

Engineering Assignment Coversheet

Student Number(s)

671116

Group Code (if applicable):

Please note that you:

- Must keep a full copy of your submission for this assignment
- Must staple this assignment
- Must NOT use binders or plastic folders except for large assignments

Assignment Title:	Project 1: Characterisation & Simple Application of Sensors
Subject Number:	MCEN90032
Subject Name:	Sensor System
Student Name:	Akshay Kumar Bharat Kumar
Lecturer/Tutor:	Mitchell Khoo
Due Date:	19/8/2018

For Late Assignments Only

Has an extension been granted? Yes / No (circle)

A per-day late penalty may apply if you submit this assignment after the due date/extension. Please check with your Department/coordinator for further information.

Plagiarism

Plagiarism is the act of representing as one's own original work the creative works of another, without appropriate acknowledgment of the author or source.

Collusion

Collusion is the presentation by a student of an assignment as his or her own which is in fact the result in whole or in part of unauthorised collaboration with another person or persons. Collusion involves the cooperation of two or more students in plagiarism or other forms of academic misconduct.

Both collusion and plagiarism can occur in group work. For examples of plagiarism, collusion and academic misconduct in group work please see the University's policy on Academic Honesty and Plagiarism: <http://academichonesty.unimelb.edu.au/>

Plagiarism and collusion constitute cheating. Disciplinary action will be taken against students who engage in plagiarism and collusion as outlined in University policy. Proven involvement in plagiarism or collusion may be recorded on my academic file in accordance with Statute 13.1.18.

STUDENT DECLARATION

Please sign below to indicate that you understand the following statements:

I declare that:

- This assignment is my own original work, except where I have appropriately cited the original source.
- This assignment has not previously been submitted for assessment in this or any other subject.

For the purposes of assessment, I give the assessor of this assignment the permission to:

- Reproduce this assignment and provide a copy to another member of staff; and
- Take steps to authenticate the assignment, including communicating a copy of this assignment to a checking service (which may retain a copy of the assignment on its database for future plagiarism checking).

Student signature

Date

18/8/18

MCEN90032 Sensor Systems 2018:

Project 1: Characterisation and Simple Application of Sensors

Project Completed by:

Akshay Kumar Bharat Kumar, 671116

Due by:

19th August 2018

A.) INTRODUCTION:

The use of sensors is prevalent in the modern world, being integrated in almost every piece of technology. Ranging from phones, computers, aeroplanes and HVAC systems, understanding how they are utilised and implemented is important. To establish this, the project as proposed to the students of MCEN90032 was divided into 3 parts:

- 1.) Familiarisation of the Arduino hardware and software
 - a. This involved programming the Arduino to blink its LED on and off for an indefinite amount of time.
 - b. This ensured that any sensors used in conjunction with the Arduino was wired, programmed and hence run correctly.
 - c. The understanding will be necessary for future projects.
- 2.) Characterisation and Calibration of the MEMS accelerometer, the 3 axes, ADXL335.
 - a. The accelerometer is to be used for the creation of a simplistic and highly idealised pedometer.
 - b. To ensure this, the accelerometer needs to be calibrated to be $\pm 1G$ on each of its axes.
- 3.) Building a Pedometer
 - a. With the calibration complete, the Arduino and accelerometer were then to be designed to act as a pedometer, i.e. Count the number of steps taken as the student/ operator walks.

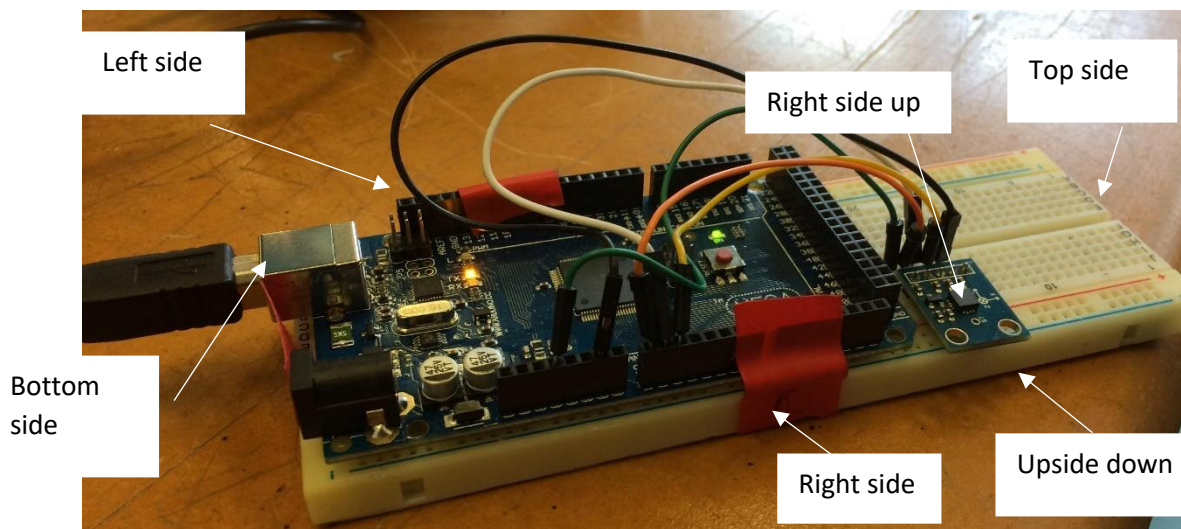


Figure 1: Pedometer setup, consisting of the Arduino Mega, Wires, Breadboard and ADXL335 accelerometer. Labelled are the predefined directions for the purposes of calibration

With these 3 different challenges to be tackled, the following report outlines the different methods and procedures, in addition to the results produced.

B.) PART 1: FAMILIARISATION OF THE ARDUINO HARDWARE AND SOFTWARE

To familiarise the student with the Arduino and the programming style, a simple LED 'blink' function was tasked to be created.

To do this, the LED pin number of the Arduino was required. It is known that the pin number was defined to be '13' which was then defined to be an output using the function:

PinMode();

To turn the LED on, the function 'digitalWrite' was used to manually set the LED to on, or HIGH.

Similarly, to turn the LED off, the LED was set to off, or LOW.

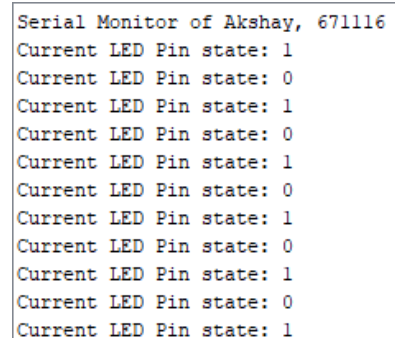
A delay of 1 seconds was included between the different LED modes, to ensure that the blinking was visible and noticeable. Additionally, to ensure that it would continue indefinitely, this was placed within the 'void loop()' function.

In addition to being able to see the changes visibly, a way was needed to track the changes within the Arduino in of itself. To do this, the following statement was used:

Serial.begin(9600);

This would allow communication between the board and the Serial Monitor/ Plotter, allowing for ease of data tracking and collection. The image below shows the Serial Monitor indicating when the LED was on (Current LED Pin state: 1) and was off (Current LED Pin state: 0).

With the blinking task completed, the next step involved the use of the MEMS accelerometer.



```
Serial Monitor of Akshay, 671116
Current LED Pin state: 1
Current LED Pin state: 0
Current LED Pin state: 1
Current LED Pin state: 0
Current LED Pin state: 1
Current LED Pin state: 0
Current LED Pin state: 1
Current LED Pin state: 0
Current LED Pin state: 1
Current LED Pin state: 0
Current LED Pin state: 1
Current LED Pin state: 0
Current LED Pin state: 1
```

Figure 2: Serial Monitor showing when the lights turn on and off

C.) PART 2: CHARACTERISATION AND CALIBRATION OF THE MEMS ACCELEROMETER, THE 3 AXES, ADXL335:

The MEMS accelerometer used was 3 axes, ADXL 335. This accelerometer was an ideal selection due to the relatively cheap price, in addition to its ability to measure both static and dynamics accelerations.

The accelerometer has 5 pins; 1 GND pin, 1 VCC pin, and an X, Y and Z pin. The VCC and GND pin are simply the way to power the sensor via the Arduino, by connecting it to the 3.3V and GND pin on the Arduino respectively.

The X, Y and Z pins were then connected to the ANALOG IN section of the Arduino board, into pins A0, A1 and A2 respectively.

With the wiring complete, the data being collected from the accelerometer must be transferred and read by the Arduino. To do this, the function **analogRead()** was used, where the input argument was the pin number attached to the X,Y or Z pins.

Plotting the data in the Serial Monitor produced the following graph:

This plot gives the user quick feedback as to the data collection, however, provides no useful information. As such, the data was printed into the Serial Monitor, which was then mass copied, and hence plotted using Microsoft Excel. The resulting graph, shown below is similar Figure 3 below.

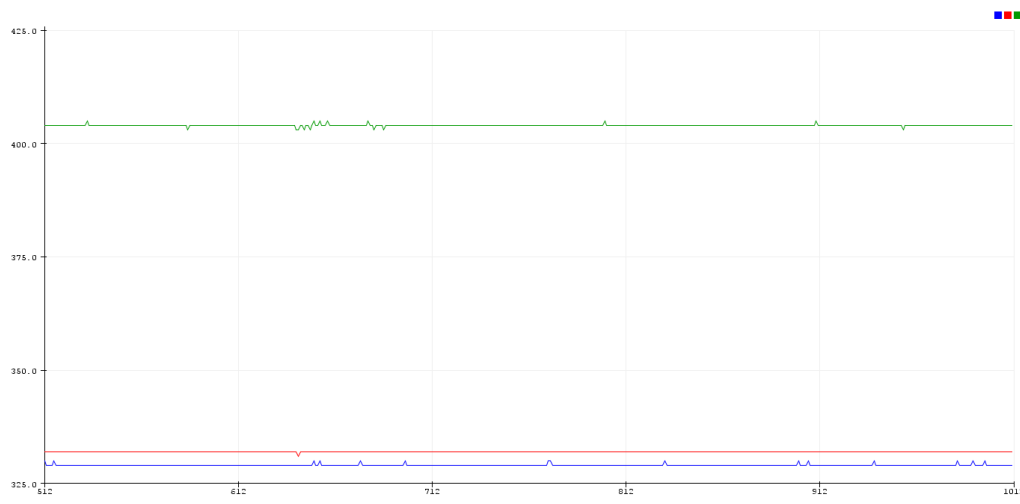


Figure 3: Serial Plotter of the accelerometer reading when placed Right Side Up.

These raw values are the ADC ranges (0 to 1023) which are present in the Arduino. Given that the ADXL335 can read accelerations from +3G to -3G, when the accelerometer is laid flat, it is reading approximately 400 due to the fact that only +1G is acting on it. If the full 3Gs were experienced, the maximum ADC range would be reached. These values however seem arbitrary and provided no significant meaning. Hence, using these values for the purposes of a pedometer required a calibration process.

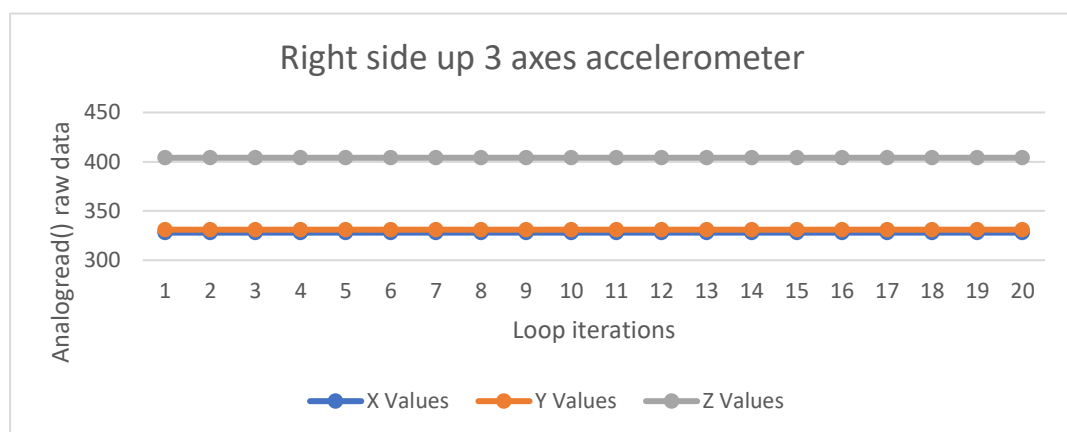


Figure 4: The same data being plotted in Microsoft Excel after the data had been extracted via the Serial Monitor.

Given the sensors ability to measure static accelerations, when at rest, the only acceleration which would be acting on it would be the force of gravity in of itself, which has been defined to be 1G for simplicity.

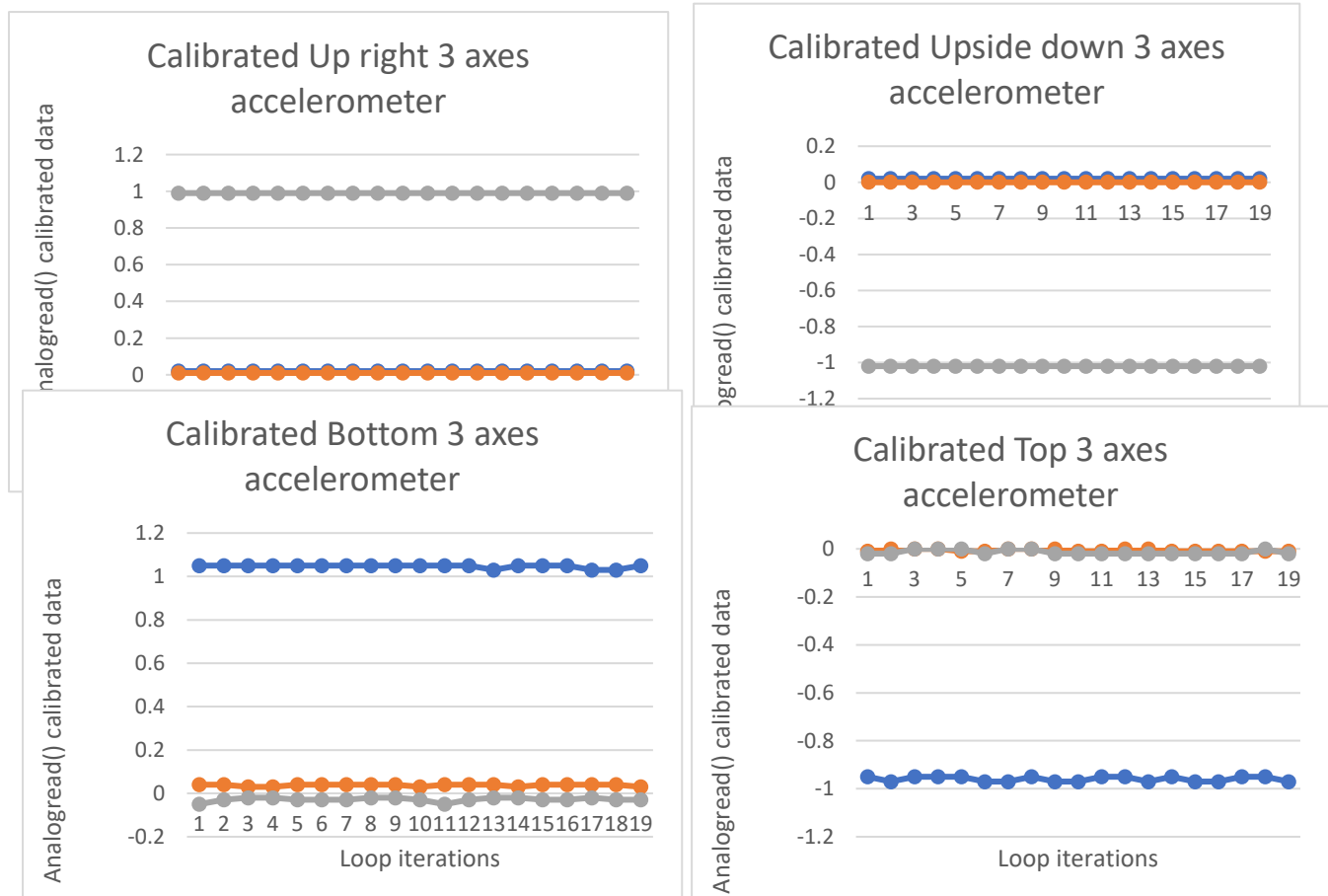
By tilting the accelerometer onto its different axes, placing it there and imparting no force on to it, the raw data being read would be equivalent to that of 1G. For example, by laying the accelerometer

flat against breadboard as shown in the figure setup (Figure 1), a maximum Z Value can be obtained. Similarly, by flipping the accelerometer upside down, a minimum Z value can be obtained. Both these values correspond to both +1G and -1G respectively. To accurately determine the value at which this maxima or minima was reached, a 10 second calibration period was used, where the accelerometer was placed right side up, and the maximum value was recorded, and conversely, the minimum value recorded when placed upside down. This calibration process was completed a couple of times and once fully analysed to for consistency in obtaining the maximum and minimum values, the section of code was commented out to increase speed.

Following this, a modified version of the in built map() function, which was named in the sketch as map2_float() was used. This allowed for floats to be returned, instead of only integer values, hence calibrating these maxima and minima to be equal to +1G and -1G of acceleration. This process was repeated for the other 2 axes, by tilting the breadboard on the other 4 planes of surface. The maximum and minimum raw results for each setup are shown in the table below.

<i>Axes Setup</i>	<i>IMS Raw Value</i>
Up Right, Maximum Z Value	404
Upside Down, Minimum Z value	272
Right, Maximum Y Value	398
Left, Minimum Y Value	264
Top, Minimum X value	264
Bottom, Maximum X Value	398

The following 6 images were the calibrated data. Notice how the static accelerations now only range between approximately +1G to -1G.



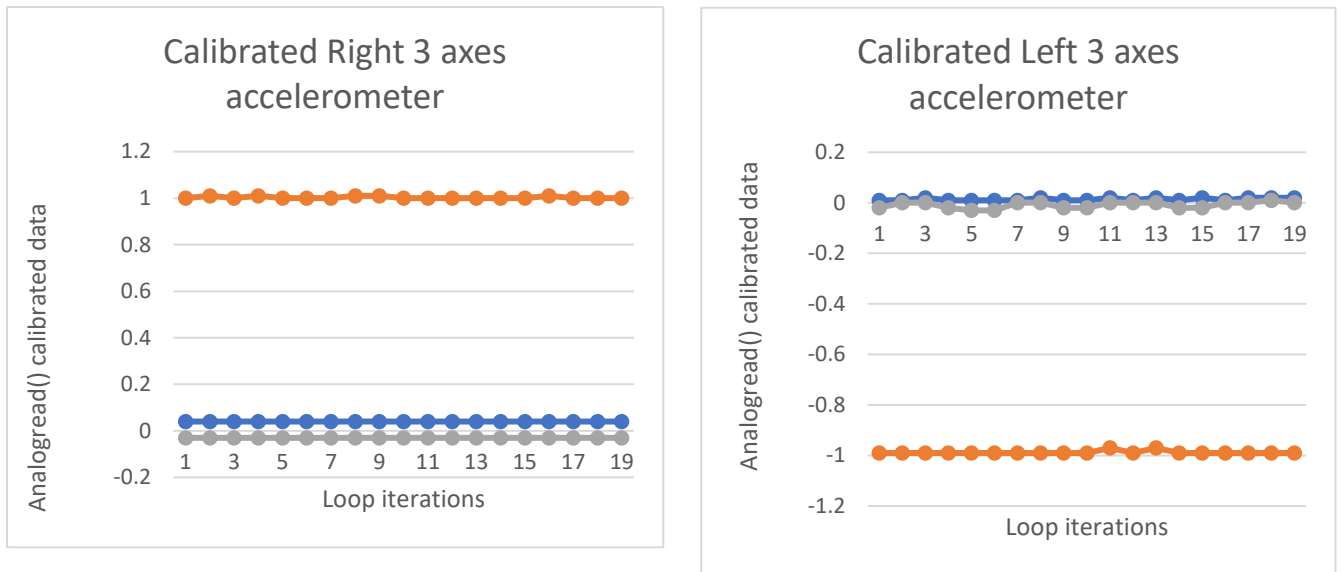


Figure 5: All the 6 different axes being read after being calibrated to be equal to that $+1G/-1G$ for the maxima and minima. The grey is the Z axis, orange is Y axis, and Blue is the X axis.

D.) PART 3: BUILDING A PEDOMETER USING THE ACCELEROMETER

With the calibration process completed, now we can check how the values change as the student walks around with accelerometer placed in his pocket. The following graph was produced as the student walked 4 slow steps. As can be seen, with each step taken, there were peaks noticed prominently in the X, Y and Z value.

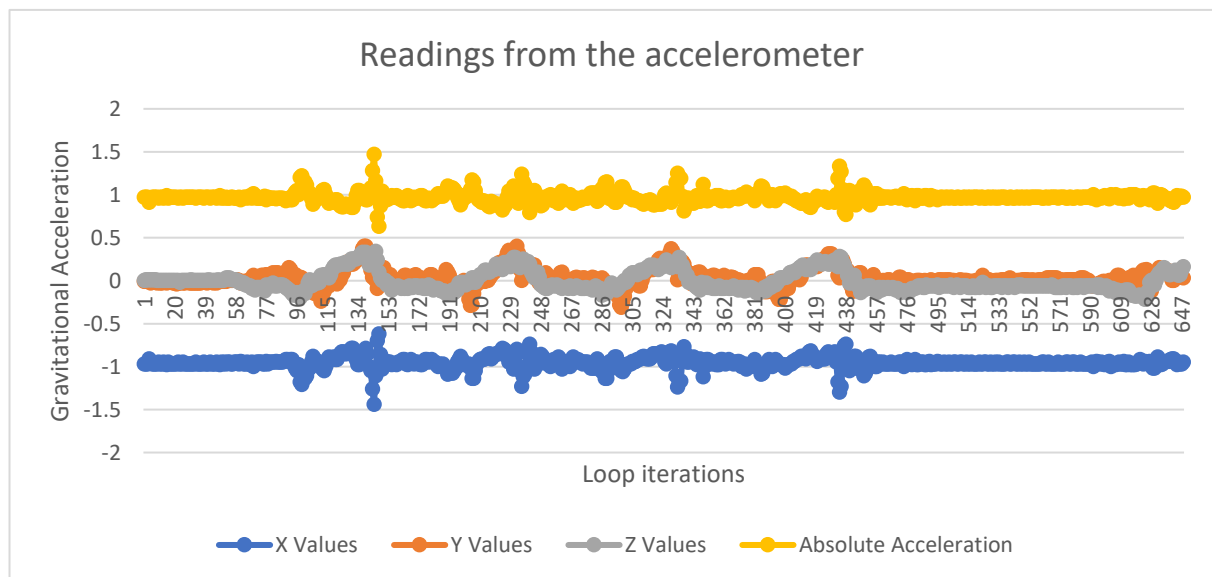


Figure 6: The 'pedometer' readings as the student took 4 slow steps across the room. Each peak represents the step taken.

This was primarily due to the way the accelerometer and Arduino attached to the breadboard were placed within the pocket, as shown in Figure 7 due to the wire connecting the Arduino and computer being short.

From Figure 6, the yellow line is the absolute acceleration. Given that the 3 different axes directions, an overall and absolute acceleration was necessary to obtain reliable and useful data. To do this, the absolute acceleration was calculated through the following equation:

$$\text{Absolute acceleration} = \sqrt{(\text{acceleration}_x)^2 + (\text{acceleration}_y)^2 + (\text{acceleration}_z)^2}$$

Following this analysis, a threshold was needed to ensure that the pedometer would record and accurately count each step allocated for. Each counter was to count both the left and right steps and ensure that no double counting occurred within a single step. Moreover, light stepping and quick running were needed to be counted accurately with the accelerometer.

After some trial testing with walking, running, and sneaking, an appropriate threshold was determined and as such, was implemented into the sketch. This threshold is as follows:

When the absolute acceleration exceeds 1.05Gs, or when the x value acceleration drops below -1.05Gs, count that as a step. To ensure no double counting occurred, a delay of approximately 0.5 seconds was introduced. This would allow the fluctuating acceleration readings to return to below the threshold before the next step was counted.

The reasoning behind using 2 sets of accelerations was to add redundancy to the system so that no counts were missed. Additionally, the X-acceleration values would help with the detection of the left foot step. Given that the 'pedometer' was placed in the right pocket, steps taken with the left leg would be less noticeable to the accelerometer. As such, the X acceleration was used to overcome this challenge.

The other axes could be utilised to add further redundancy to the system. However, from testing, the current layout and threshold was more than sufficient for counting steps and as such, no additional redundancy was thought to be necessary for this simplistic pedometer. The following graph shows the spikes in the total acceleration (absolute) and the x-value accelerations which line up and hence correspond to each successful step taken.

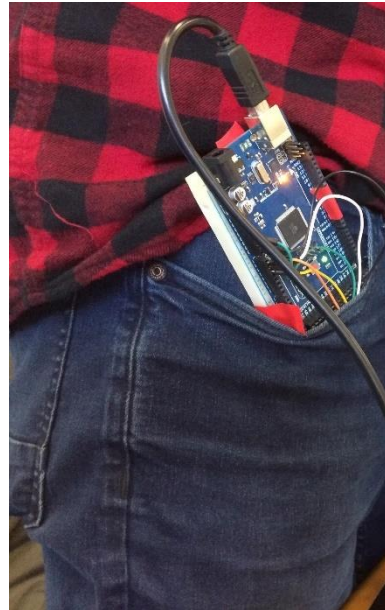


Figure 7: The 'pedometer' being placed within the pocket of the student.

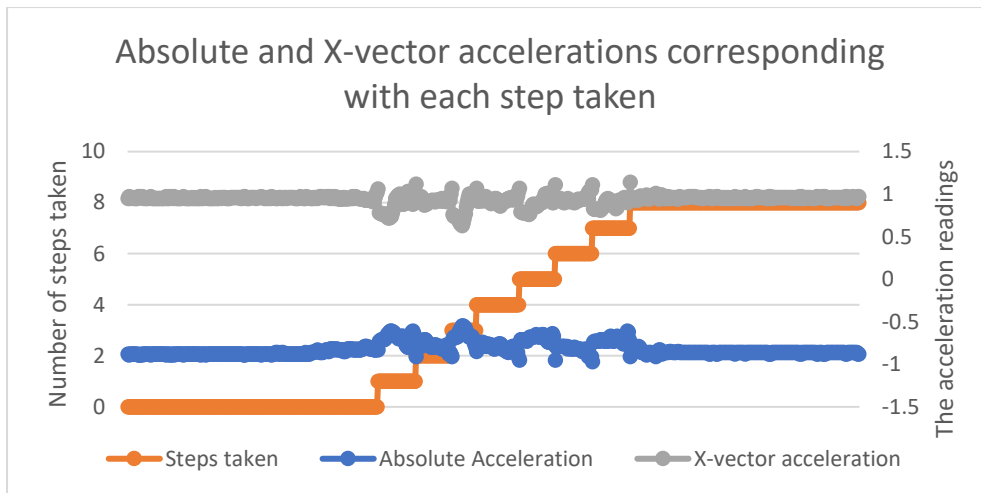


Figure 8: The absolute and x-vector acceleration trend corresponding with each step taken

From this, the accelerometer and Arduino combination have thusly been calibrated and are hence quite suitable its use as a pedometer. Of course, there are limitations to this, as the breadboard, wires, Arduino and accelerometer which were used to make this pedometer are quite bulky and not easy to carry around. Moreover, the Arduino requires a connection to a computer to continuously keep it powered, which would make it quite cumbersome to use as a pedometer for outside purposes.

E.) CONCLUSION:

From this project, a greater understanding behind an Arduino and how to use and program it was obtained. Moreover, being to wire up, read data sheets and being able to analyse and understand the data being read by the Arduino will greatly help in future projects involving more MEMS and more complicated applications.

The project offered significant insight into how challenging it is to make a functioning and reliable pedometer which is robust against false steps and giving live feedback to the user.

Even the current code, calibration and thresholds used to count the steps is only suitable for when the Arduino and accelerometer are arranged in the manner shown in Figure 7 and normal paced walking is done. If the steps taken are too slow or far too quick, the steps will be double counted or not be counted for the respective scenarios. As such, the system is not too robust, and this may be simply a limitation of the MEMS accelerometer used or the threshold and calibration process undertaken.

As such, further in-depth fine tuning and optimisations would be necessary for the pedometer to accurately count every step to more accuracy. One such option is utilising and implementing a high pass filter to filter and remove the noise in the signals being read.