UNIVERSITY OF EDINBURGH

SCHOOL OF INFORMATICS

PDIOT Coursework (2017-18)

TITLE:

DESIGN AND IMPLEMENTATION OF A WIRELESS STEP TRACKER BASED ON INERTIAL MEASUREMENT UNIT WITH INTERFACE TO AN ANDROID APPLICATION

BY

CHUDALU EZENWAFOR

S1777982

DECEMBER 2017

ABSTRACT

The scope and applications of Internet of Things (IoT) is broad, and has lead to a lot of researches and findings. As technology advances, its applications also get advanced. This has allowed the growth and importance of Internet of Things to be recognised. A lot of research has been conducted on the use of these technologies to maintain human heath, finances, facilities and so on. This report gives a glimpse of the application of IoT systems used for monitoring health status and daily activities of the users. This report presents the steps and measures that were taken to develop a step tracker that uses a wearable sensor and is interfaced with a developed android application for user interface. The step tracker should be able to accurately track the steps of its user when walking or running on levelled surfaces, or gradient surfaces (like the staircase) and deduct the height displaced with the use of the atmospheric pressure sensor. This step tracker system was designed with a nRF52-DK development kit, a motion tracking sensor device (MPU-9250) with 3-axis accelerometer and gyroscope (an IMU breakout board), atmospheric pressure sensor (BMP280) and a newly developed android application. The step tracking process composes of two sub processes. The first sub-process is handled by the hardware – the IMU breakout board is placed at any position on the calf or foot, and it takes readings from the 3-axis accelerometer and 3-axis gyroscope. These readings are 6 streams of sensor values which are inputs to the nRF52-DK development kit. These inputs are processed by a developed "Covariance algorithm" to derive a single stream output. The second sub-process is handled in the android application, which is connected to the nRF52-DK board via Bluetooth. The nRF52-DK board output is taken as its input and processed by a step detection algorithm to recognise and count the steps encountered. These counted steps are relayed in real-time to the display of the application. The user is able to see the steps being counted and also view information deducted in real-time from the steps taken. An optimal placement of the IMU breakout board should be on the calf, changing the placement results in some slight changes in the number of steps counted. As a result, a step tracker that has a detection accuracy of over 90% for any walking mode or instance is developed. The performance degrades when user runs due to the step detecting method of the step detection algorithm.

ACKNOWLEDGEMENT

I want to thank my partner Zoe Petard for making it possible for us to complete the coursework by designing an awesome application and assisting a lot in the development of the step tracking algorithm used, while I was able to program the hardware and develop the algorithms used.

TABLE OF CONTENTS

1.0. INTRODUCTION

The Internet of Things (IoT) is a growing field that is advancing as technology advances. There are varieties of views on the scope, meaning or applications of IoT, which has enabled various research and experimentation on the use of better/new technologies in the IoT field. Internet of Things can be conveyed as a network of manufactured objects or devices that are connected with a specially utilised network, or to the internet or cloud to communicate so as to result to an advantageous use of these manufactured objects/devices and for the enhancement of the well-being of man and society. The applications of IoT systems are seen in the development of smart cities, smart homes, smart accessories, health-monitoring systems, smart transport systems, etc. that communicate together with the cloud/internet or private networks. According to [1], the Internet of things idea has been discussed for decades, and the first internet-connected toaster was unveiled at a conference in 1989. Due to technological advancements, various useful sensors, devices, and embedded systems have been manufactured. This has improved the advancements in Internet of Things. Now we can see the influence of IoT in various industries and markets. This course – Principles and Design of IoT systems, introduced the idea of Internet of Things and its implementations, security etc. and provided an opportunity for us to learn and work on an application of IoT for health/body monitoring by conducting a project to design a step-tracker that is interfaced to an android application, and allows its user to know how many steps he/she has taken in real-time.

This project was aimed at developing a Step-tracker that could track the steps of its user in real-time, while walking or running either on levelled surfaces or gradient surfaces. The step-tracker could have an additional feature that can tell the current height of the user. The step-tracker was designed with the use of an Mbed development kit (nRF52-DK), an IMU breakout board (MPU-9250) with 3-axis accelerometer and 3-axis gyroscope, and a developed android application for user interface. The nRF52-DK development kit was connected to the IMU breakout board and was programmed to calibrate and receive readings from the IMU breakout board after every 50 milli-seconds. This was the "sampling rate" used to get readings of the user's motion. An android application was developed to connect to the Mbed development kit through Bluetooth by utilising a Bluetooth connection protocol and to receive the IMU's readings from the Mbed development kit and then to store these readings into a .txt file. This file comprised of readings from the 3-axis accelerometer and 3-axis gyroscope, resulting in 6 streams of sensor readings. This file is then converted to .csv and analysed. The IMU breakout board is placed at several positions on the leg of the user, and walks are taken at these positions. A start and stop button is used in the application to begin a new walk or stop a current walk. New .txt files are created for each walk, and the .txt files are converted to .csv and then analysed. During graphical analysis, the readings are plotted against time-steps, and the z-plane gyroscope reading was discovered to somewhat accurately provide information on the steps taken by counting the peaks of the graph when the IMU breakout board was placed flat on the foot. The android application was programmed to detect every time a reading goes over a threshold, and then define it as a step by also setting flags. This method constrained the position of the IMU breakout board to be placed only on the feet, in a specific orientation (flat-faced). This was a disadvantage, and more analysis of the readings was done to improve on the orientation and placement position of the IMU breakout board. An algorithm was developed coined "Covariance algorithm" that processed the 6 streams of readings to give a single stream of readings. These readings are graphed against time-steps, where one can easily count the number of steps by counting the peaks. This means that the graph shows the steps taken by one of the leg instead of two. When the user takes 40 steps, the processed readings would have a graph that comprises of 20 peaks. This method allowed the IMU breakout board to be placed at various positions, and still provide a graph that comprises of peaks that accurately tell the number of steps taken by the leg with the IMU placement. The android application was programmed with a step-detection algorithm, which facilitates the detection and counting of steps. This algorithm detects a step by using a sliding window, that records

25 readings from the covariance algorithm. This is used to count the steps whenever a qualified peak is encountered, and whenever there are multiple peaks detected in the sliding window, it should be regarded as a single step. To detect the peak, a low peak threshold is given which has been chosen from various analysis as the lowest qualified peak that can define a step, and which might also allow multiple qualified peak readings in the sliding window. Ninety percent of the highest peak reading in the sliding window is taken as the next threshold value for qualified peaks. This threshold value changes as various qualified peak readings pass through the sliding window, which enables an adaptive detection of steps based on the type of walking motion the user makes. Considering that when the true steps taken by user is 40, but the detected steps taken from leg with IMU placement is 20, this algorithm finds a way that is variable, and based on the speed of walking/running, to display the immediate step of the second leg immediately after the leg with the IMU placement takes a step. A set minimum number of time-steps (50 milli-seconds intervals) between steps is used to guard against multiple peaks being counted as steps. The number of steps are displayed to the user in real-time.

As a result, the step-tracker accurately displays the number of step taken by the user when walking or running on a levelled or gradient surface. This was tested by walking on the stair-case and flat surfaces in buildings and outdoors. This report is organised as follows: - Chapter 2 provides a literature review of similar works. - Chapter 3 details the system descriptions and organisation, descriptions of algorithms implemented and software /firmware, and testing of the system. Chapter 4 provides the results and analysis of the systems, where this developed step tracker is compared with the other schemes presented in the reviewed works. Chapter 5 provides a conclusion of the project.

Fig. 1. *Picture showing the Step-tracker hardware placement on ankle position.*

Fig. 2. *Picture showing Step-tracker hardware placement on ankle position.*

Fig. 3. *Picture showing Step-tracker hardware placement on calf position.*

Fig. 4. *Picture showing Step-tracker hardware placement on calf position.*

2.0. LITERATURE SURVEY

Some related works that focus on the step-tracking are reviewed in this chapter. Nowadays motion sensors are embedded into smart-phones and various devices. This has allowed researchers to introduce various step detection methods based on the acceleration sensor in the smart-phone. Two works are reviewed, and their methods are compared. These works utilise different algorithm, and methods. Their results are compared in this chapter, and compared with the results from this project in chapter four.

2.1. [2] A NOVEL STEP COUNTING ALGORITHM BASED ON ACCELERATION AND GRAVITY SENSORS OF A SMART-PHONE. *By Qingchi Zeng, Biao Zhou, Changqiang Jing, Nammoon Kim and Youngok Kim.*

This paper proposed a step counting algorithm based on the 3-axis acceleration and the gravity sensors, which is used to enhance the estimation performance regardless of the position of the smart-phone and the motions of the user when walking or running. This paper aimed to enhance the estimation performances of step tracking algorithms based on the acceleration sensor in smart-phones by using the gravity sensor, such that regardless of whether the user in walking or running, and the phone is placed trouser pockets, shirt pocket, or bags etc. it should be concise and effective in counting the steps taken in real time.

2.1.1. Description of Proposed Scheme

This proposed step counting algorithm was based on 3-axis acceleration sensor which measures acceleration along the X, Y, Z axes and the gravity sensor provides 3-dimensional vector data indicating the direction and the magnitude of gravity. The gravity sensor is a software-based sensor (virtual sensor). When the user moves in steady state, the acceleration and gravity sensor values are almost the same, but changes when the smart phone is in a state of motion. The acceleration sensor is therefore used for measuring acceleration data, while the gravity sensor is used to recognise the gesture of the smart phone. A developed android application was used to collect these data for analysis.
This scheme enhances the conventional methods introduced by other researchers because in the conventional methods, the accelerometer is used to collect acceleration data. As the moving speed of the accelerometer increases the accuracy of the step detection based on the acceleration sensor degrades significantly. Besides this, when the gesture of the smart phone is changed, the acceleration data also changes significantly. To overcome these disadvantages, this proposed scheme used the acceleration sensor as well as the gravity sensor. After analysing the data, they are processed as follows:

$$s = g_x a_x + g_y a_y + g_z a_z$$

where a, represents the acceleration parameters along X,Y, and Z axis respectively, while g represents the gravity sensor parameters along 3 different axes. Thus, the relationship between the gravity components and the gravity acceleration value g, can be given as:

$g_x^2 + g^2_y + g_z^2 = g^2$

ratio of coefficient (ROC) is also defined to show the weight of each axis. This ratio of coefficient of the 3 axes continuously changes as the orientation of the phone changes. Auto calibration coefficients, which are adjusted to the ideal ratio is applied for data processing. This method of processing the raw data makes sure that the orientation of the smart-phone is accounted for. This removes the disadvantage of relying only on axis of the accelerometer. A low pass filter was also employed to reduce high frequency noise in the raw acceleration sensor data.

For the step detection, a modified peak detection scheme is presented in this paper. To detect peaks, the adjacent data are stored in memory and the differences of the value of these data are computed:

$P_n = s_{n-1} - s_{n-2}$
$P_n = s_{n-1} - s_n$

Where P_n is a flag that can be used for peak detection. If $P_{n-1} > 0$ and $P_n > 0$, S_{n-1} is a peak. If $P_{n-1} < 0$, S_{n-1} is a valley.

This paper presents a step detection scheme unlike conventional peak detection schemes using a threshold, two different thresholds are used for peak detection and valley detection, respectively. Either the peak or the valley is called as a pole point. In this proposed scheme, if two adjacent pole points are all peaks or valleys, the smaller on is abandoned. Only when the former is a peak and the following point is a valley, then there is a possibility that a new step has occurred. Time intervals between two pole points is also considered importantly.

2.1.2. Results

For the evaluation of the proposed scheme, 50 times of experiments were performed with 200 steps for the walking case. Four situations where the smart phone is placed in hand, in handbag, in trousers and in shirt pocket were considered. For the running case, 50 times of experiments were also conducted with 200 steps every time. Two situations where the smart phone is placed on hand and in trouser pocket.

This proposed scheme was compared with other methods used for step counting algorithm. These are:

s1 = using acceleration data along vertical orientation (Z-axis)
s2 = using the Root Mean Square of three axes
s3 = using the rotation matrices to transform the acceleration data from device frame to earth frame (Acceleration & Matrices)
s4 = This proposed scheme (using Acceleration and gravity sensor data)

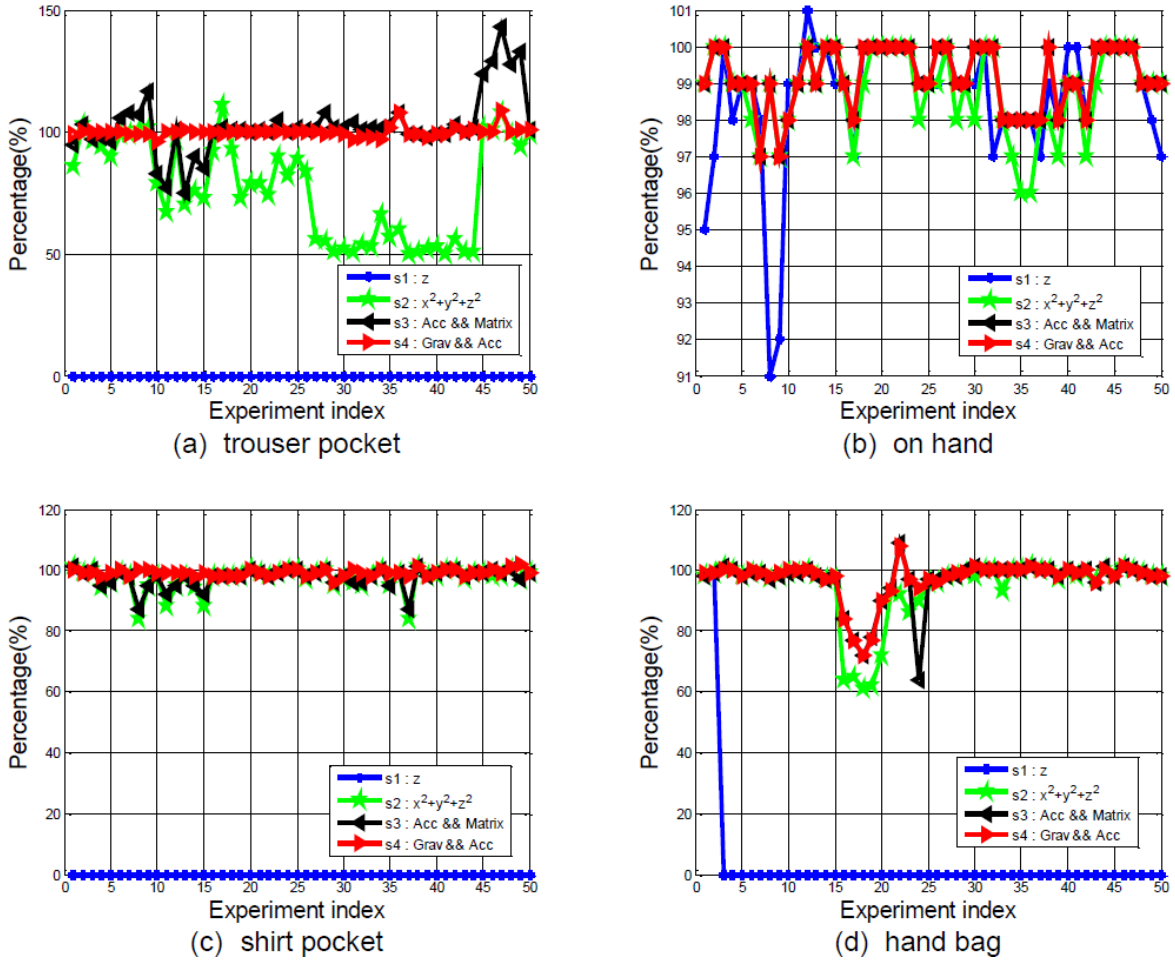
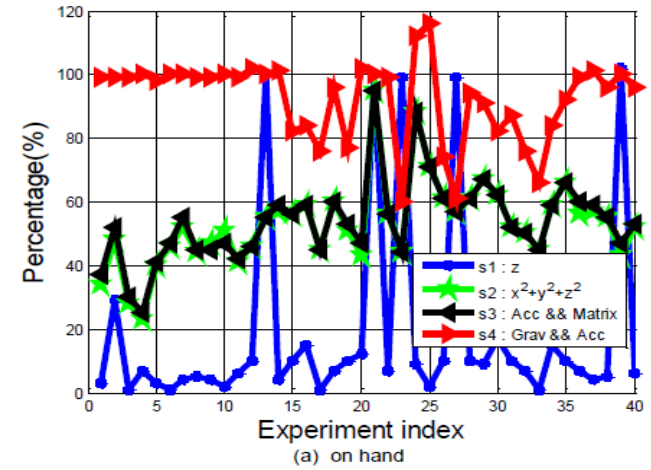
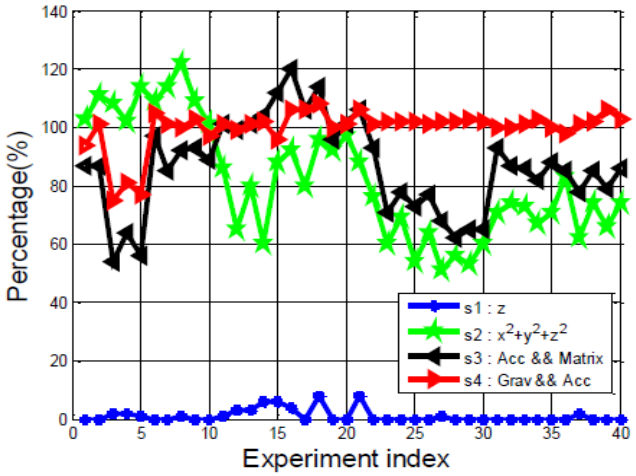


Fig. 5. *Experimental Result of Walking Cases.*



(a) on hand



(b) trouser pocket

Fig. 6. *Experimental Results of Running Cases.*

		1	2	3	4	5	Average
hand	our app	100	101	100	100	101	100.4
	S-Health	99	99	99	100	100	99.4
	Noom Walk	100	99	99	100	100	99.6
	Pedometer (Ver.1.3.7)	97	97	97	98	99	97.6
pocket	our app	99	100	99	100	99	99.4
	S-Health	100	99	97	99	98	98.6
	Noom Walk	99	100	97	100	99	99.0
	Pedometer (Ver.1.3.7)	98	97	95	97	95	96.4

Fig. 7. *The performance of different Applications*

method	walking				running	
	trouser pocket	hand	shirt pocket	handbag	trouser pocket	hand
S1: z	0	98%	0	13%	0	19%
S2: $x^2+y^2+z^2$	77%	99%	97%	94%	81%	53%
S3: Acc&Matrix	97%	99%	97%	96%	87%	54%
S4: Grav&Acc	99%	99%	99%	95%	99%	92%

Fig 8. *Comparison of Averaged Estimation Performance*

2.2. [3] A SMARTPHONE STEP COUNTER USING IMU AND MAGNETOMETER FOR NAVIGATION AND HEALTH MONITORING APPLICATIONS. *By Maan Khedr and Nasser El-Sheimy.*

This paper proposed a novel step detection algorithm using smartphones for detecting steps regardless of the device's pose, use-case, and step mode of the user (running or walking). An adaptive low-pass filter that continuously tunes the cut-off frequency is proposed, where the filter selects the appropriate cut-off frequency reducing the motion noise in the signal while preserving crucial walking information. Fusion of information from the angular rates and magnetic intensity with the acceleration norm was proposed for peak detection verification in cases of high angular rates with periodicity. All the thresholds used for detection are adaptively computed during operation to cope with step mode variation and are independent from user specific information and behaviour.

2.2.1. Description of Proposed Scheme

This proposed step detection algorithm does not require a classifier to adaptively tune the parameters for the step detection. The algorithm is based on conclusions drawn from extensive analysis of the three signals used for the step detection which are the acceleration norm, angular rates vector, and magnetic vector.
The acceleration norm is used without gravity compensation to avoid errors from the separation process and the transformation of measurements into the local level frame. Some properties of the acceleration norm were defined after the norms of acceleration in the case of a fixed device pose was studied to obtain the pattern of accelerations exerted by walking motion. These properties are:

- The acceleration norm computed from the equation below has a sinusoidal pattern where each pair of peak/valley represents a step.

$$Acc^2 = a_x^2 + a_y^2 + a_z^2$$

Where Acc is the net acceleration magnitude; and a_x, a_y, a_z are the acceleration in the body frame.

- The magnitude difference between the peak/valley pair is inversely proportional to the step duration and proportional to the motion space.
- The use of net force acceleration norm- uncompensated for gravity component, magnifies the pattern in the signal around the peaks and valleys while also smoothing it around the gravity shift component. This is due to the combined factor from both linear acceleration and gravity as shown in the equation below.

  $$Acc^2 = \text{specific force}^2 + g^2 - 2 * F_{acc} * g * \cos(phi)$$

  Where Acc is the net acceleration magnitude; specific force is the linear acceleration vector norm; g is the gravity norm; phi is the angle between gravity and linear acceleration;

- Although in many studies it is assumed that the measured norm is the root of sum of square od gravity and linear acceleration, from physics, the resulting force from both vectors is computed, as in the second equation above. Simply subtracting the gravity value from the resultant does not yield the linear acceleration where residuals from gravity remains due to the component derived from the angle between the vectors.

As for the case of angular rates measured by the gyroscope and the magnetic vector measured by the magnetometer, the resulting signals are useful in the phone dangling use-case. In this case, the patterns generated also resemble a sinusoidal wave, but each half of the signal represents a step.

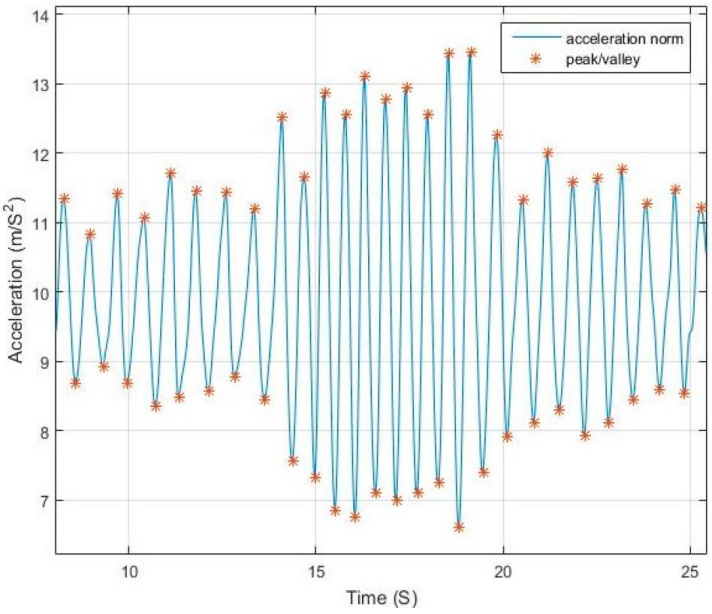


Fig. 9. *Walking with varying pace.*

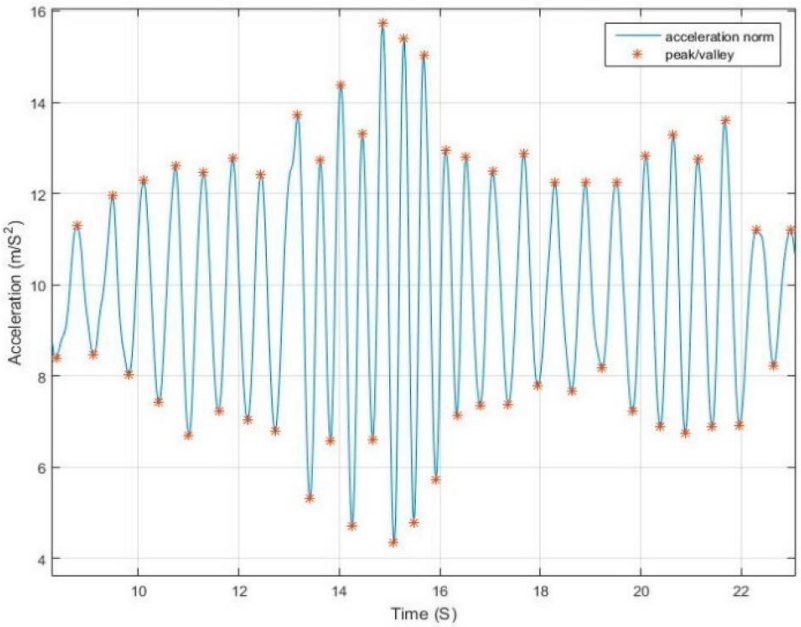


Fig. 10. *Walking and running alternation.*

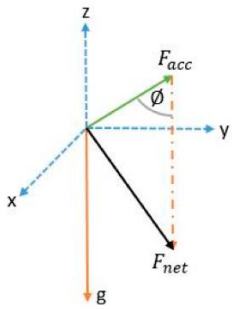


Fig. 11. *Forces applying to the accelerometer and their equivalent.*

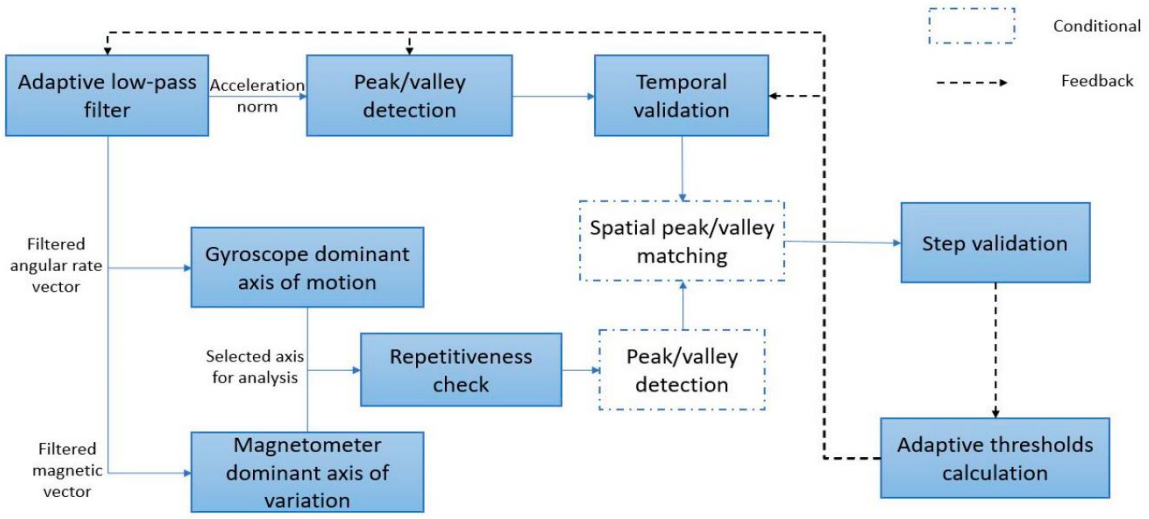


Fig. 12. *Step detection block diagram.*

The proposed algorithm as shown in figure 9, starts by filtering the measurements from the sensors using an adaptive low-pass filter. After that, it applies a peak/valley pair detection for the acceleration norm, with time filtering based on the peak-to-valley magnitude and peak-to-valley delay. Verification of peaks and valleys for cases of high device motion is also applied through further investigating the dormant axis of angular rotation and magnetic change rate, where a peal/valley extraction is also applied conditionally in the case of repetitive-high-variance patterns to the gyroscope and magnetometer measurements. The peaks and valleys of the angular velocity and magnetic field should coincide within a threshold from the peaks of the acceleration. Finally, the step is verified through the integration of the acceleration measurements within the step window. The conditional blocks are executed only in the case of the detection of a dominant repetitive signal in the gyroscope or magnetometer measurements or both.

2.2.2. Results

A group of ten users contributed to the data collection for the algorithm testing. The groups comprised of five young males and five females within ages of 21-34. Each of the test subjects carried out a walking test of 100 steps for six different phone use-cases and four walking modes summing up a total of 24 tests per user. The figure below shows the tests carried out by the test subjects.

Device Pose	Datasets			
	Step Mode			
	Walking Regular Pace	Slow Walking	Running	Combined
Compassing	10 (200 steps)	10 (200 steps)	10 (200 steps)	10 (200 steps)
Dangling	10 (200 steps)	10 (200 steps)	10 (200 steps)	10 (200 steps)
Texting	10 (200 steps)	10 (200 steps)	10 (200 steps)	10 (200 steps)
Phoning	10 (200 steps)	10 (200 steps)	10 (200 steps)	10 (200 steps)
Pocket	10 (200 steps)	10 (200 steps)	10 (200 steps)	10 (200 steps)
Free-motion	10 (200 steps)	10 (200 steps)	10 (200 steps)	10 (200 steps)

Fig. 13. *Collected dataset description.*

In addition to the data presented in the figure above, six tests were carried out where users walked 383-594 steps while changing the phone orientation, use-case, varying the walking speed, switching between walking and running, and scaling stairs. The step counts from wristbands – Fitbit Flex2 and the Xiami Mi Band 2, were sampled to be compared with the proposed implementation.

Device Pose	Detection Accuracy			
	Step Mode			
	Walking Regular Pace	Slow Walking	Running	Combined
Compassing	100%	99.85%	99.9%	99.9%
Dangling	99.85%	99.65%	99.85%	99.8%
Texting	99.9%	99.00%	99.9%	99.65%
Phoning	99.95%	99.85%	99.95%	99.9%
Pocket	100%	99.85%	99.95%	99.95%
Free-motion	99.8%	99.4%	99.75%	99.6%

Fig. 11. *Average accuracy of the proposed algorithm.*

Test	Performance		
	Proposed	Flex2	MiBand2
Test 1 (383 steps)	381–99.47%	378–98.69%	376–97.66%
Test 2 (411 steps)	409–99.51%	405–98.54%	407–99.02%
Test 3 (433 steps)	430–99.31%	426–98.38%	425–98.15%
Test 4 (485 steps)	482–99.38%	475–97.93%	473–97.52%
Test 5 (530 steps)	527–99.43%	522–98.49%	520–98.11%
Test 6 (594 steps)	592–99.66%	582–97.97%	579–97.47%
Overall (2836)	2821–99.47%	2788–98.31%	2780–98.02%

Fig. 14. *Performance evaluation in comparison to Flex2 and MiBand2.*

2.3. CONCLUSION

The first paper [2] proposed a novel step counting algorithm based on the built-in acceleration sensor, in which gravity sensors of a smartphone is also proposed to enhance the estimation performance regardless of the position of a smartphone and the motions of a pedestrian user either walking or running. The objective of this paper was achieved as seen in the results, where the proposed scheme for counting steps compared to the other conventional schemes produced a much more accurate step counts.

The second paper [3] proposed a novel step detection algorithm also using smartphones but based on acceleration and magnetometer sensor. The proposed algorithm shows high versatility with a maximum accuracy of 100% in some cases of fixed device pose and a minimum of 99.1% in the case of slow walking while texting due to screen tapping induced noise as deducted in the paper. The proposed scheme surprisingly outperforms the accuracy of the two fitness bands used for comparison while not requiring any extra hardware to be used.

The schemes employed in both papers are quite familiar, but the second paper does not use gravity compensation as used in the first paper. Both schemes make use of accelerometer data for step detection. The second paper can seem somewhat more complicated to implement than the first scheme seen in the first paper. Both schemes successfully achieve the goal of step counting regardless the pose of the smartphone, but with the proposed scheme from the second paper [3] being a slight improvement on the proposed scheme from the first paper [2].

3.0. METHODOLOGY

In this chapter, the description of the proposed step tracker system organisation and implementation is provided. The descriptions of the methods, algorithms and hardware that were used to develop the step tracker are provided and explained. A description of the software structure is provided, then descriptions of tests taken to verify the step counting algorithm and methods, and grade the performance are provided with their resulting outcomes. During the development of this step tracking system, I handled the programming of the firmware, development of the covariance algorithm and joint development of the step detection algorithm, which are all explained in detail in the sections ahead.

3.1. DESCRIPTION OF SYSTEM ORGANISATION AND IMPLEMENTATION

The description of the hardware and software organisation that forms up the proposed step tracker system is provided below. These descriptions detail the hardware and software compositions and relationship that makes up the step tracker system

3.1.1. Hardware Description

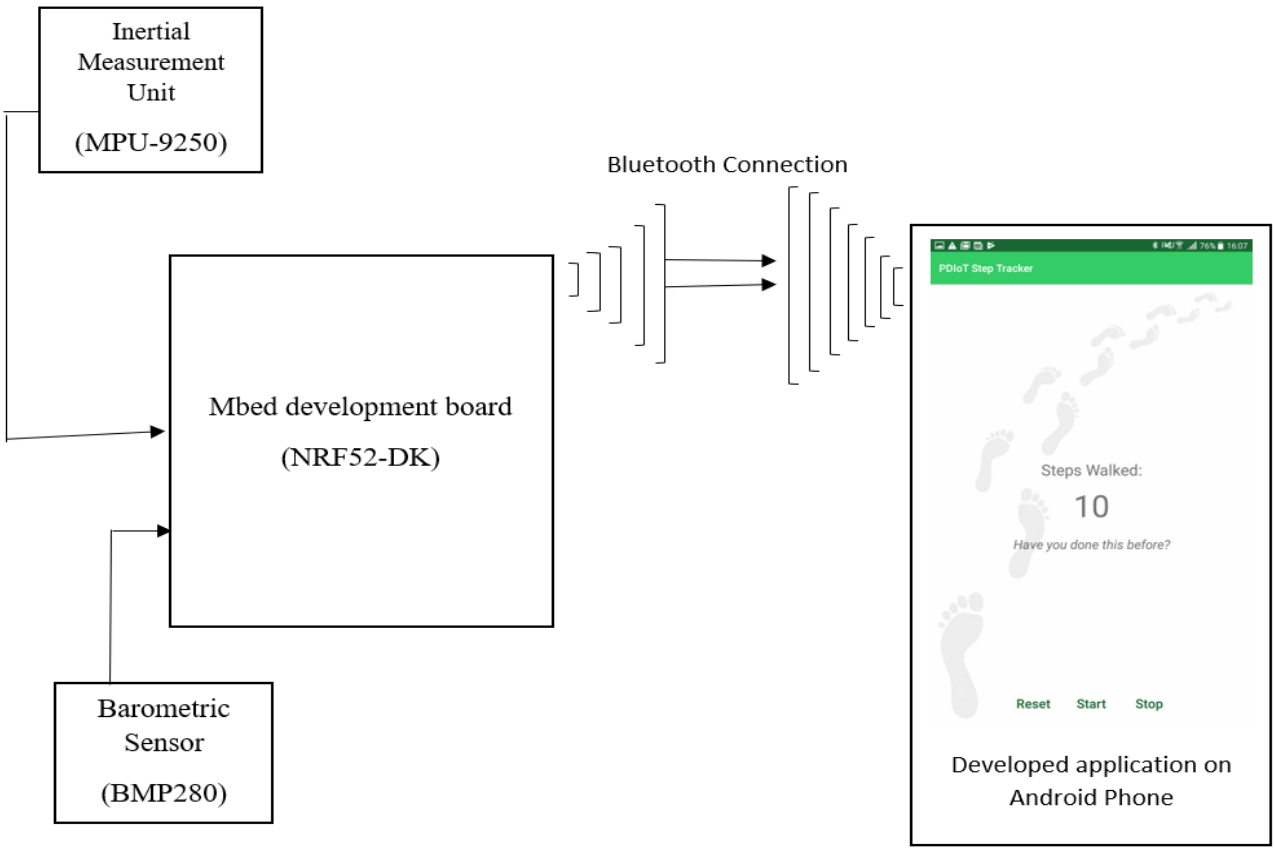


Fig. 15. *Block Diagram showing the Hardware organisation of the proposed system.*

As seen in the previous figure, the hardware composition of the system makes use of the following components listed and discussed below.

3.1.1.1. Inertial Measurement Unit (MPU-9250)

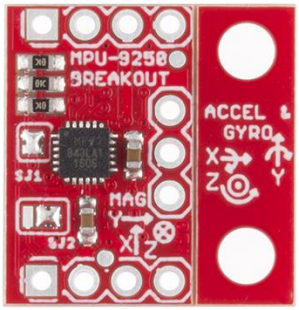


Fig. 16. *MPU-9250* [4].

According to [4], the MPU-9250 IMU breakout board features latest 9-axis MEMS sensor from InvenSense. Each of these 9DoF breakout feature an MPU-9250 with System in Package (SIP) that combines two chips: the MPU-6500, which contains a 3-axis gyroscope as well as a 3-axis accelerometer, and the AK8963, which features a 3-axis magnetometer. The MPU-9250 uses 16-bit Analog-to-Digital Converters (ADCs) for digitizing all axes, making it a very stable 9 Degrees of freedom board.

3.1.1.2. Mbed development kit (Nordic nRF52-DK)

According to [5], the nRF52-DK development kit is a single-board development kit for Bluetooth Smart, ANT and 2.4GHz proprietary applications using the nRF52 series SoC. This development kit supports ARM Mbed tool-chain for rapid prototyping and development using Mbed's cloud-based IDE and tool-chain with an extensive range of open-source software libraries. Some significant features of this board are:

- Nordic nRF52 System-on-Chip combining Bluetooth v4.2 compliant 2.4GHz multiprotocol radio, on-chip NFC tag and ARM Cortex-M4F processor on a single chip optimised for ultra-low power operation.
- All input/output and interfaces available via connectors
- Accepts power through:
  a. USB
  b. External source (1.8V-3.6V)
  c. Single 2032 coin-cell battery, onboard battery holder.
- USB drag and drop programming and USB virtual COM port for serial terminal.

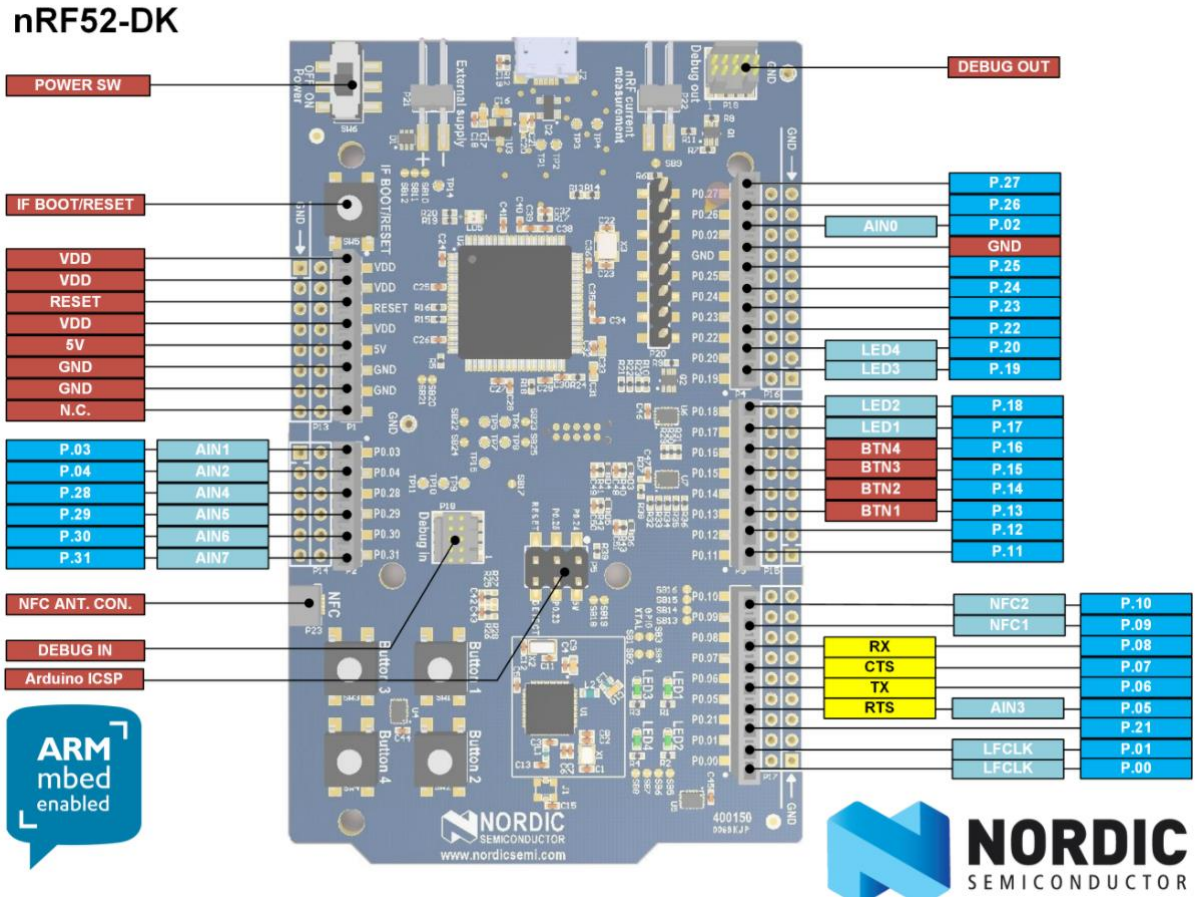


Fig. 17. *Nordic nRF52-DK* [5].

3.1.1.3. Atmospheric pressure /Barometer sensor (BMP280)

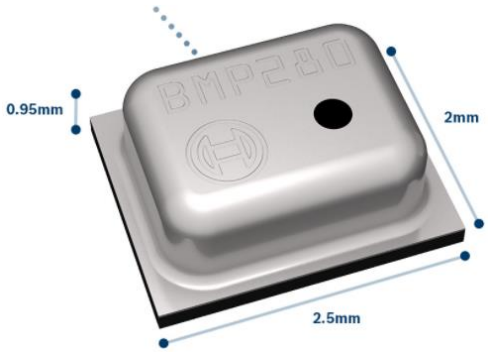


Fig. 18. *BMP280 Atmospheric pressure sensor* [6].

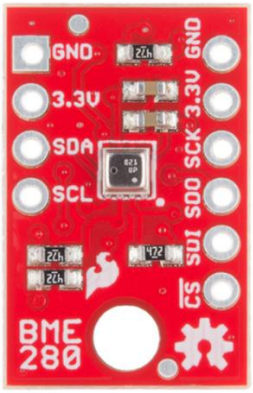


Fig. 19. *BME280 mounted on a breakout board* [7].

According to [6], BMP is an absolute barometric pressure sensor especially designed for mobile applications. The sensor module is housed in an extremely compact package. Its small dimension and its low power consumption allow for the implementation in battery powered devices.
This sensor measures atmospheric (barometric) pressure, humidity, and temperature readings. With these readings, the current height of the user and several information about the user's environment can be deducted.

3.1.1.4. Android Device

Any android device can be used to operate the developed android application. During the development and testing of the android application, a Samsung Tablet was used to operate the application. This android device should be Bluetooth enabled to allow communication to the nRF52-DK development board.

3.1.2. Firmware Description

A description of the programmes that are composed in the nRF52-DK development kit, which facilitates the operation of the step-tracking system is provided in this section. The various classes employed, and the implemented functions are listed and discussed.

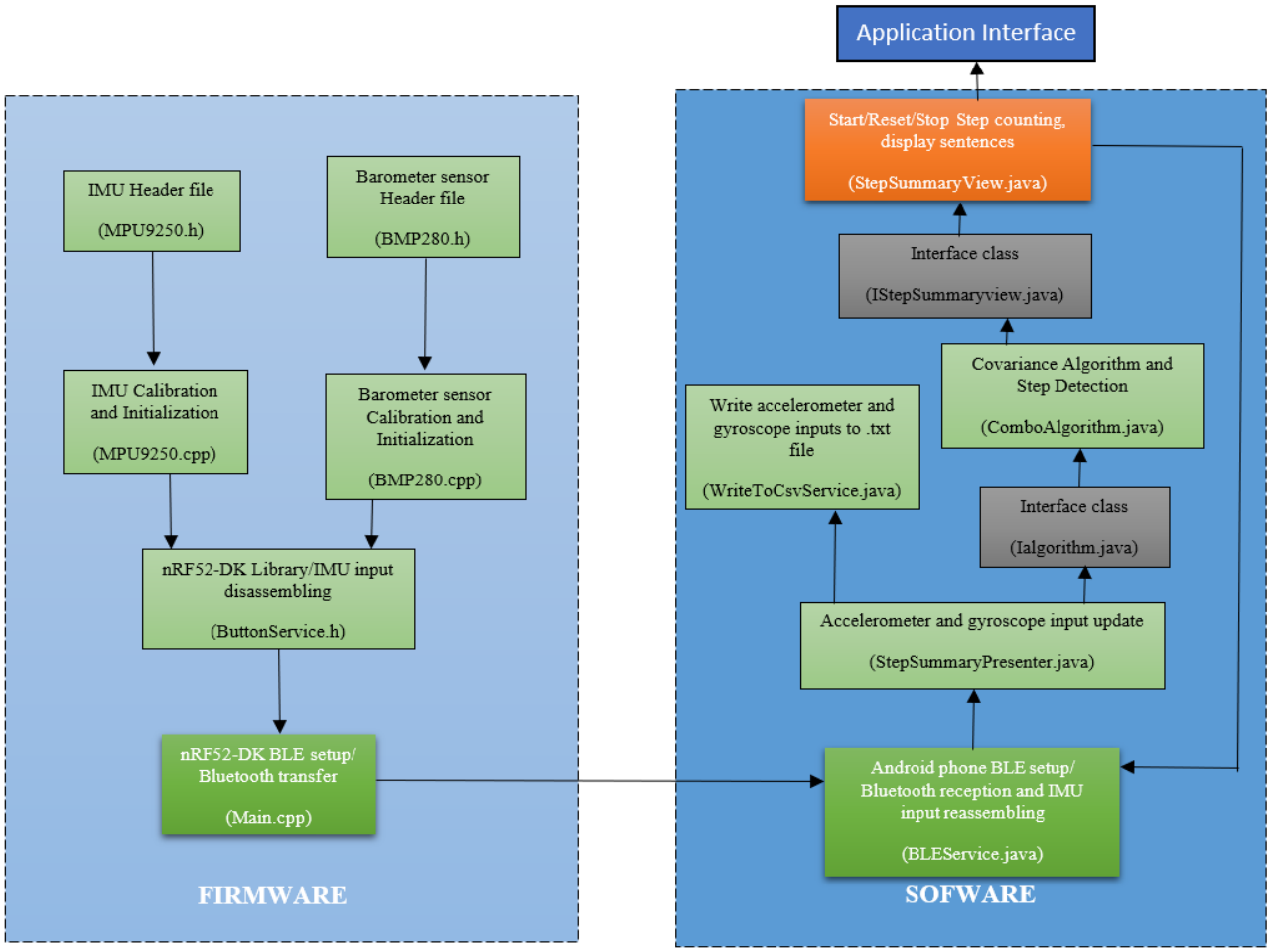


Fig. 20. *Simple Block Diagram showing the firmware and software structure.*

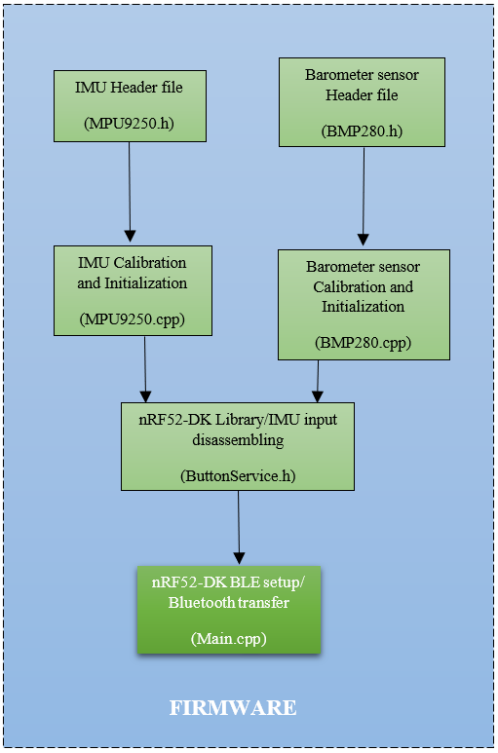


Fig. 21. *Simple Block diagram showing firmware structure.*

The firmware is the programme that runs the nRF52-DK development kit. The figure above shows a structural representation of the significant classes and libraries that make up the firmware. The firmware was developed to handle the calibration, disassembling and the Bluetooth transfer of the IMU (MPU9250) measurements to the developed android application for further processing and step detection in real time. The different classes and libraries/stages that compose the firmware are briefly discussed in this section.

3.1.2.1. IMU Header file (MPU9250.h)

This header file deals with the definitions of functions and variables, which are imported to be used in the MPU9250.cpp program. This header file was provided through the Mbed platform which allows users to access programmes of various authors.

3.1.2.2. Barometer (Barometric) sensor Header file (BMP280.h)

This header file deals with the definitions of the functions and variables, which are imported to be used in the BMP280.cpp program. This header file was provided through the Mbed platform which allows users to access programmes of various authors.

3.1.2.3. IMU calibration and Initialization (MPU9250.cpp)

This class initializes, and configures the IMU sensor to allow for accurate readings. In every system, there is error/noise that need to be compensated for. This class handles the error that are present in the accelerometer, gyroscope, and magnetometer readings. The errors are handled using filters programmed in this class. An implementation of Sebastian Madgwick's "…efficient orientation filter for… inertial/magnetic sensor arrays" which fuses acceleration, rotation rate, and magnetic moments to produce a quaternion-based estimate of the absolute device orientation, which can be converted to yaw, pitch, and roll. The performance of this orientation filter is at least as good as Kalman filter. This class file was provided through the Mbed platform which allows users to access programmes of various authors.

3.1.2.4. Barometer (Barometric) Sensor Calibration and Initialization (BMP280.cpp)

This is a BMP280 combined humidity, temperature and pressure sensor library authored by Toyomasa Watarai and accessed via the Mbed development platform. This class initializes the BMP280 and

calibrates the sensor readings. The readings are filtered to allow accurate reading of the sensor values.
The resultant pressure value is returned as output in hPa, while the temperature is returned in Fahrenheit.

3.1.2.5. nRF52-DK Library/ IMU inputs Disassembling (ButtonService.h)

This is header file is sourced from the Mbed Microcontroller Library. In this stage, the inputs from the accelerometer, gyroscope and BMP280's pressure and temperature readings, are received as 16-bit integers and disassembled into 8-bit integers which are placed in an 8-bit integer array for Bluetooth transfer.
The accelerometer and gyroscope readings come in 3-axes, which axes are the X, Y and Z axis. Each of these readings are disassembled into two parts, by dividing the 16-bit integer to upper and lower 8-bit integers. The pressure and temperature values are also disassembled into 8-bits integers. The values from the pressure and temperature readings are used to calculate the current height of the sensor relative to the pressure at sea level. This would result to a number that tells the height of the sensor from sea level.

$$h = \frac{((\frac{P_0}{P})^{\frac{1}{5.257}} - 1) \times (T + 273.15)}{0.0065}$$

Fig. 22. *Hypsometric formula for calculating height from pressure and temperature* [8].

[8] Given the current location pressure, P, pressure at sea level, P_0, and current location temperature in Celsius, T.

The resultant height value is received as a 16-bit integer, which is also disassembled into upper and lower 8-bit integers. These values are also placed in the array for Bluetooth transfer.

3.1.2.6. nRF52-DK BLE setup/Bluetooth transfer (Main.cpp)

This class is the main class of the firmware. It sets up the Bluetooth advertisement and connection with the android phone. When connection is achieved, the 8-bit integer array is sent every time at a frequency of 50 milliseconds. This is set to 50 milliseconds so as to allow accurate sampling of user's motion which enables accurate detection and counting of steps.

3.1.3. Wireless Communication

As seen in figure 15, showing the hardware description of the designed step-tracker, the communication between the IMU inputs and the step detection and counting algorithm which is implemented in the nRF52-Dk development kit and android application respectively is done with a Bluetooth connection. This wireless connection scheme was chosen because of various reasons which are discussed in this section.

3.1.3.1. Bluetooth Low Energy (BLE)

According to [9], Bluetooth Low Energy (BLE) is a wireless technology standard for personal area network. BLE is targeted for very low power devices, devices that can run on a coin cell battery for months or years.

One of the defining factor on why BLE was chosen as the means of wireless communication is that it was supported by both the nRF52-DK development board and the android application. As explained seen earlier, BLE focuses on low power devices, line the nRF52-DK which supports an alternate coin cell power supply. Over the years, progress has been made on BLE, and its applications [9] can be seen in Health care, fitness trackers, beacons, smart homes, security, entertainment, proximity sensors, industrial and automotive. The BLE ecosystem is centred around smart phones, tablets and PCs.

3.2. STEP COUNTING METHOD AND ALGORITHMS

As the IMU inputs are received at a rate of 50 milliseconds over the Bluetooth connection to the android application, they are acted upon by 2 Algorithms (processes) that derives a single stream of input, and count the detected steps from the given input. These two algorithms are discussed in this section, with their step by step implementation detailed also.

3.2.1. Covariance Algorithm

The understanding of how to calculate the variance of a dataset serves as the basic principle used in the formulation of this algorithm. Given various inputs of the IMU sensor (MPU9250) that represents the position of the user's leg or any part of the body with the IMU sensor placement during motion, inputs from the 3-axis accelerometer and 3-axis gyroscope are six streams of readings. At every interval of 50 milliseconds, new input data set are received. These datasets represent the position of the body part with sensor placement at a time. Even though the readings are different, each dataset represent the same position of the user's body part at a particular time during motion. Since the dataset represent the same position despite the different representation in X, Y and Z axis, the variance of the dataset is calculated. The variance is calculated in a special way, this is what derives the name "Covariance algorithm".
For a little understanding of some terms used in this algorithm, some simple definitions are given below:

- Mean (Empirical mean): This is an average of the values in the dataset.

  Mean, $M = (a_x + a_y + a_z + g_x + g_y + g_z)/6$

- Variance: can be defined as the difference between the expectation of a squared random variable and expectation of that random variable squared. Variance can also be defined as the expectation of the squared deviation of a random variable from its mean. It measures how far a set of values are spread out from their average value.

  Variance, $S^2 = \sum(x-\bar{x})^2/ n -1$

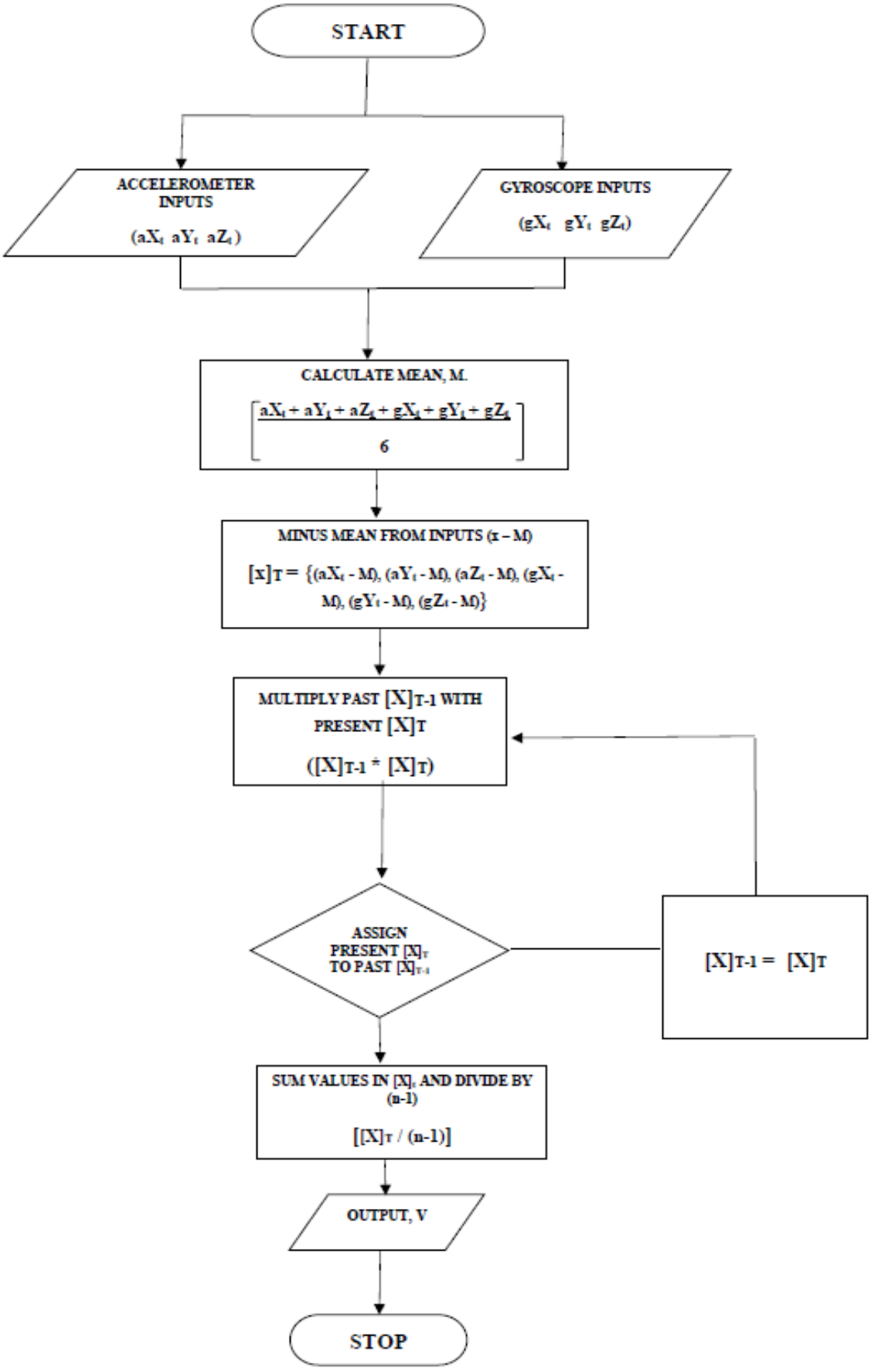


Fig. 22. *Flowchart showing the process/stages of the Covariance Algorithm.*

The Processes shown in the flowchart shows the steps that are followed in processing the six streams of input from the IMU sensor to give a single output. This output is used in the step detection algorithm to define a step and count the steps taken.

- Accelerometer and Gyroscope Inputs (aX, aY, aZ, gX, gY, gZ)
  These inputs are received via Bluetooth transfer, and are placed in an array.

- Calculate Mean, M
  The average of the inputs is taken which gives us the mean, M of the dataset.

- Minus Mean from Inputs (x – M)
  The mean is subtracted from each of the inputs. These current values are stored in the array ($[X]_T$ as shown in the flowchart)

- Multiply Past $[X]_{T-1}$ with Present $[X]_T$
  The first (current) values in the loop $[X]_T$ are multiplied with the previous values, which is $[X]_{T-1} = 0$. The first result array, [Cov] will be a zero array.
  The current array, $[X]_T$ would then be assigned to the previous array, $[X]_{T-1}$. When the next set of inputs comes in, a new current array, $[X]_T$ is calculated and multiplied with $[X]_{T-1}$. Each of the values of the same type in the arrays are multiplied respectively.
  That is;
  $(aX_{t-1} - M_{t-1}) * (aX_t – M_t)$
  $(aY_{t-1} – M_{t-1}) * (aY_t – M_t)$
  …
  $(gX_{t-1} – M_{t-1}) * (gX_t – M_t)$
  …..

  The resultant array is assigned to an array, [Cov].
  The multiplication of the past and present array calculates the variance in the current sensor readings, and the variance in the past sensor readings. Instead of squaring the differences of the current sensor readings which would result to the variance of the current data set, the differences of the current sensor readings are multiplied with the differences of the past sensor readings respectively, as these readings are taking at very close intervals to each other. The multiplication would result to a broad representation of the variance of the sensor readings, which would provide a graph that shows the details of the user's movements.

- Assign Present $[X]_T$ to Past $[X]_{T-1}$
  After the current and previous values are multiplied, the current values are assigned to the previous value. This is used in calculating for the output value, with the new current value.

- Sum Values in [Cov] and Divide by (n-1)
  Where n, is the number of values in the array, [Cov]. The sum of the values in [Cov] is divided by five, which gives us a slightly different type of variance output, hence the name "Covariance algorithm".

- Output, V
  As new IMU sensor readings are processed, this algorithm is followed, and the output is analysed in the step detection algorithm to detect the possible steps and count them.

Accel(X,Y,Z), Gyro(X,Y,Z)	Mean	(Accel - mean) (Gyro - mean)	Product of Accel) (Product of Gyro)	SUM	COVARIANCE
[1] ax1,ay1,az1, gx1,gy1,gz1	SUM[1]/6	(Each value) - mean	(ax1*ax2), (ay1*ay2).... (gx1*gx2), (gy1*gy2)...	sum[1]	sum[1]/5
[2] ax2,ay2,az2, gx2,gy2,gz2	SUM[2]/6	(Each value [2]) - mean	(ax2*ax3), (ay2*ay3), ... (gx2*gx3), (gy2*gy3),....	sum[2]	sum[2]/5
[3] ax3,ay3,az3, gx3,gy3,gz3	SUM[3]/6	(Each value [3]) - mean	(ax3*axN), (ay3*ayN), (az3 *azN) (gx3*gxN), (gy3*gyN), (gz3 *gzN)	sum[3]	sum[3]/5
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
[N] axn,ayn,azn, gxn,gyn,gzn	SUM[N]/6	(Each value[N]) - mean	.	sum[N]	sum[N]/5

Fig. 23. *Description of the Covariance algorithm.*

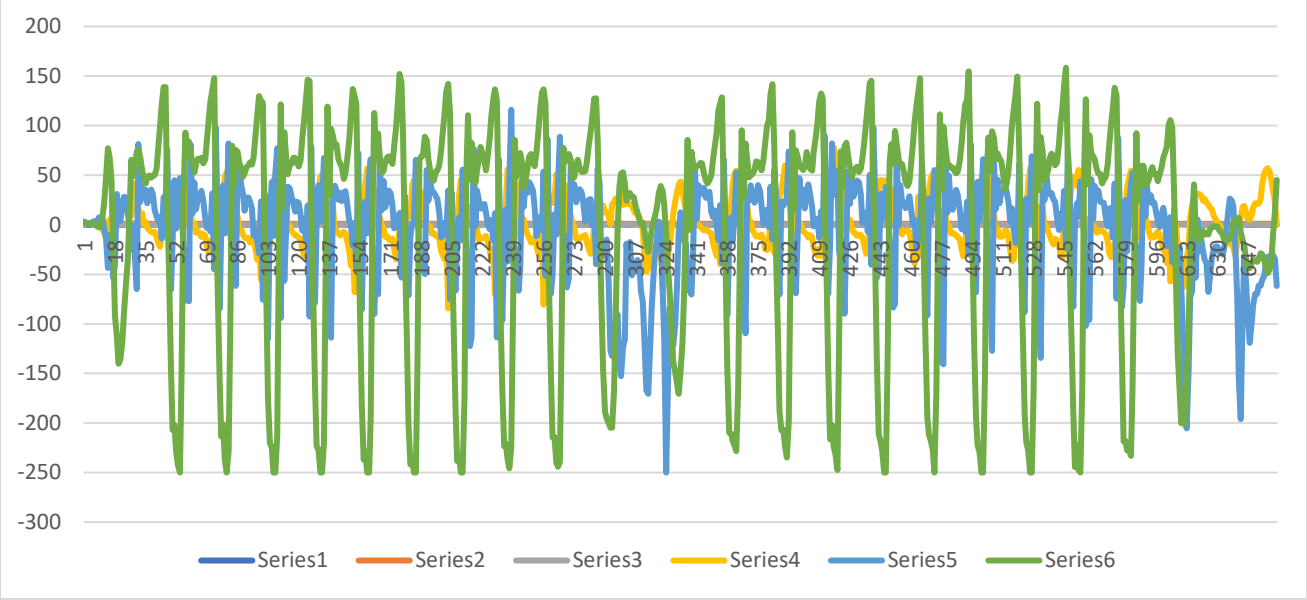


Fig. 24. *Graph showing the calf placed IMU sensor readings for 45 steps.*

The figure above presents the graphical representation of the IMU sensor inputs. As observed in the graph, only three graphs can be seen. This is because the units of the gyroscope readings are in hundreds, and the units of the accelerometer readings are in tens. The covariance algorithm actually makes use of the gyroscope readings, given that for calculating for the single stream variance output from these six streams of readings would make the accelerometer values very small and negligible. As one can observe in the graph, the rough accurate number of steps can be easily counted by counting the number of negative or positive peaks. These readings were gotten from the IMU sensor placed in a flat orientation on the foot. Changing the orientation or the position of the IMU sensor to the ankle or calf in various orientation, or changing the user would give different numbers and range positive and negative peaks, and a different graph pattern for the same 40 steps taken. The covariance algorithm solves this problem, such that the same number of peaks can be counted even though the position and orientation of the IMU sensor is changed with only some variation of at most two more or 2 less peaks than expected.

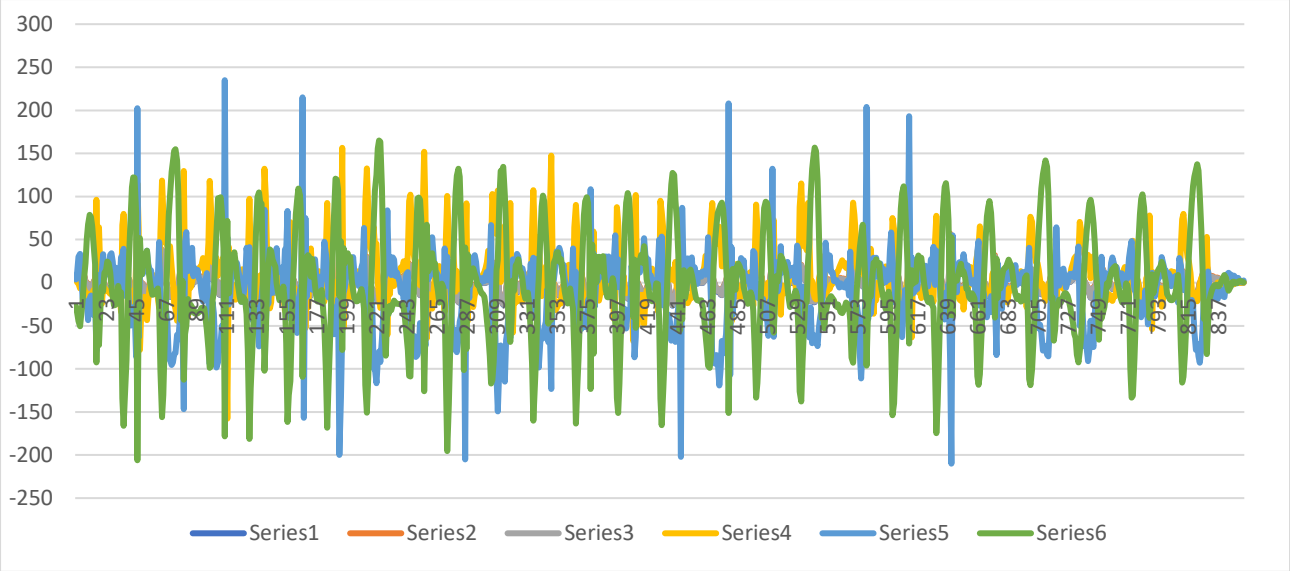


Fig. 25. *Graph showing the calf placed IMU sensor readings for 50 steps climbing upstairs.*

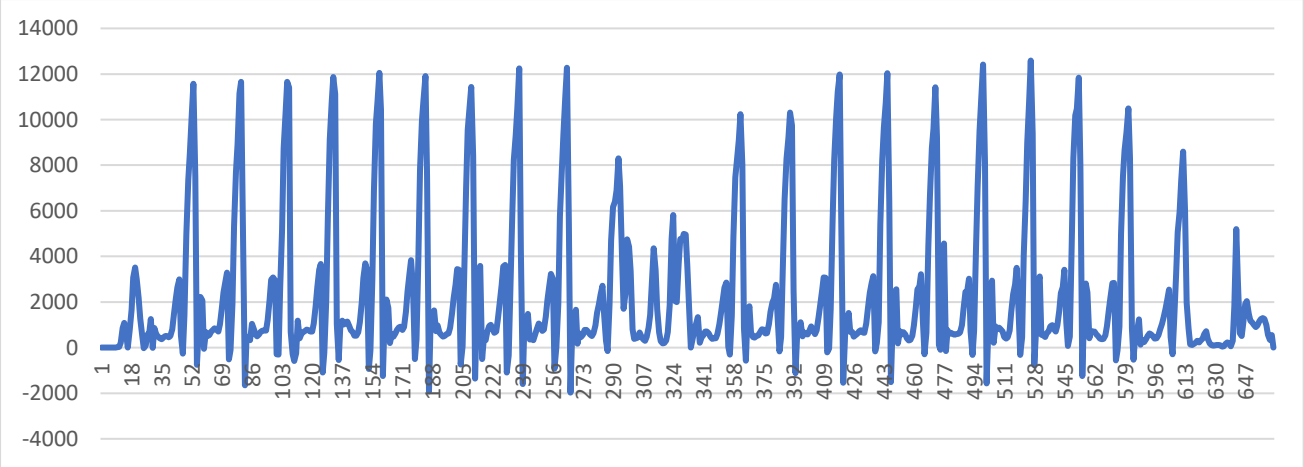


Fig. 26. *Graph showing covariance algorithm processed calf placed IMU sensor readings for 45 steps*

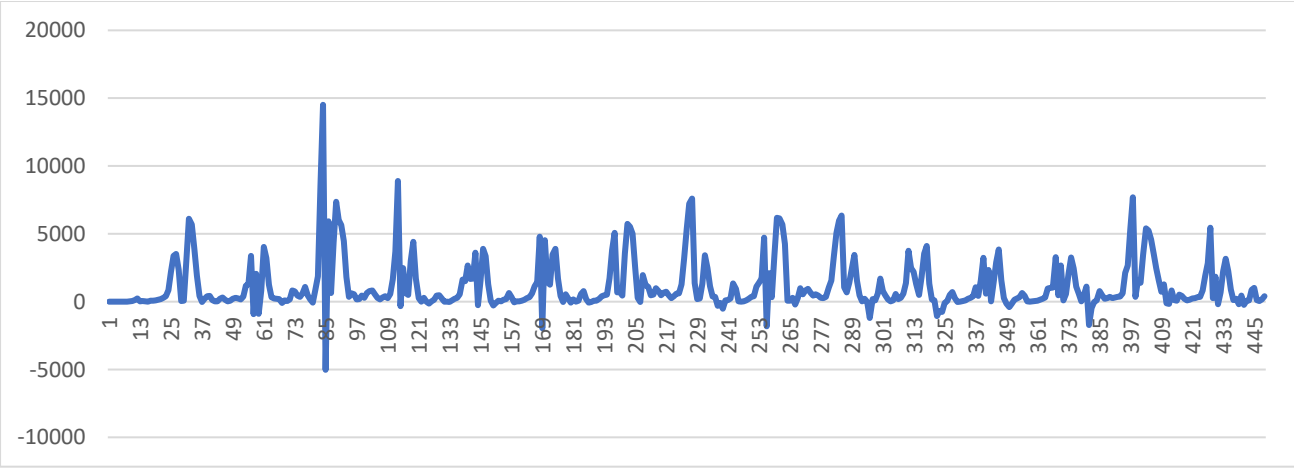


Fig. 27. *Graph showing covariance algorithm processed calf placed IMU sensor readings for 30 steps climbing upstairs*

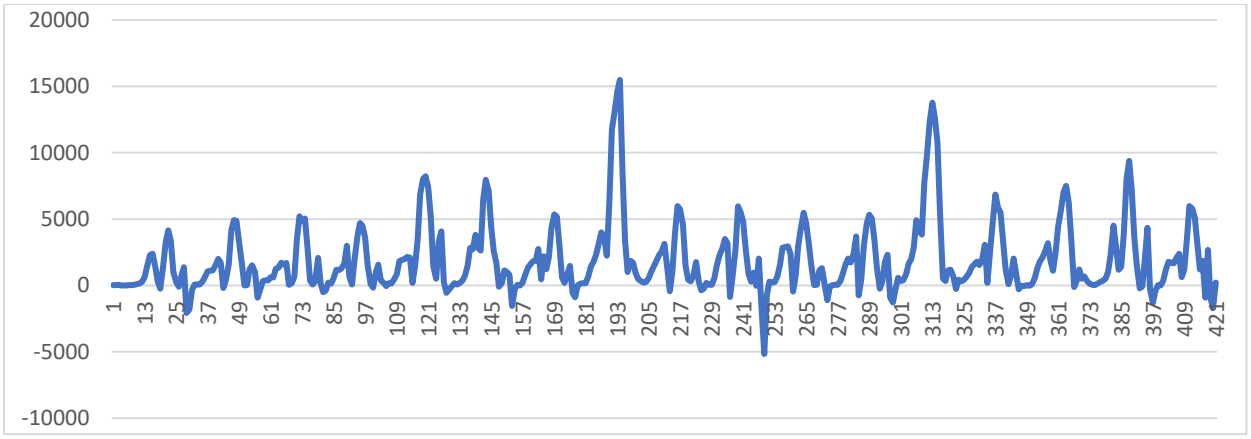


Fig. 28. *Graph showing covariance algorithm processed calf placed IMU sensor readings for 34 steps descending downstairs.*

3.2.2. Step Detection Algorithm

```
StepsCounted = 0;
LastTimeStep = 0;
MinimunmLastStepTimeInterval = 10;
AverageChangeInStep = 25;
RelativeThreshold = 0.9;
SlidingWindowSize = 25;
MinimumThreshold = 1500;
IMUDataIteration = 0;

LOOP: Algorithm() {
    GetStepCounts(IMUData) {
      If (LastStepTaken == AverageChangeInStep/2) {
        StepsCounted++;
      }
    CovInput = CovarianceAlgorithm();
    SlidingWindowArray[IMUDataIteration % SlidingWindowSize] = CovInput;
    StepDetectionAlgorithm((int) (RelativeThreshold * Math.max(MaxInSlidingWindowArray(), MinimumThreshold)));
    IMUDataIteration++;
    LastTimeStep ++;
   Return StepsCounted;
```

```
  }
StepDetectionAlgorithm(int Peak)) {
    If (CovInput > peak && !AtPeakPosition) {
      If (LastTimeStep > MinimunmLastStepTimeInterval){
        StepsCounted++;
        AverageChangeInStep = (AverageChangeInStep + LastTimeStep)/2;
        LastTimeStep = 0;
      }
     AtPeakPosition = true;
    }
   If (CovInput < Peak && AtPeakPosition) {
      AtPeakPosition = False;
    }
}
```

The Step detection algorithm as seen in the pseudocode above, presents a way to detect the steps taken given the inputs from the Covariance algorithm. This algorithm makes use of a sliding window array that gets updated after every loop (Entry of new dataset from Bluetooth transfer). The sliding window array is checked to see the highest peak value and ninety percent of that value is used as the threshold for step/peak detection. Rapid inflow of inputs to the sliding window array would result to constant change in this threshold value which is an adaptive change according to the user's mode of walking or running. This threshold value is used to detect when a step is taken. To further increase the accuracy of this detection of steps, a minimum number of time steps (or Iterations) is used to guard against close multiple peaks being counted as two or more steps. If there are two or more peaks in the sliding window array, the are regarded as a single step.

This algorithm only recognises the step taken on one foot, which is the foot with the IMU sensor placement. To get the steps taken by the user, the interval between the last step taken and the current step taken is a full cycle time of two steps taken by the leg with the IMU sensor placement from the first step to the next step. Some where in between that cycle, the other leg would have taken a step. This algorithm assumes that the other leg takes a step in the half (centre) of the cycle time. When a step is detected, the average between the previous full step cycle time and the current full step cycle time taken by the foot with IMU placement is derived. The half (median) of this averaged cycle time is used as the time to count the step of the other foot. This method allows for adaptive changes in the counting of the steps taken by the other foot based on the speed/mode of waking.

The number of counted steps is returned, which is displayed to the user.

3.3. DESCRIPTION OF DEVELOPED ANDROID APPLICATION

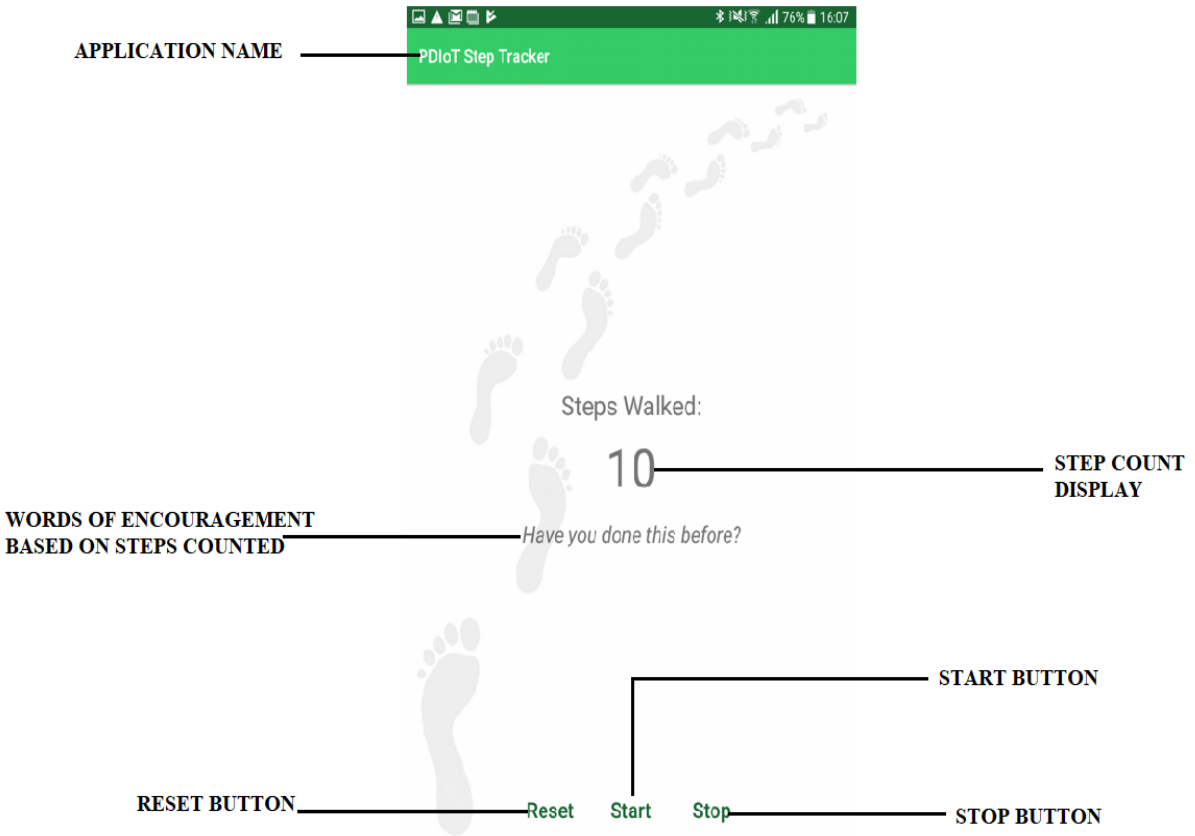


Fig. 29. *Description of developed android application.*

The figure above shows the description of the interface of the developed android application. The listed features are characteristics that make up the android application. These features are described in this section.

- Application Name
  This displays the name of the android application.

- Step Count Display
  This displays the number of steps detected and counted. These values all change in real time according to the number of steps taken in real time.

- Words of Encouragement Based on Steps Counted
  These are programmed words that are displayed to the user as the number of steps counted reaches a specific range of numbers. This function can be used to derive information concerning the status of the user's body motion or the calories dissipated and so on.

- Start Button
  This is pressed whenever the user wants to use or resume the use of the step tracker to count the steps taken.

- Reset Button
  This is pressed to reset the step counter to zero.

- Stop Button

  This is pressed to pause the step counter. Start button would have to be pressed to resume counting.

3.4. SOFTWARE STRUCTURE

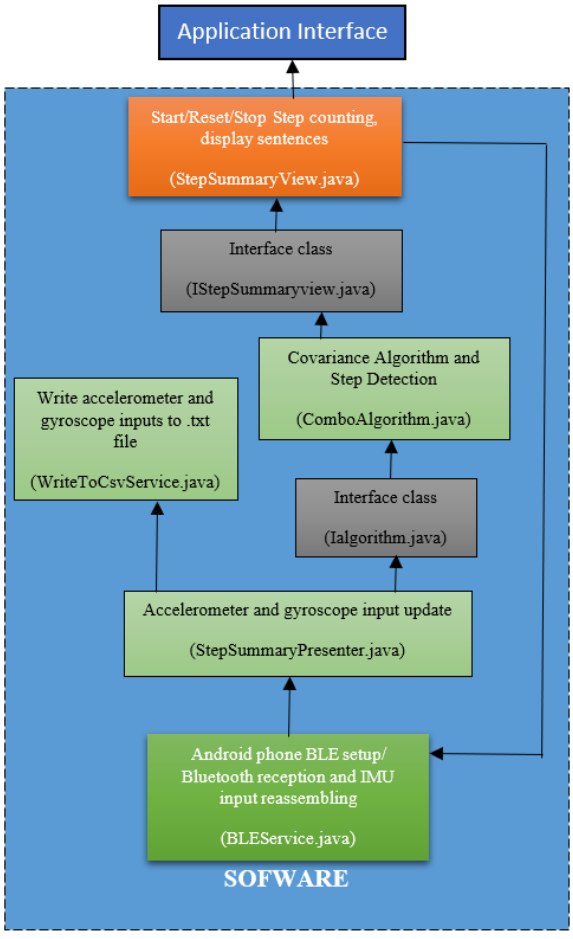


Fig. 30. *Block diagram depicting the Software Structure.*

This section discusses the different classes/stages of the software structure. As shown in figure 30, which describes the structure of the software, these classes are briefly discussed.

3.4.1. BLEService.java

At this stage, this class file sets up the Bluetooth connection between the nRF52-DK development kit and the android phone. After the connection is set up, the IMU sensor inputs are transferred disassembled. This class handles the reception and the reassembling of the inputs.

3.4.2. StepSummaryPresenter.java

At this stage, this class initializes and receives the dataset input from the BLEService class. A public method – updateGyroAccel(), sets up the data to be written to .txt file, and calls the IStepSummaryView.java method – updateStepCount, which update the number of steps counted displayed to the user, and calls the step counter method in the ComboAlgorithm.java which houses the covariance algorithm and step detection algorithm. The objects of the various classes are used in calling these classes and their methods every time there is an update from the IMU sensor readings. This keeps the software in loop. This class also sets up the input datasets to be written in a given format to the .txt file. It defines an object of the WrtieToCsvService.java class, which is used to record the datasets, and open or close a new .txt file.

3.4.3. WriteToCsvService.java

This class sets up the writing to .txt file service for the input datasets. It sets up the name format of the .txt file which made use of the date and time of that period of writing to .txt file. This class also verifies whether the external storage to be used to store the .txt file is writable, readable and available, when verified, various methods are used to write the dataset accordingly and close the file when walk is finished.

3.4.4. Ialgorithm.java

This is an interface class, that has two methods, where the first method – getStepCount(), returns the number of steps and the second method – resetStepCount(), sets the number of steps counted to zero.

3.4.5. ComboAlgorithm.java

This class houses the covariance and step detection algorithm which has been discussed earlier. This class implements the Ialgorithm,java interface class. The steps counted are returned when acted upon by the algorithms.

3.4.6. IStepSummaryView.java

This is an interface class that is composed of a single method – updateStepCount(), which uses the returned value from the ComboAlgorithm.java class as updates to the previous counted steps. This updated value is what displayed and updated as the number of steps counted to the user in real time.

3.4.7. StepSummaryView.java

This class implements the IStepSummaryView.java interface class. This class sets up the interface of the application, and some words (encouragements) are written, and each of them are set to display when the user's number of steps taken reaches a given range respectively. The functions of the buttons – Start, Stop, and reset, are defined in this class.

3.4.8. Application Interface

This is the outlook of the designed application as shown in figure 29. All the processes done in the software reflects on the change/update of the number of steps counted shown in the application interface along with the various words of encouragement. The user expresses control by using the buttons to start, stop or reset the counting of the steps taken.

3.5. TESTING

To analyse and evaluate the workings and accuracy of the designed step tracker, several tests were taken which are described in this section. Multiple walks and runs were taken on the flat surface, climbing up the stairs and descending the stairs. The correct counted steps and the steps counted by the step tracker are compared, where the error rate and accuracy of the scheme is derived and later compared with the other schemes proposed in the reviewed works in Chapter two and detailed in next chapter. The placement of the IMU sensor was changed and steps are taken to verify the placement and orientation independency of the system.

Table 1. *Steps counted for walk on flat surface by step tracker*

FLAT SURFACE	**Steps Taken**		
Steps Taken	**First Walk**	**Second Walk**	**Third Walk**
20 Steps	19	19	18
30 Steps	28	30	28
40 steps	40	39	40
50 Steps	48	49	48

Table 2. *Steps counted for climbing upstairs by step tracker*

CLIMBING UPSTAIRS	**Steps Taken**		
Steps Taken	**First Walk**	**Second Walk**	**Third Walk**
20 Steps	19	16	19
30 Steps	26	28	26
40 steps	35	40	40

Table 3. *Steps counted for descending downstairs by step tracker*

DESCENDING STAIRS	**Steps Taken**			
Steps Taken	**First Walk**	**Second walk**	**Third Walk**	**Fourth Walk**
20 Steps	18	19	18	18
30 Steps	30	26	28	27
40 Steps	40	38	37	35

The tables above are tests done with the IMU sensor placed on the calf in different orientations whenever a new walk is to begin. After several walk tests were done, tests on running scenarios where also done, and it was discovered that the performance degrades because further improvement to detect steps when running was not implemented in this work. The steps as anyone can see, are mostly few steps behind the correct steps taken, due to the different placement and walking scenarios. When the tests for climbing and descending the stairs were taken, the when walking and the user encounters a landing, due to change in the style of walk which would lead to the change in the pattern of the graph and peaks. For the step detection algorithm, it would need some time to adjust to the new type of peaks encountered, which can lead to no detection of some steps. Therefore, some steps are missed, which gives an incomplete number of steps that are counted.

4.0. RESULTS AND ANALYSIS

In this chapter, the result gotten from the tests taken are used to deduce the detection accuracy of this step tracker, that is compared with the other schemes presented in the review works in chapter two.

4.1. Results

The test results as seen in the testing section show the accuracy of the step tracker to detect the number of steps. The tables below show the average detection success of the step tracker. Noting that each of the walks taken are done with change in the orientation of the IMU sensor, which has resulted to varying number of counted steps. It was identified that the step tracker performance degrades as the user runs, which is resulted from the step detection algorithm employed. Seeing that the BMP280 (Barometric) sensor is used in the hardware design, much of its implementation was not used, which would need to be improved for implementation. The stairs climbing and descending test where done in the Appleton tower, University of Edinburgh. The flat surface test was done indoors and outdoors.

Table 4. *Average Percentage of the performance of step tracker on flat surface*

FLAT SURFACE	**Detection Accuracy (%)**			
Steps Taken	**First Walk**	**Second Walk**	**Third Walk**	**Average**
20 Steps	95	95	90	**93.33%**
30 Steps	93.33	100	93.33	**95.33%**
40 steps	100	97.5	100	**99.17%**
50 Steps	96	98	96	**96.67%**
Total Average	**96.13%**			

Table 5. *Average Percentage of the performance of step tracker for climbing the stairs.*

CLIMBING UPSTAIRS	**Detection Accuracy (%)**			
Steps Taken	**First Walk**	**Second Walk**	**Third Walk**	**Average**
20 Steps	95	80	95	**90%**
30 Steps	86.67	93.33	86.67	**88.89%**
40 steps	87.50	100	100	**95.83%**
Total Average	**91.57%**			

Table 6. *Average Percentage of the performance of step tracker for descending the stairs.*

DESCENDING STAIRS	**Detection Accuracy (%)**				
Steps Taken	**First Walk**	**Second Walk**	**Third Walk**	**Fourth walk**	**Average**
20 Steps	90	95	90	90	**91.25%**
30 Steps	100	86.67	93.33	90	**92.50%**
40 steps	100	95	92.50	87.50	**93.75%**
Total Average	**92.50%**				

4.2. Analysis

Comparison of the performance of this step tracker with the schemes used in the reviewed works are shown below in the table. The proposed scheme according to [2] is compared with this step tracker based on the performance of the proposed scheme for walking as depicted in the results seen in the review in chapter two. The proposed scheme according to [3] is compared with this step tracker based on the performance of the proposed scheme when walking at regular pace, where the results are also depicted in the review in chapter two.

Table 7. *Comparison of developed step tracking scheme with the schemes proposed in the reviewed works.*

	1	2	3	4	5	6	**Average**
Developed Step Tracker	96.13	91.57	92.50	–	_	_	**93.4%**
Proposed Scheme According To [2] (S4: Gravity and acceleration)	99	99	99	95	_	_	**98%**
Proposed Scheme According To [3] (Walking at regular pace)	100	99.85	99.90	99.95	100	99.80	**99.75%**

Looking at the results of the step tracker and the comparison with the other schemes as seen in the reviewed works, shows a good detection accuracy of the developed step tracker.
As stated in the testing section of chapter three, for the testing of climbing and descending the stairs was done with staircase with landings. These landings were about 10 steps apart, which caused the step detection algorithm to miss steps before adjusting to the different type of peaks (graph pattern). Generally, the developed step tracker is able to detect steps with an accuracy of over 90% when walking on flat or gradient surfaces, climbing or descending the stairs. Finding the best orientation and placement of the step tracker would improve the performance of the step detection ability of the step tracker. The calf position, just below the knee was identified to be the most suitable position for the IMU sensor placement.

5.0. CONCLUSION

In this project, a step tracker that is based on using an IMU sensor, nRF52-DK development kit that connects to a developed android application to count the steps of the user's motion is developed. Steps taken, and methods followed in conducting this project were discussed to give a full understanding of the developed system, which depicts the limitations and capabilities of the step tracker. According to the results, the developed step tracker works with a high level of accuracy and can be implemented in different placements and orientation of the IMU sensor on the user's leg. The product of this project can be a step tracker system that can be placed in shoes or on trousers, where the step tracker system can serve as a fitness application. During the analysis of the resultant data processed from the covariance algorithm as seen in chapter three, where it shows different graph patterns for walking on flat surface, climbing or descending the stairs. These differences can be harnessed to help in classifying the type of motion the user is making, which would tell whether the user is walking, climbing or descending. The derived information can be used to deduct an accurate number of calories lost or energy dissipated by the user by taking into consideration the kind of walking motion the user makes. The implementation of the Barometric sensor was not completed. This feature is important and can be improved upon to display the user's current height. Since the BMP280 (Barometric sensor) is equipped with temperature sensing capability also, it can be used to derive weather reports to the user and can be used to deduct heathy tips for the user based on the state of the environment the user is walking in as sensed by the BMP280 sensor.

The developed step tracker can be improved upon also by improving the step detection algorithm and properly exploiting and understanding the covariance algorithm. This would improve the accuracy of this scheme and allow it to be used properly in all possible walking or running motion. Another area of concern would be in the power supply of the hardware. Because of the constant transfer of data to the android application, the battery of the nRF52_DK development kit runs out faster than expected. This would be resolved by using a specially made chip/board that can be used for the sensing and transfer of the data, and another can be to use protocols that would allow the hardware kit to switch off Bluetooth when not in use and get reactivated when to be used. This would conserve energy and allow the system to be used when desired and when necessary. The implementation of this developed step tracker system, introduces another approach that can be improved and used in step tracking/counting applications. The results as seen in chapter four, shows that this approach can be developed and improved upon for further use.

REFERENCES

[1] N. Kobie, “theguardian,” 6 may 2015. [Online]. Available: https://www.theguardian.com/technology/2015/may/06/what-is-the-internet-of-things-google. [Accessed 24 December 2017].

[2] Q. Zeng, C. Jing, N. Kim and Y. Kim, “A Novel Step Counting Algorithm Based on Acceleration and Gravity Sensors of a Smart-Phone,” *International Journal of Smart Home,* vol. IX, pp. 211-224, 2015.

[3] M. Khedr and N. El-Sheimy, “A Smartphone Step Counter Using IMU and Magnetometer for Navigation and Health Monitoring Applications,” *Sensors,* 2017.

[4] “Sparkfun Electronics,” [Online]. Available: https://www.sparkfun.com/products/13762. [Accessed 29 12 2017].

[5] “arm MBED,” [Online]. Available: https://os.mbed.com/platforms/Nordic-nRF52-DK/. [Accessed 29 12 2017].

[6] “Bosch Sensortec,” [Online]. Available: https://www.bosch-sensortec.com/bst/products/all_products/bmp280. [Accessed 30 12 2017].

[7] “SparkFun Electronics,” [Online]. Available: https://www.sparkfun.com/products/13676. [Accessed 30 12 2017].

[8] “StackExchange,” [Online]. Available: https://physics.stackexchange.com/questions/333475/how-to-calculate-altitude-from-current-temperature-and-pressure. [Accessed 31 12 2017].

[9] “arm MBED,” [Online]. Available: https://os.mbed.com/teams/Bluetooth-Low-Energy/. [Accessed 31 12 2017].