | As a user I want to create a new account so my information can be saved for further retrieval |
| Acceptance | A new account is created for the user and the user is logged in |
| Flow of Events | The user enters a username and password and submits  The application will notify the user if the username has been already taken or if the submitted details are invalid. If valid the a new account is created and the user is logged in. |
| Assumptions | The user is using a secure network |

|  |  |
| --- | --- |
| **Use Case Name** | **Login** |
| Reference | 002-002 |
| Use Case Actor | User |
| Description | The user enters a username and password to login to their account |
| Goal | As a user I want to login to my account so the application knows who I am |
| Acceptance | The application recognizes if the user is valid or not |
| Flow of Events | The user enters a username and password  The application opens with a login screen if the user is valid  The application asks the user to re-attempt if the user is invalid |
| Assumptions | The user is using a secure network |

|  |  |
| --- | --- |
| **Use Case Name** | **Create new Gym Routine** |
| Reference | 003-001 |
| Use Case Actor | User |
| Description | The user creates a gym routine to be carried out over a number of weeks. |
| Goal | As a user I want to create a new gym routine so I can track my progress for my new goal |
| Acceptance | A new gym routine, according to the user’s selected options and is saved for future retrieval |
| Flow of Events | The user selects a start date. An option is then available to select the number of days to train each week. The user can then select what exercises to do each day. Finally, the user selects the amount of weeks and gives the routine a name and submits the information. |
| Assumptions | The user has some knowledge of gym planning. |

|  |  |
| --- | --- |
| **Use Case Name** | **View Gym Routine** |
| Reference | 003-002 |
| Use Case Actor | User |
| Description | The user can look at a selected gym to view progress details and edit any data if desired. |
| Goal | As a user I want to view and edit my gym routine to see and maintain my progress goal |
| Acceptance | The user sees a clear view of all the data in the routine and easily changes data that is required |
| Flow of Events | The user selects a particular routine from a list. An overview of the routine can be viewed and edited. Weeks and days can be selected for the selected routine and all the exercise data can be view for any selected day. Data can be edited and changes saved. When a day si selected, the user can start a session for that day. |
| Assumptions | The user has some knowledge of gym planning. |

|  |  |
| --- | --- |
| **Use Case Name** | **Create new Exercise** |
| Reference | 004-001 |
| Use Case Actor | User |
| Description | The user can record exercise movements for the application to recognise |
| Goal | As a user I want to record an exercise so the application can recognize for future use |
| Acceptance | The new exercise is recognised by the application |
| Flow of Events | The user executes the new exercise in perfect form. The application records the data and sends to the cloud for training. |
| Assumptions | The user carries out the exercise enough times for the application to learn |

|  |  |
| --- | --- |
| **Use Case Name** | **Begin Session** |
| Reference | 005-001 |
| Use Case Actor | User |
| Description | A user begins a session and the application goes into session mode. |
| Goal | The actor’s goal is to get the application into session mode at the commencement of a gym session |
| Acceptance | As a user I want to begin a session mode so my phone is prepared |
| Flow of Events | The user selects a begin option from the view plan area. The screen goes dark and waits for the user’s indication for the start of the next exercise set. |
| Assumptions | A full gym plan has been created and all exercises are known and trained into the application. |

|  |  |
| --- | --- |
| **Use Case Name** | **Start Set** |
| Reference | 006-001 |
| Use Case Actor | User |
| Description | The user indicates to the application that a set for an exercises is about to begin. |
| Goal | As a user I want to start a set so the phone can collect my data as I exercise |
| Acceptance | The application waits for 5 seconds and beeps to the user to begin exercises. Sensors are then turned on and data is collected. |
| Flow of Events | The user taps the phone and grabs weights. After 5 seconds of the tap, the phone beeps and the user begins exercise. The phone collects data while the user exercises. |
| Assumptions | The phone has the correct sensor hardware |

|  |  |
| --- | --- |
| **Use Case Name** | **End Set** |
| Reference | 006-002 |
| Use Case Actor | User |
| Description | The user indicates to the application that a set has completed |
| Goal | As a user I want to end a set so the phone can stop reading data and send it to the cloud |
| Acceptance | The application beeps in recognition and data is sent for analysis |
| Flow of Events | The user completes a set and taps the phone to indicate completion. The phone beeps in response. Data is sent to the server for analysis |
| Assumptions | The phone has the correct sensor hardware and data has successfully been tracked. |

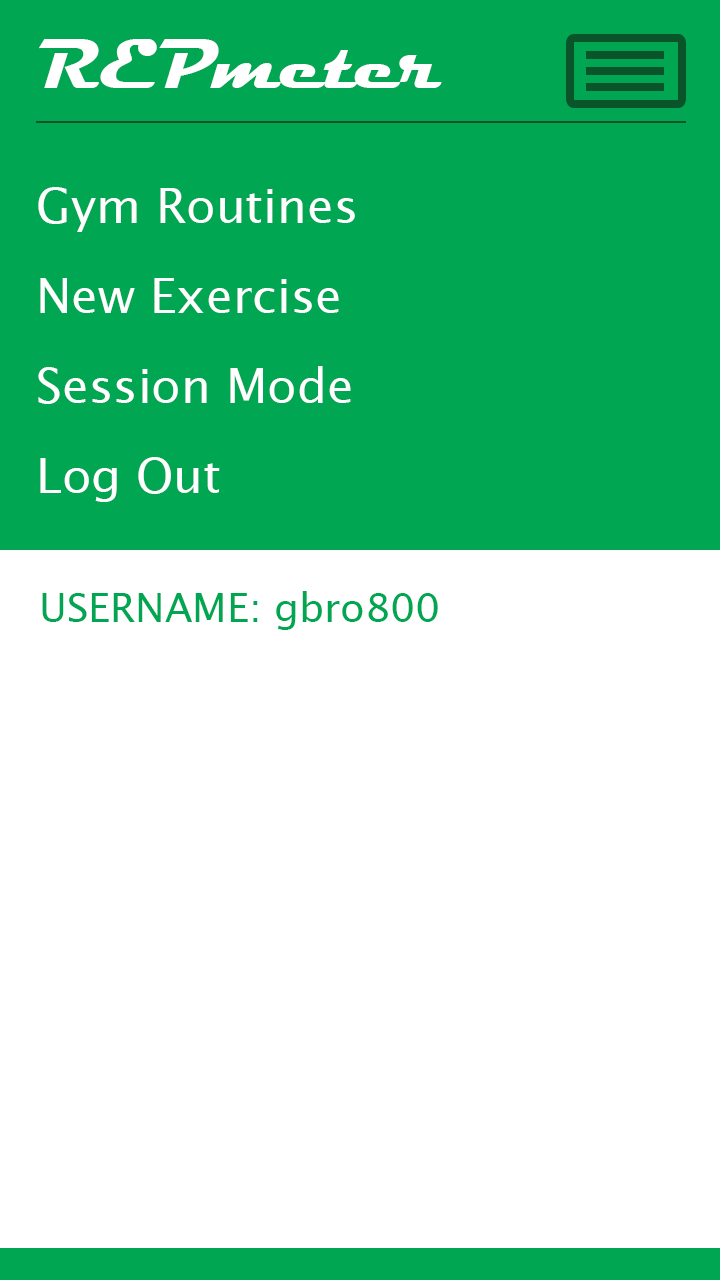
|  |  |
| --- | --- |
| **Use Case Name** | **Receive Feedback** |
| Reference | 007-001 |
| Use Case Actor | User |
| Description | The user receives feedback from a completed set |
| Goal | As a user I want to receive and review the application feedback so I can be advised as to how I am progressing |
| Acceptance | The user can view correct data of the set including the name, amount of reps and form. Gym routine is adjusted correctly. |
| Flow of Events | After completion of an exercises, data is sent to the cloud for analysis. When feedback is received, the phone beeps, the screen comes on and the user can view all feedback. The user can confirm details are correct and accept new adjustments to the routine. |
| Assumptions | The internet is capable of sending and receive data almost instantaneous. |

|  |  |
| --- | --- |
| **Use Case Name** | **End Session** |
| Reference | 005-002 |
| Use Case Actor | User |
| Description | The user indicates to the application that the session is complete. |
| Goal | As a user I want to end a session so my phone can return to normal mode |
| Acceptance | The application exits session mode. |
| Flow of Events | The user selects to end session. The application ends session mode and the screen runs as normal. |
| Assumptions |  |

|  |  |
| --- | --- |
| **Use Case Name** | **Log out** |
| Reference | 002-003 |
| Use Case Actor | User |
| Description | A user selects to log out from their account |
| Goal | As a user I want to logout of my account for security reasons or so another user can login to their account |
| Acceptance | The user is logged out and a login screen appears |
| Flow of Events | The selects the log out button and the application logs out the user. A log screen is presented to the user to log in. |
| Assumptions | The user is logged in |

|  |  |
| --- | --- |
| **Use Case Name** | **Exit application** |
| Reference | 001-002 |
| Use Case Actor | User |
| Description | The user selects to exit the application |
| Goal | As a user I want to exit the application so it is not using phone resources when not in use |
| Acceptance | The application is terminated and doesn’t use any of the phone’s resources. |
| Flow of Events | The user selects the exit option. The application requires a confirmation. After confirmation the application terminates. The user can cancel selection where exit is cancelled |
| Assumptions | The user is logged out. |

### 4.2.3 Prototype



***Figure 4.2-2*** Menu Prototype ***Figure 4.2-3*** Gym Routines Prototype

Above are prototype screenshots of the application. On the left is a demonstration of how the menu slides down. The diagram on the right shows how the Gym Routine section may look like. The use can press a routine and the weeks drop down for selection. On the bottom of the list is an option to create a new routine. An alternative to the week drop down list is to create the list in a new screen where the user could full navigate through a selected routine.

To prototype the data, the Android device was used to read sensor which was sent to the server. This data was then saved as a JSON object in a Mongo database. Some exercises recorded over 2000 data samples over 10 repetitions. This is explained in detail in Sprint 1 of the Implementation chapter.

## 4.3 System Design

### 4.3.1 Architecture

**REST API**

**ORACLE DATABASE**

**MONGO DATABASE**

**DATA LAYER**

**FUNCTIONALITY PROCESSING**

**ANDROID SCREEN LAYOUT**

**DATA PROCESSING FOR SENDING AND RETRIEVING FROM SERVER**

**APPLICATION LAYER**

**TOMCAT SERVER**

**PRESENTATION LAYER**

**ANDROID APP**

**DATABASE LAYER**

### 4.3.2 State Diagram – Gym Routine Plan

Routine Created

Weeks added

Return Frame

Weight and data adjusted

Weight and rep data added

Days added to each week

Exercises added to days

### 4.3.6 Sprints

The project will have 8 sprints. The first sprint will last one week and each sprint after that will have a period of two weeks. The final spring will last three weeks.

|  |  |
| --- | --- |
| **Sprint 1** | **Project Setup - Prototype** |
| Commence | Monday, 08-December-2014 |
| Complete | Friday, 12-December-2014 |
| Tasks | -Set up software tools and environment  -Implement and end-to-end function: Send sensor data from the phone to the server and save to database. |

|  |  |
| --- | --- |
| **Sprint 2** | **Presentation Layer** |
| Commence | Monday, 19-January-2015 |
| Complete | Friday, 30- January-2015 |
| Tasks | -Implement all screens and menus in Android application  -Add in all necessary buttons, labels and text inputs  -Add functionality to menus and any other navigation component to create proper screen sequence |

|  |  |
| --- | --- |
| **Sprint 3** | **User Accounts** |
| Commence | Monday, 02-February-2015 |
| Complete | Friday, 13-February-2015 |
| Tasks | -Set up SQL database for user accounts  -Add create account functionality to phone and server applications  -Add logging in and out to phone and server applications |

|  |  |
| --- | --- |
| **Sprint 4** | **Gym Routines** |
| Commence | Monday, 16-February-2015 |
| Complete | Friday, 26-February-2015 |
| Tasks | -Implement functionality on client and server for creating gym routines and saving them to the database  -Configure database  -Create functionality for retrieval and editing routines |
| **Sprint 5** | **Session and Data recording** |
| Commence | Monday, 02-March-2015 |
| Complete | Friday, 13-March-2015 |
| Tasks | -Implement session mode in Android application  -Implement sensor reading  -Implement functionality to send data to server  -Implement functionality on server to read data and save to Mongo Database |

|  |  |
| --- | --- |
| **Sprint 6** | **Training Neural Network** |
| Commence | Monday, 16-March-2015 |
| Complete | Friday, 27- March-2015 |
| Tasks | -Implement neural network to train data collected from client  -Create algorithms and requirements for network to learn |

|  |  |
| --- | --- |
| **Sprint 7** | **Comparing New Data against Neural Network** |
| Commence | Monday, 30-April-2015 |
| Complete | Friday, 10-April-2015 |
| Tasks | -Implement functionality to read new data and compare to trained data  -Implement Android feedback message for the user to read |

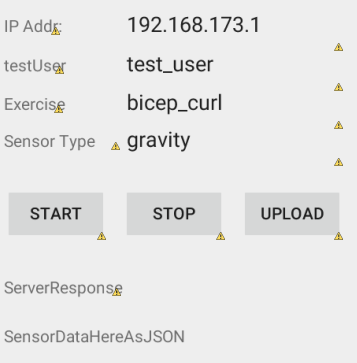
|  |  |
| --- | --- |
| **Sprint 8** | **Project review, testing and completion** |
| Commence | Monday, 13-April-2015 |
| Complete | Friday, 01-May-2015 |
| Tasks | -Review project and complete any unfinished functionality  -Test functionality and capability of application  -Complete documentation |

# Chapter 5: Implementation

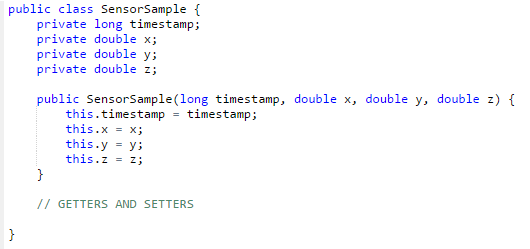
## 5.1 Sprint 1: Start-up/Prototype

The first sprint involved setting up the environment and getting an end-to-end implementation on sending data. More specifically, sensor data from the Android device was sent to the server and saved on a database. All Android and Server programming was done in Eclipse IDE.

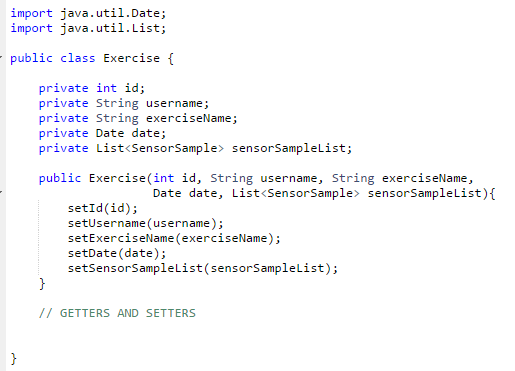
First a design is created using labels and buttons.



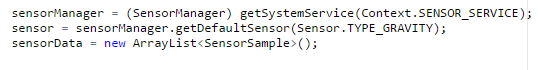
A custom SensorSample class was created to hold the data of each sample recorded. This object is instantiated and data added on every sensor change which could be as rapid as every 10 milliseconds.



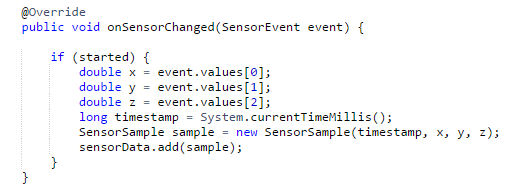
A custom Exercise class was created to hold data for an exercise with a List of all the Sensor samples. The following snippet shows the class header and its attributes.



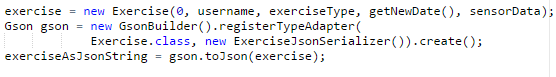
Just one Android Activity class was used for this sprint. This class extends Activity and implements SensorEventListener and OnClickListener. A Sensor Manager class is instantiated to manager the sensor service. A Sensor class was instantiated with TYPE\_GRAVITY sensor which is a virtual fused sensor.



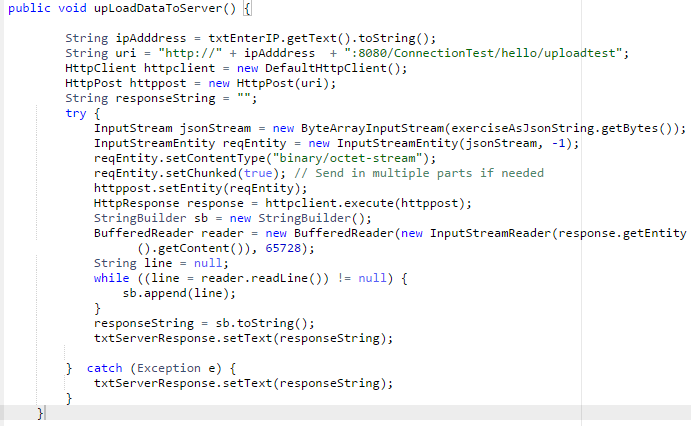
The following snippet is the onSensorChanged method which needs to be implemented as part of the SensorEventListener interface. This executes for every new sensor change. Data values and a timestamp are assigned to variables and used to create the sample object which is then added to the List of samples. The started condition was to check if the user had pressed the start/stop button on the device.



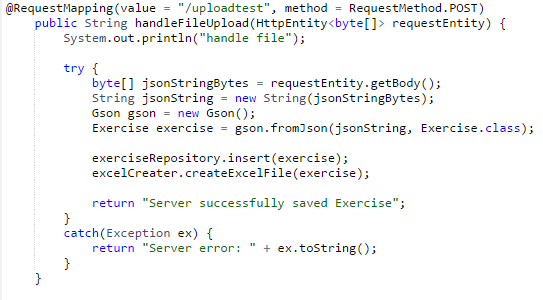
On sensor stop, the exercise object is created with information from the text fields and the sensorData List. Now the Exercise object is created, it is converted to a JSON String using the Gson library for transfer to the server.



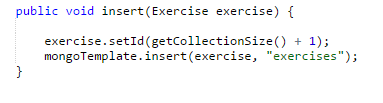
The most complex of code in this sprint involved sending the JSON data to the server. The Apache HttpClient was used to send the data over the network.



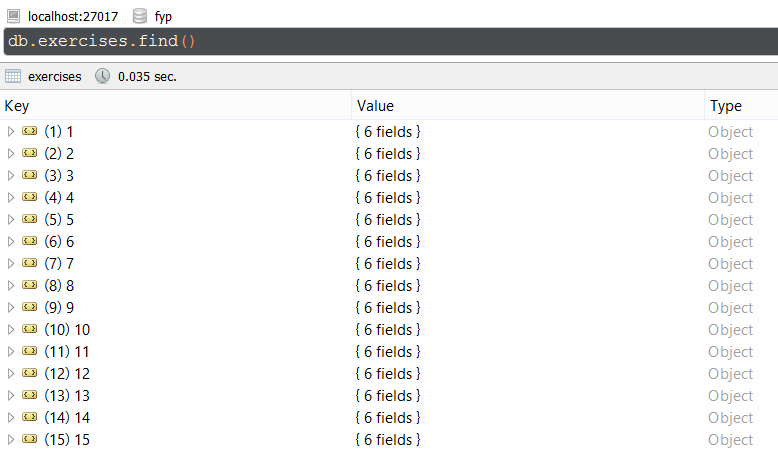
On the server, a java application with Spring framework processes all the data. Here is the method in the REST layer that receives the data from the Android device. Here the JSON object in byte[] is streamed into a String. Using Gson, the data from the JSON string is transferred into an Exercise object. Now the data can be inserted into the database with the exerciseRepository object and the data is saved as an Excel file.



The Exercise Repository is a Spring class customized to save the Exercise class to the Mongo database. This also creates the \_id which increments start at 1.



Robomongo, a tool for viewing data in a Mongo database, was used to view the raw data generated by the Android.



The final stage in demonstrating end to end data transfer shows the Android data saved into the database. This is \_id 11, a lateral raise by test\_user with the data and time shown. With this Exercise, there are 1081 samples of data in the array.

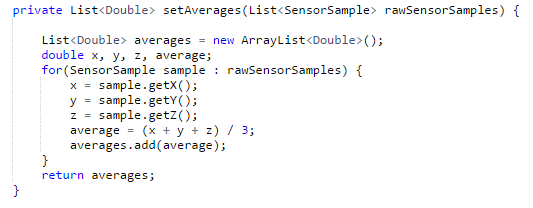


## 5.2 Sprint 2: Pre-Processing of Data

This Sprint involved pre-processing the raw sensor data that was taken from the Android device. This pre-processing was necessary so the Artificial Neural Network could read in the data successfully. For the data to be ready for the network, all values would need to be between the range of 1 and 0. Also, inputs would have to be of a set amount which means each dataset from the different exercises would always have to be the same size.

Since an exercise set could involve any number of reps, datasets would greatly vary in size. For this reason, it was decided to extract each rep from a set and send each rep to the neural network to be learned individually. This didn’t solve the size problem completely since each rep could vary in size also, depending on the exercise, the user and the speed of execution. However, it would be easier adjust each rep to a standard size compared to a whole set which could involve any random number of reps from three or four and right up to twenty and even more.

The first step of the pre-processing stage involved blending the raw data from the three axis into one array. A simple solution of getting the average value from the three axis values was chosen.



***Figure 5.2-1*** Method to find the average of each sample

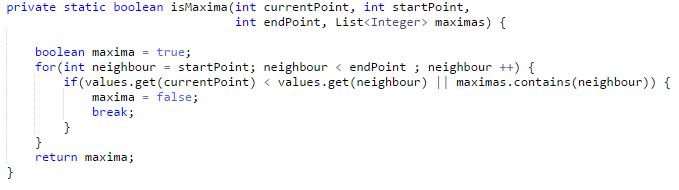
In ***Figure 5.2-1***, the method for getting the average values is shown. It takes in a List of the SensorSamples and then loops through it to and calculates the average. Each average is added to a List of Double and which is returned when complete.

***Figure 5.2-2*** Raw data of X, Y & Z

***Figure 5.2-3*** Single List of Averages

The results of this Sprint were saved to an Excel file and the above Excel charts show how the raw data of 15 bicep curls was taken in by the setAverages() method and a new List of Doubles was created. Note how the raw data values ranged from about -10 to 10 but when averaged, this was reduced from about -7 to 5. The most obvious explanation to this would be the lesser extreme values of Series 1 (blue) influenced the higher values of the other two axis to reduce when averaged. Though the extreme values had been averaged, the new set of data still showed 15 clear and similar patterns which could lead to a successful learning outcome for the neural network.

The next step was to find a way to extract each Rep from the data. The chart of Averages shows that each rep consists of one peak (maxima) and a lower points at the start and end (minima). First each maxima would need to be detected.

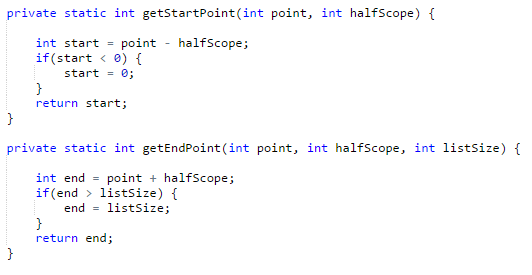
******

***Figure 5.2-4*** Maxima

To detect a maxima, each point in the Average List had to be checked and compared to its neighbouring points. If it had the highest value within a certain range of points, it was deemed a maxima. After some trial and error, a range of 100 sample points always gave a successful peak detection. The isMaxima() method in ***Figure 5.2-4*** is called for each point in the Average List. It first assumes the current point is a maxima. With a range of 100, it will loop the 50 previous points and the 50 proceeding points from the current one. If it finds that any of its 100 neighbouring points has a higher value, then current point cannot be a maxima. Another condition is that if a maxima has already been detected within the range, the current one cannot be a maxima either. This condition was included for a situation where a peak had two neighbouring high points of the same value.

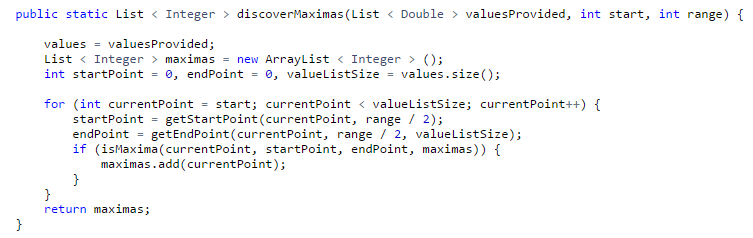
|  |  |
| --- | --- |
|  |  |
| ***Figure 5.2-5*** Narrow Range | ***Figure 5.2-6*** Wide Range |

Figures ***Figure 5.2-5*** and ***Figure 5.2-6*** shows what could occur when the range is not set correctly. If the range is too narrow, small insignificant peaks are detected while if the range is too wide, some genuine peaks are skipped.



***Figure 5.2-7*** Finding the start and end points

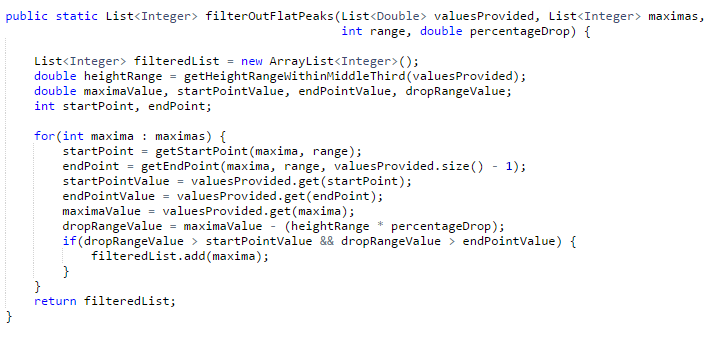
Other issues included getting out-of-bounds errors towards the start and end of the Average list. In the beginning, if it tries to loop through 50 of the previous points from the current point, it could go into minus values if the current point is below 50. Likewise, towards the end, if it tries to loop through 50 of the points proceeding the current point, it would go out of bounds if there were less than 50 points remaining in the Average list. The methods in ***Figure 5.2-7*** solved these issues.



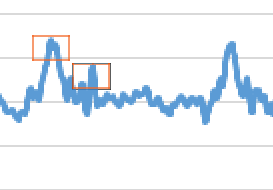
***Figure 5.2-8*** Finding the start and end points

The discoverMaximas() ***Figure 5.2-8*** was the method used to loop through each point in the Average list. It takes in the list of doubles (Average list), a starting point which could be 0, and a range. The range was set to a variable as the best range wasn’t known until after trial and error. This method calls on the get Start/End point methods and then uses these values when calling the isMaxima() method. All maxima points are added to a List and returned.

***Figure 5.2-9*** Different size peaks

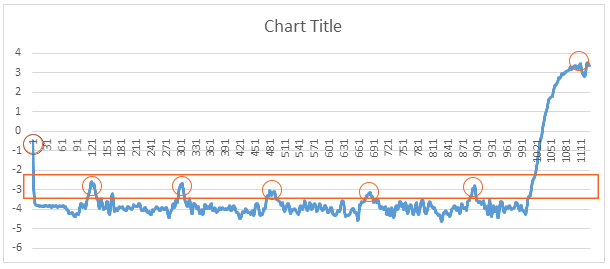


***Figure 5.2-10*** Filtering out flatter peaks code



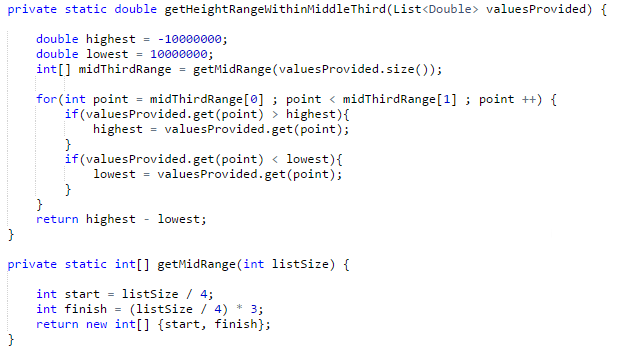
***Figure 5.2-11*** Filtering out flatter peaks illustration

Certain exercises such as the shoulder shrug in ***Figure 5.2-9*** show that many peaks of different sizes can form in the data. Many unwanted peaks could be detected in this scenario so a method filterOutFlatPeaks ***Figure 5.2-10*** was used to filter out these. A genuine maxima would rise clearly above all the others so by moving a certain distance to the right and left, and then down a certain distance, the location should still be above the data line. The second peak in ***Figure 5.2-11*** fails this test since there is other data within this range, and so is filtered out of the maxima list. Instead of actually filtering out each maxima, a new list is created where each genuine maxima is added and the list is returned by the method.



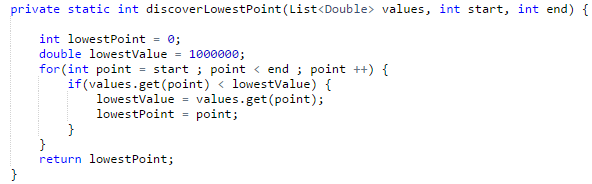
***Figure 5.2-12*** Filtering out by a certain height range

The chart in ***Figure 5.2-12*** illustrates how a dataset could have a peak detected at the very start and end of the chart. This can be caused by random movement for a few seconds before and after the execution of the exercise. Since this is random movement, extreme value ranges are recorded in the sensor data and peaks may be detected within these areas. To solve this problem, peaks outside 30% of the height range were filtered out. Using a percentage of the height was a problem since extreme heights were recorded at the start and end. Instead the height within the interquartile range was used where the highest and lowest values were found. 30% gave the best results, although it depended on the exercise also.



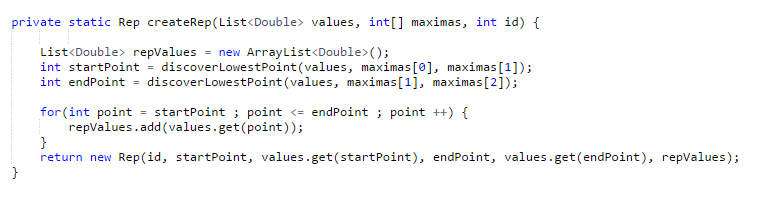
***Figure 5.2-12*** The Interquartile Range (originally used for the middle third)

The screen shot in ***Figure 5.2-12*** shows how the height within the interquartile range was got. Originally the middle third was used as the name of the method suggests, however this was eventually changed to the interquartile range. This was easily changed in the getMidRange() method, dividing by 4 got a quarter (originally 3 to get a third). By getting the mid-range points, the loop can now use these values to loop within the interquartile range.



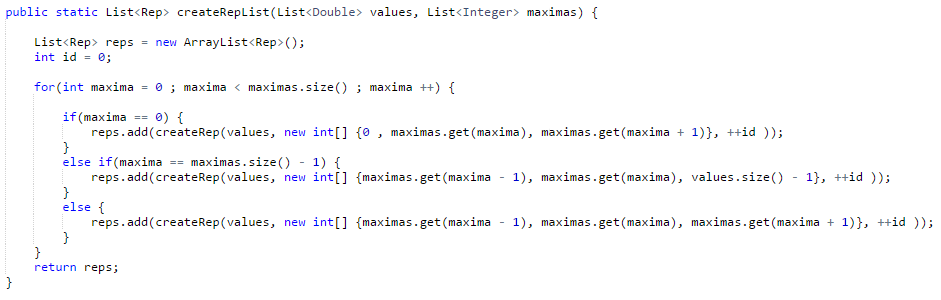
***Figure 5.2-13*** Discovering the lowest points (Minima)

All the static methods shown so far were saved into PeakDetect.java, a custom made utility class. These methods were created so a handler could call on them to find each maxima for extracting the reps. The method in ***Figure 5.2-13*** was taken from the next utility class, RepCreator.java. Once each maxima has been discovered, the lowest point within these points must be found to find the start and finish point of each rep. This method took in the Average list and the index numbers which represent a maxima point each. The for-loop goes through all the points within these two indexes and finds the lowest point.



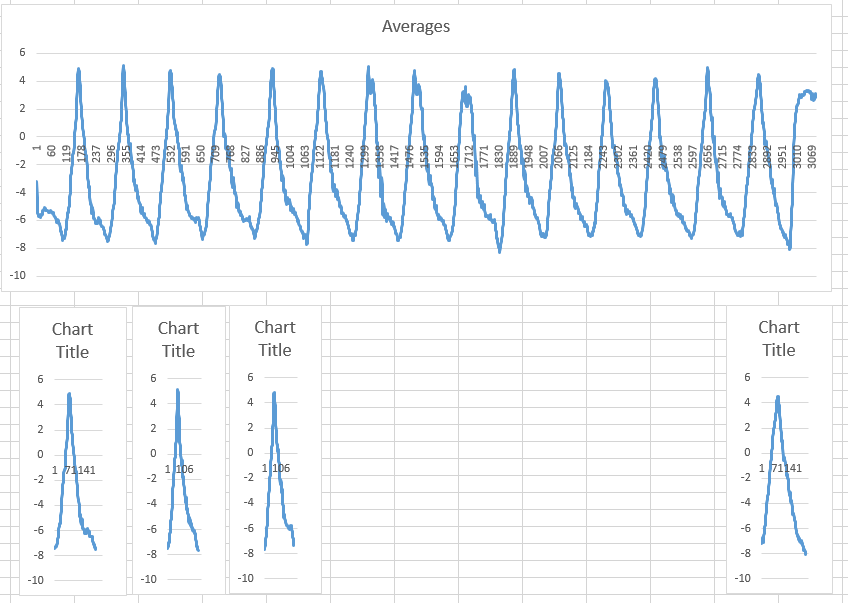
***Figure 5.2-14*** Creating a Rep

The creating Rep method in ***Figure 5.2-14*** is the method that uses the discoverLowestPoint() to find the start and end of each Rep. It then creates an List of all the values within these two lowest points.



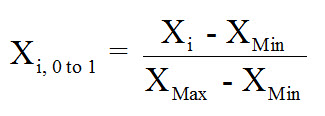
***Figure 5.2-15*** Creating a List of Rep

Finally a method is used to create a list of Rep for each exercise. It calls on the methods previously discussed. The If conditions are in place to handle the first and last reps as they measure the lowest points from the very start and end respectively.



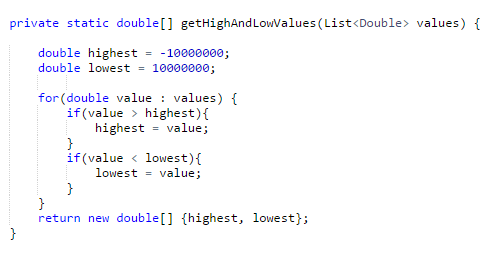
***Figure 5.2-16*** Rep Extraction Charts

To illustrate that the reps were extracted successfully, diagrams were inserted into Excel using the values from some of the Reps extracted. This was done with the 15 Bicep Curl exercise.

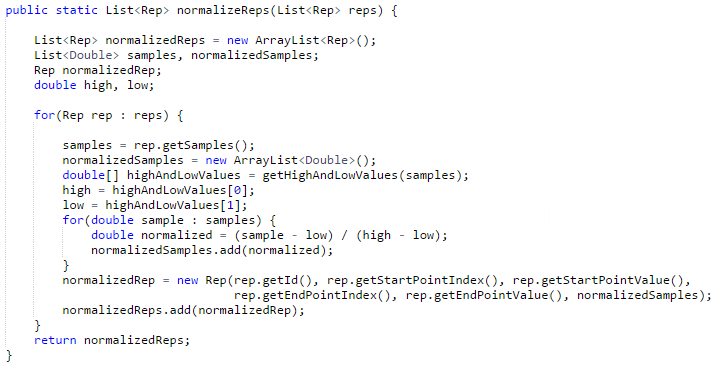


***Figure 5.2-17*** Normalization Equation (Etzkorn, 2011)

The next step in the Pre-Processing stage was to normalize the data so all values would be between 1 and 0. This was done using a normalization equation for each of the values in the Average list. The max and min values first had to be found and then the equation had to be applied.



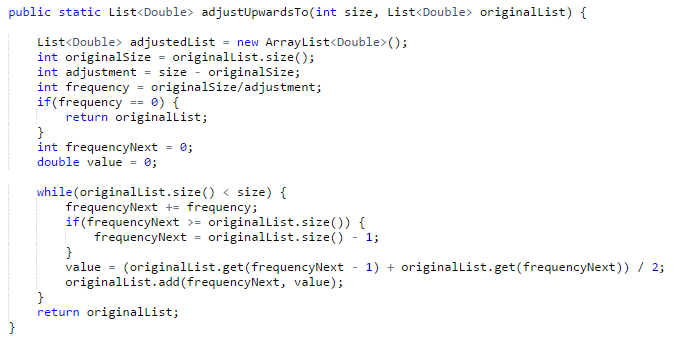
***Figure 5.2-17*** Finding the Max and Min values

The method in ***Figure 5.2-17*** shows how the mad and min values were found. An enhanced for-loop goes through the values of an extracted Rep and determines the highest and lowest values.

***Figure 5.2-18*** Creating a List of Normalized Reps

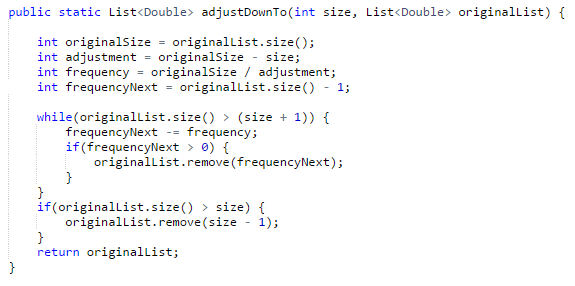
A method was used to create a List of normalized Reps. It first tooks in the List of originally created Reps and returned a normalized version of it. Once the high and low values are taken for the particular rep, the inner enhanced for-loop normalizes each value using the equation previously mentioned. A new Rep is created once the normalization is complete and added to the list.

The final step in the Pre-Processing stage was to adjust the Rep size to exactly 200 values each. For example, if a Rep had only 150 samples, 50 samples were added evenly across the list. In this example, a value would be added after every 3 samples (150 / 50). The value added would be the average of the one before and after it. Where a Rep would be 250 samples, 50 samples would be removed from the list. In this instance, every fifth sample would be removed (250 / 5).



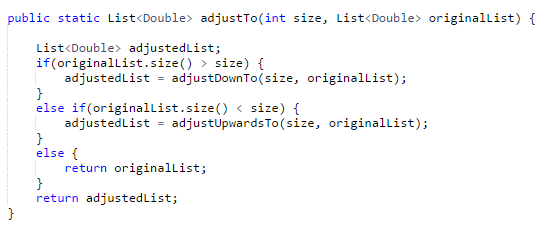
***Figure 5.2-19*** Adjusting a list up to 200

The method in ***Figure 5.2-19*** shows how a list smaller than 200 samples would be adjusted up to 200. The original list being the normalized rep of original size. The frequency represents how often a value is added (i.e. every 3 samples for a size of 150 as explained above). The *value* variable is the average of the samples directly before and after it. The while loop keeps looping while the original size has not yet reached the required size of 200.



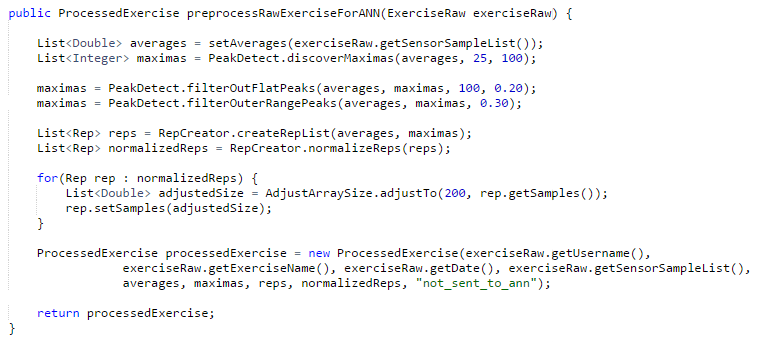
***Figure 5.2-20*** Adjusting a list down to 200

The method to adjust down was a little less complex as values were not added. It removes values while the original list is over 200. For calculating the frequencyNext, -= was used instead of a += because otherwise the frequencyNext variable would get larger as the list became smaller, causing an out of bounds error. If-Conditions were included to prevent other errors such as trying to remove a negative index value from the list. Some trial and error was needed, changing the directions of the loop, the amount of times of iteration and sometimes the loop would stop at 201 which the final If-condition would remove.



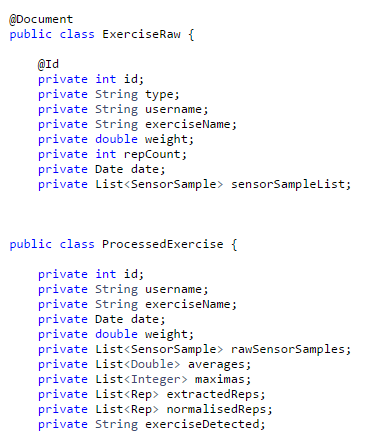
***Figure 5.2-20*** Adjust List to given size

The method in ***Figure 5.2-20*** decides if the List needs to be increased or decreased.

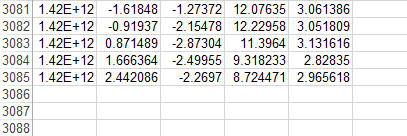
******

***Figure 5.2-21*** The Pre-Processing Handler using all the static methods

To method in ***Figure 5.2-21*** summarizes everything in this sprint into one single method. The method takes in a Raw Data object and calls all the methods discussed in this sprint. First it gets the average values, then discovers the maximas, then filters the maximas, creates the reps, normalizes the reps, adjust the size of the reps to 200. A List is created for the Reps and the normalized Reps and added to a new ProcessedExercise object. Original data is also taken from the Raw object and transferred to the Processed one.

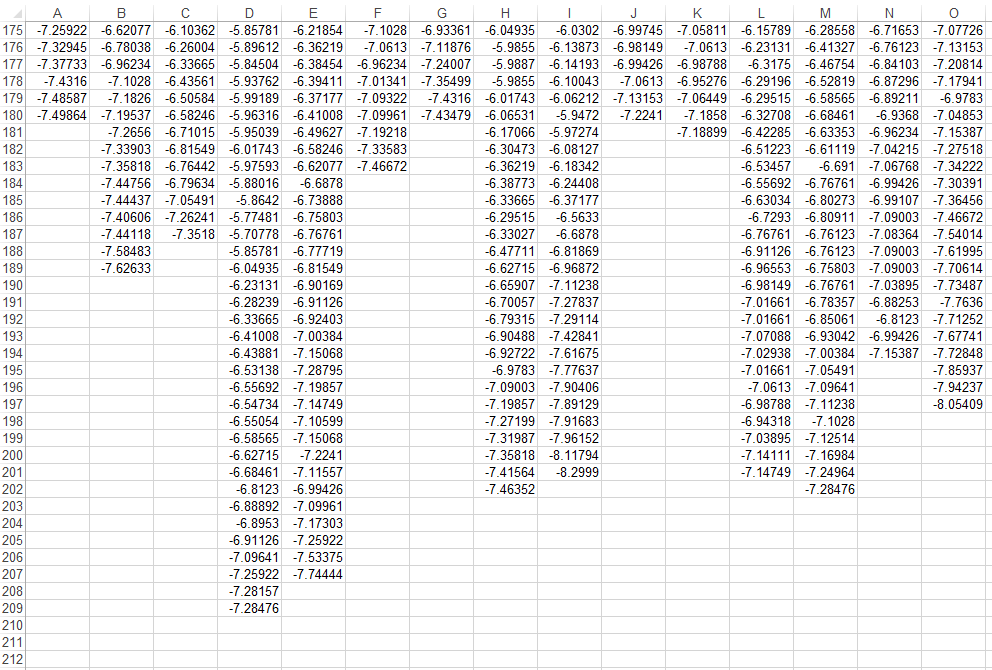


***Figure 5.2-22*** Snippets from the classes shows the extra attributes in ProcessedExercise



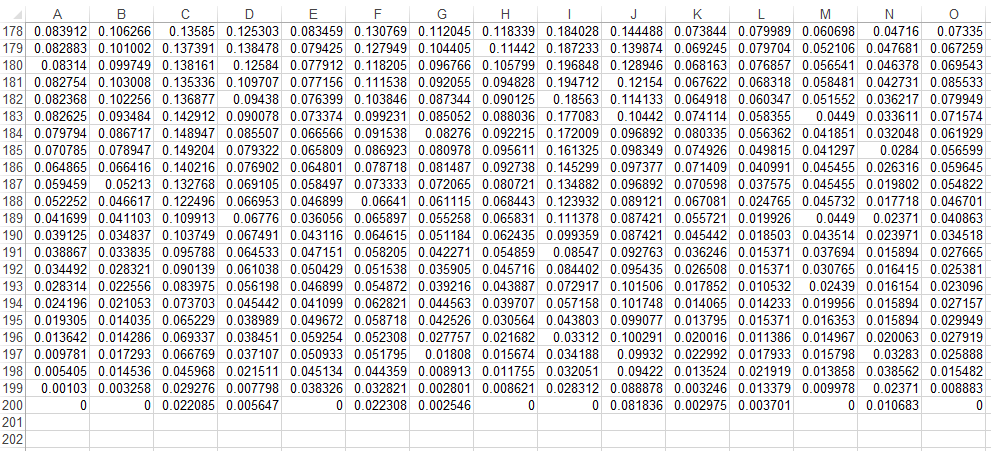
***Figure 5.2-23*** Raw data and average data

To step through the results in Excel, ***Figure 5.2-23*** shows the last five lines in the original raw data. The 16 reps come to a total of 3085 samples. The first column is the time stamp, the following three are X, Y and Z while the final one is the average value of the axis values.



***Figure 5.2-24*** Extracted Reps of different size

The Extracted Reps worksheet shows from line 175 downwards. This data shows Reps were in the range of 180 to 209 values. Since a sample is created at about one every 10ms, each Rep took between 1.8 seconds to 2.1 seconds. Since the columns go up to O, it shows that 15 Reps were correctly extracted. All values are of negative values since each rep is coming to an end (de-acceleration).



***Figure 5.2-25*** Normalized and adjusted reps

The next screenshot looks at the end of the Normalized Reps worksheet. Now all data is between 1 and 0 and every rep has exactly 200 values each.

|  |  |
| --- | --- |
|  |  |
| ***Figure 5.2-26*** Narrow Range | ***Figure 5.2-27*** Wide Range |

***Figure 5.2-26*** and ***Figure 5.2-267*** shows the difference between an original Rep and a Normalized rep. The original values went from about -7 to 5 with 180 values while the Normalized one is between 0 and 1 with 200 values. These diagrams also importantly show that the normalization had very little effect in the pattern change of the rep, despite the adjusting of the array size. Both patterns are almost identical which means the normalized rep data will enter the Neural Network representing the correct patterns.

## 5.3 Sprint 3: Artificial Neural Network

SPRINTS

1 December

2 Pre-Processing

Get averages, Peaks (detect & filter), normalize (adjust to 200), Create Rep

3 ANN

4 Android

5 Front-end Analysis

6 Front-end CSV

7 Front-end ANN & ANN trials (3-axis Reps)

8 Feedback

# Chapter 6: Findings and Analysis

## 6.1 Introduction

--

# References

*Accelerometer & Gyro Tutorial*. (2014). Retrieved from instructables.com: http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/step1/The-Accelerometer/

Altheris. (2014). *What is a gyroscope?* Retrieved from Altheris.com: http://www.altheris.com/products/what-is-a-gyroscope.htm

Android. (2014). *Position Sensors.* Retrieved from Androind Developers: http://developer.android.com/guide/topics/sensors/sensors\_position.html

Android. (2014). *Sensor.* Retrieved from Android Developers: http://developer.android.com/reference/android/hardware/Sensor.html

Android. (2014). *Sensors Overview.* Retrieved from Android Developers: http://developer.android.com/guide/topics/sensors/sensors\_overview.html

Bros, M. (2014). *Introduction to Neural Networks*. Retrieved from Neural Network Solutions: http://www.neuralnetworksolutions.com/nn/intro2.php

Buckley, P. (2011, March 24). *Next-generation iNEMO Engine offers new level of motion-sensing applications on smart consumer devices.* Retrieved from EE Times Europe: http://www.analog-eetimes.com/en/next-generation-inemo-engine-offers-new-level-of-motion-sensing-applications-on-smart-consumer-devices.html?cmp\_id=7&news\_id=222901769&page=0

Butt, C. (2001). *The WeighTrainer*. Retrieved from weightrainer.net: http://www.weightrainer.net/training/coefficients.html

Chipworks. (2012, June 1). *Inside the Samsung Galaxy SIII.* Retrieved from Chipworks: http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/inside-the-samsung-galaxy-siii/

Dixon-Warren, S. (2014). *STMicroelectronics’ Integrated Inertial Sensor MEMS Device Technology*. Retrieved from chipworks.com: http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/stmicroelectronics-integrated-inertial-sensor-mems-device-technology/

Electronics, F. (2014). *What is an Accelerometer*. Retrieved from Futureelectronics.com: http://www.futureelectronics.com/en/sensors/accelerometers.aspx

Elmenreich, W. (2014). *An Introduction to Sensor Fusion*. Retrieved from academia.edu: https://www.academia.edu/649942/An\_introduction\_to\_sensor\_fusion

Faragher, R. (2012, September). *Understanding the Basis of the Kalman Filter.* Retrieved from University of Cambridge: http://www.cl.cam.ac.uk/~rmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf

Gyroscopes.org. (2014). *Uses for gyroscopes*. Retrieved from Gyroscopes.org: http://www.gyroscopes.org/uses.asp

HyperPhysics. (2014). *Gryoscope*. Retrieved from HyperPhysics: http://hyperphysics.phy-astr.gsu.edu/hbase/gyr.html

Johnson, C. (2014, May 24). *Inertial measurement unit integrates accelerometer, gyro*. Retrieved from EE Times: http://www.eetimes.com/document.asp?doc\_id=1261819

Jovanović, M. (2014, January 13). *Set and Rep Schemes in Strength Training*. Retrieved from elsitefts: http://articles.elitefts.com/training-articles/set-and-rep-schemes-in-strength-training-part-1/

Kasei, A. (2004). *AK8975 3-axis electronic compass IC.* Retrieved from Asahi Kasei Microdevices Corporation: http://www.akm.com/akm/en/product/datasheet1/?partno=AK8975

Levi, J. (2011, October 19). *What Can You Do With a Barometer on a Smartphone?* Retrieved from Pocket Now: http://pocketnow.com/android/what-can-you-do-with-a-barometer-on-a-smartphone

Mann, B. (2014). *Hot Topic Power and Bar Velocity Measuring Devices*. Retrieved from NSCA: https://www.nsca.com/Education/Articles/Hot-Topic-Power-and-Bar-Velocity-Measuring-Devices/

Materko, W. (2007, Febuary). *Scielo.br.* Retrieved from Scielo.br: http://www.scielo.br/pdf/rbme/v13n1/en\_07.pdf

Mathas, C. (2012, 04 12). *Senser Fusion: The Basics.* Retrieved from Digikey: http://www.digikey.com/en/articles/techzone/2012/apr/sensor-fusion-the-basics

Microelectronics, S. (2012, September). *LSM330DLC iNEMO inertial module: 3D accelerometer and 3D gyroscope.* Retrieved from ST.com: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037200.pdf

Museum, C. S. (2014). *Background Information for Science Surprises*. Retrieved from Canada Science and Technology Museum: http://www.sciencetech.technomuses.ca/english/schoolzone/Info\_Science.cfm#5

Nielson, M. A. (2014). *Nueral Networks.* Determination Press. Retrieved from neuralnetworksanddeeplearning.com: http://neuralnetworksanddeeplearning.com/chap1.html

Omega. (2014). *What are Accelerometers.* Retrieved from Omega.com: http://www.omega.com/prodinfo/accelerometers.html

Sailing, M. P. (2014). *Velocitek Shift*. Retrieved from Mauri Pro Sailing: http://www.mauriprosailing.com/us/tech/Velocitek/2-Velocitek-Shift.html

Samsung. (2014). *Samsung Galaxy S3*. Retrieved from Samsung,com: http://www.samsung.com/global/galaxys3/specifications.html

Showa. (2007). *Applicaitons of 6-Component force sensor*. Retrieved from Showa Measuring Instruments Inc: http://www.showa-sokki.co.jp/english/technical\_note\_e/6\_bnryoku\_cell\_ap\_e.html

Specifier, E. (2013, July 8). *Micro Motion for Medical Devices*. Retrieved from Electric Specifier: http://www.electronicspecifier.com/sensors/es-design-magazine-analog-devices-adi-micro-motion-for-medical-devices

ST. (2014). *iNEMOEngine\_PAAPPAAP engine PRO - Android Platform.* Retrieved from st.com: http://www.st.com/web/catalog/tools/FM147/CL1818/SC1528/PF252504

Stanford. (2000). *Biological Neurons*. Retrieved from The Intellectual Excitement of Computer Science: http://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Biology/index.html

Stone, M. H. (2000, June). *Training Principles: Evaluation of Modes.* Retrieved from arturmonteiro.com.br: http://www.arturmonteiro.com.br/wp-content/uploads/2014/04/Artigo-1.9.pdf

Tatton-Brown, M. (2012, September 6). *Accelerometers are the most under-appreciated technological innovation*. Retrieved from Wired.co.uk: http://www.wired.co.uk/news/archive/2012-09/06/accelerometers-are-awesome

Tuthill, M. (2014). *Classic Muscle Strength Programs Reviewed*. Retrieved from muscleandfitness.com: http://www.muscleandfitness.com/workouts/workout-tips/classic-muscle-strength-programs-reviewed

Weil, R. (2014, May 30). *Resistance Training*. Retrieved from Emedicine Health: http://www.emedicinehealth.com/strength\_training/article\_em.htm