James Bithell

6th Form Physics

C2 Practical Project

Programmable Electronics

Accelerometer driven footfall measuring device

Table of Contents

Introduction

Working in a pair with William Ashton, under the supervision of Mr Thorley, we have spent the last 5 weeks, during Monday's periods 3 & 4, working on a programmable electronics project – and accelerometer driven step counter, powered by an Arduino micro-controller. The project has been broadly successful, though it has not been without its challenges. We've learned a lot, as this writeup will detail, and had to adapt our plans on numerous occasions, but in the end we've been able to collect and examine a huge volume of data.

Activities

Introductory Session

We started our first session by organising the pairs, and setting objectives for what we hoped to achieve with the project – we decided to try and measure footfall electronically using vibration.

We agreed that we'd place the accelerometer device under the staircase, outside room 102 (the classroom our project sessions were based in), for convenience purposes. This location (indicated by a red dot in Figure 1) would therefore measure footfall for anyone travelling to the second floor and above, and possibly also measure vibration induced by the lift.

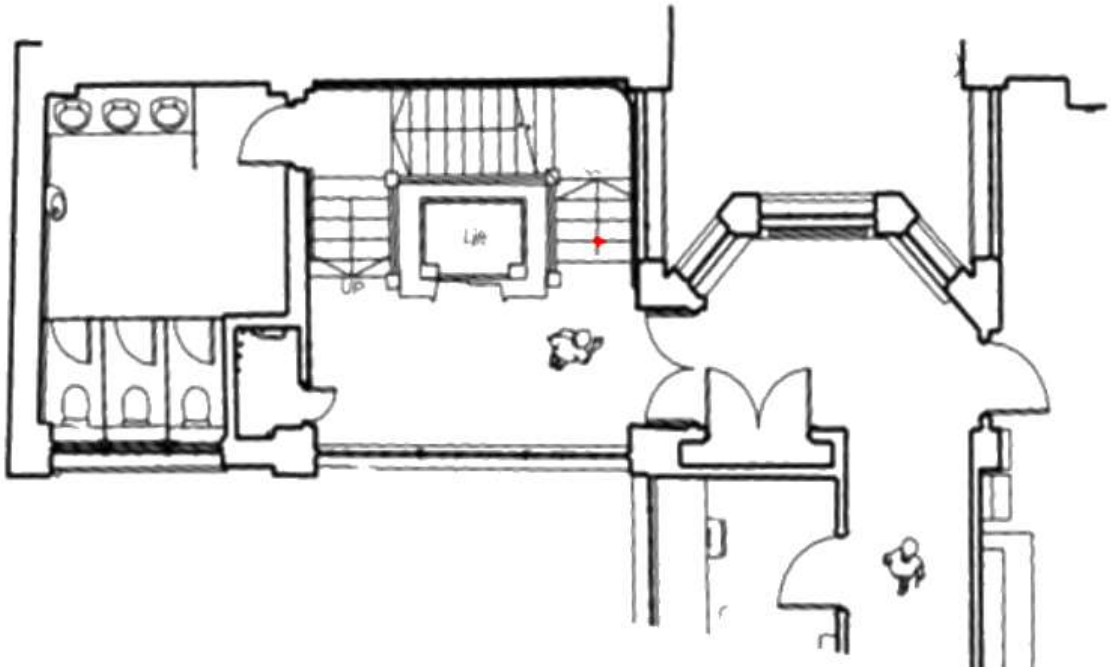

Figure 1 - Floor Plan, showing accelerometer location

We were offered the choice of an Arduino or a BBC Micro Bit to use as our micro controller, and after consulting their relevant specifications, we chose the BBC Micro Bit as we wanted to experiment with a newer device, and it had a built in accelerometer.

By the end of the first session we had powered up our accelerometer, and been able to push "Hello World" testing code to it. We then discussed how we'd store this data, and we agreed we would use a 1Gb SD card, which would be plenty of capacity on which to store a text file with all our data. Unfortunately, SD cards have not been used with the BBC Micro Bit before, though we weren't phased by this challenge, and agreed to look into how we would implement this over the

subsequent week, with Mr Thorley agreeing to solder an SD card adaptor and pins onto the BBC Micro bit board.

Week 1

During Week 1 we were able to get the accelerometer working on the Arduino, using the code in Figure 2. This was with support from the Computer Science department, for which I'm very grateful. The code essentially imports a library that enables it to run on the micro bit, then a function that converts a G value from the accelerometer, and converts it into a pixel value (the micro bit has a limited number of pixels on its screen), which is then outputted onto the screen. The more the device is shaken, the further the dot appears from the centre of the screen.

```
1.  #include "mbed.h"
2.  #include "MicroBit.h"
3.
4.  MicroBit uBit;
5.
6.
7.
8.  //
9.  // Scales the given value that is in the -1024 to 1024 range
10. // int a value between 0 and 4.
11. //
12.
13. int pixel_from_g(int value)
14. {
15.     int x = 0;
16.
17.     for(int i = -40; i <40; i+=20)
18.     {
19.         if(value > i) x++;
20.         else break;
21.     }
22.
23.     return x;
24. }
25.
26. int main()
27. {
28.     // Initialise the micro:bit runtime.
29.     uBit.init();
30.
31.     //
32.     // Periodically read the accelerometer x and y values, and plot a
33.     // scaled version of this ont the display.
34.     //
35.     while(1)
36.     {
37.         int x = pixel_from_g(uBit.accelerometer.getX());
38.         int y = pixel_from_g(uBit.accelerometer.getY());
39.
40.         uBit.display.image.clear();
41.         uBit.display.image.setPixelValue(x, y, 255);
42.
43.         uBit.sleep(100);
44.     }
45. }
```

Figure 2 - BBC Micro Bit Code to output accelerometer display

We arrived at our session to discover that Mr Thorley had kindly soldered on the adaptor, and thus we were able to insert a micro sd card into the micro bit, on which to store our data. Unfortunately,

after a great effort (based on our research the week before) we were still unable to successfully address the SD card, and therefore would be unable to actually store any data post collection. We made the executive decision to move to an Arduino board, and Mr Thorley agreed to order an accelerometer based on our selection (from a choice of 3) – we chose the MMA8451 because it was digital and had a higher resolution – thus we decided it would be most suitable for our project.

Week 2

During Week 2 we soldered the MMA8451 accelerometer to a ribbon cable (of 1.5m) and wired it into a breadboard. We then began to test it and started writing code for it, including code that could write to the SD Card.

The MMA8451 works by returning the relative position of the X, Y and Z axis, based on the device being in its "zero" or calibrated position (which for this device was flat on its back) – as the device is shaken, through the vibration of people climbing the stairs, these relative X, Y and Z positions change, and it is these positions we plotted.

Sadly, we were unable to address the external accelerometer using the Arduino, and so we ended Week 3 with a working micro bit accelerometer, and a working Arduino SD card writer, but unable to get the micro bit to write to an SD card, or the Arduino to work with the Accelerometer.

Week 3

When we arrived for our session in Week 3, we discovered that Mr Thorley had managed to fix the Arduino Accelerometer, which it transpired was not a software fault, but a hardware fault – we had failed to link two of the pins correctly to enable the board to set the write address (see wiring diagram).

We completed our code (printed in appendix), and setup the device in situ under the staircase by the end of the session. Figure 3 shows our setup, with the accelerometer under the red electrical tape, and the ribbon cable secured using duct tape. Figure 4 shows the Arduino and its associated battery pack to power it for the week in situ next to the staircase.

Figure 3 - Accelerometer under staircase treads

Figure 4 - Arduino and Battery Pack in Situ

We came to check on it later during the week, and it appeared to be working well, with Figure 5 showing it while the staircase was in use.

Figure 5 - Staircase in use

Week 4

As we planned, the Arduino had run for as long as its batteries had lasted, which was just over 70 hours. We then removed the SD card and batteries only (leaving the Arduino and the accelerometer in place), and extracted the data from the SD card, replaced the batteries, and put them all back, restarting the Arduino, so it would continue to collect data for another week.

For the remainder of the session we processed the data, converting the timestamp from milliseconds to hours, so it would display better in a graph. We then tried to use excel to generate a graph, but because of the volume of data collected (1,450,418 rows) the excel file was considerably large (and spread across multiple sheets due to size limits), and was Excel was unable to render the graph properly without crashing (see Figure 6). I agreed to look into methods for viewing the data graphically over the subsequent week.

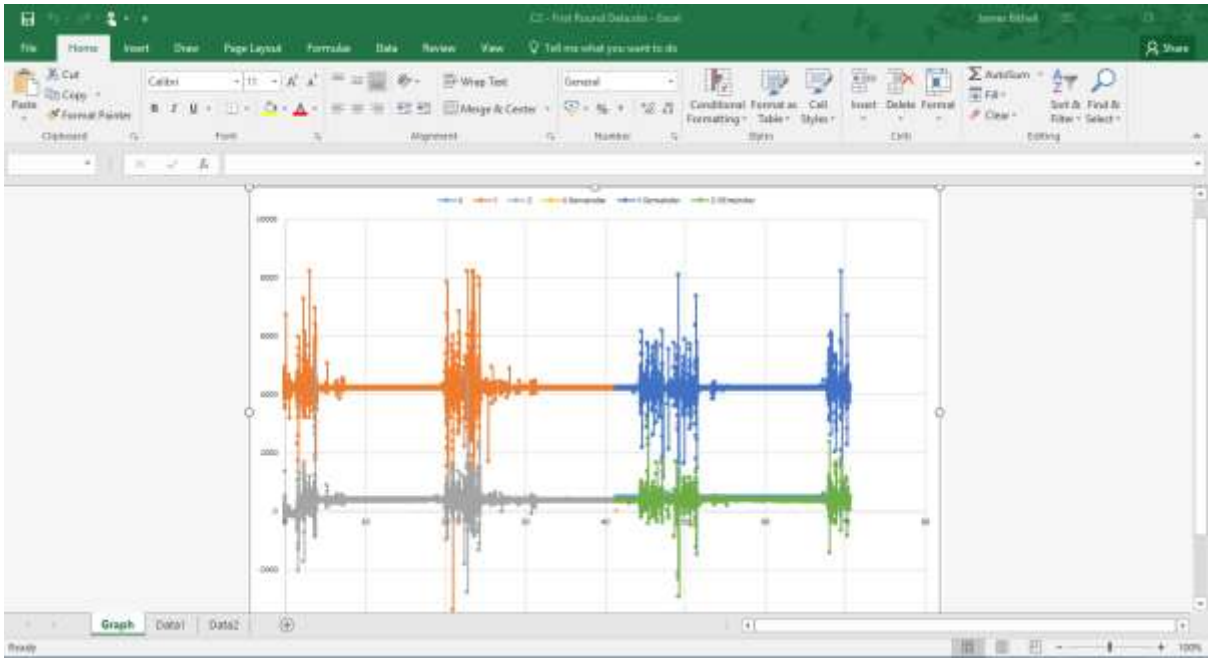

Figure 6 - Excel attempt at graphing the data

Partial Circuit Diagram

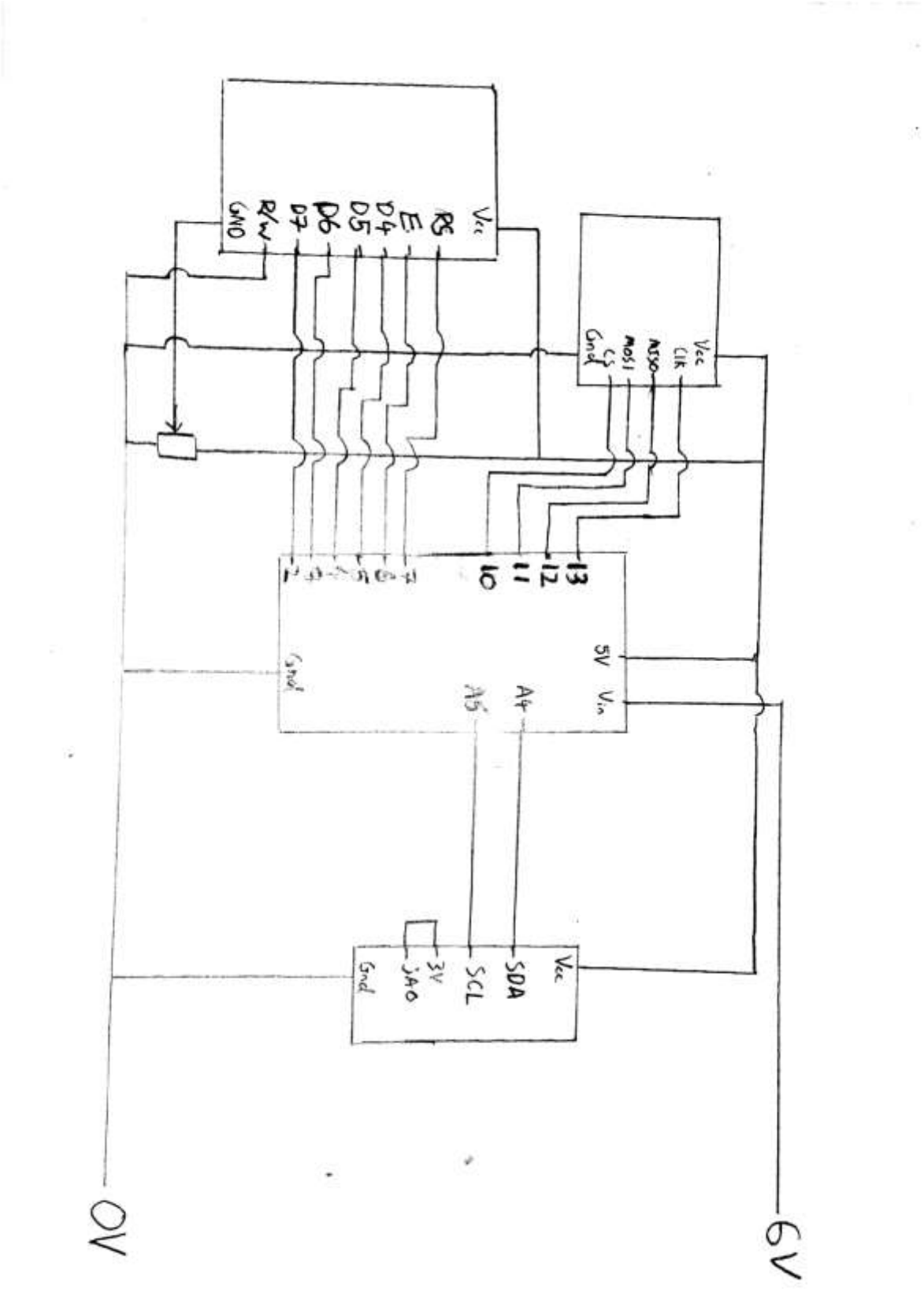

Figure 7 – Partial Circuit Diagram

Week 5

We arrived for the session and dismantled our apparatus, so it could be used in subsequent years. The Arduino had run this time for 166 hours, collecting 2,348,954 rows of data. This brought our total collected data to 3,799,372 rows, around 11million data points. The data collected during Week 3 is referred to as "Round1" and the data collected during Week 4 is referred to as "Round2" for administrative purposes.

We experimented with numerous ways of displaying our data, including Google Fusion Tables, Google Charts, and other windows based graph systems, but were unable to obtain any useful graphs that were detailed enough to explore or establish trends.

Subsequent Work

Subsequent work over the half-term break enabled the production of the graphs displayed in the data presentation section, using Excel to process the raw data, then various Linux search and replace functions to put it into a format (tab separated values) that can be handled by GNUPlot, a Linux data plotting/ graphing tool that enables big data to be presented into simple graphs in a powerful way. I rented a Linux Virtual machine with a Gigabit internet connection, 2Gb of memory and a 40Gb solid state drive, running Ubuntu 16.04, which would enable me to produce the graphs. Some sample GNUPlot code (used to generate the overall graph for round 1 - Figure 8) is below:

```
1.  set terminal png size 2000,1414
2.  set output 'output.png'
3.  set title "C2 Project - Round 1 Data"
4.  set ylabel "Position (relative)"
5.  set xlabel "Time (hours)"
6.  plot "../data/round1.dat" using 1:2 title 'X', \
7.       "../data/round1.dat" using 1:3 title 'Y', \
8.       "../data/round1.dat" using 1:4 title 'Z'
```

Data Presentation

Data Tables

As we collected 11,398,119 data points, reproducing our data here would run to over 80,000 sides of A4, so I have only produced extracts of the data below

```
START,
4981,230,4230,-18
5177,234,4220,-22
5371,238,4232,-24
5567,234,4238,-18
5763,236,4244,-20
5958,250,4242,-12
6154,248,4238,-36
6348,226,4236,-28
6544,230,4238,-22
6742,232,4240,-28
```

Table 1 – Extracts of raw data collected from Arduino in first round of data collection

Table 1 shows the raw data we collected off the SD card of the Arduino, with this particular data being the first 11 lines of data collection in our first round. The columns are time (in milliseconds), X position, Y position and Z position. Time is relative to when the device was powered on, and X/Y/Z positions are all relative to the MMA8451 being placed flat on its back. START indicates the that device was powered on (the device can be rebooted during data collection, and the START message is printed at the end of the file and collection continues, so data is not overwritten).

```
Hours,X,Y,Z
0.001383611,230,4230,-18
0.001438056,234,4220,-22
0.001491944,238,4232,-24
0.001546389,234,4238,-18
0.001600833,236,4244,-20
0.001655,250,4242,-12
0.001709444,248,4238,-36
0.001763333,226,4236,-28
0.001817778,230,4238,-22
0.001872778,232,4240,-28
```

Table 2 – Extracts of processed data as a CSV from first round of data collection, with time in Hours

Table 2 shows the processed data, which is the same as the raw data, except that the time has been converted into Hours, and START replaced with column headings, so that the file now conforms to the CSV format and can be used in graphing tools.

The full data is available online at https://github.com/Jbithell/C2Project

Graphs and Analysis

Higher resolution versions of all graphs are available online at https://github.com/Jbithell/C2Project

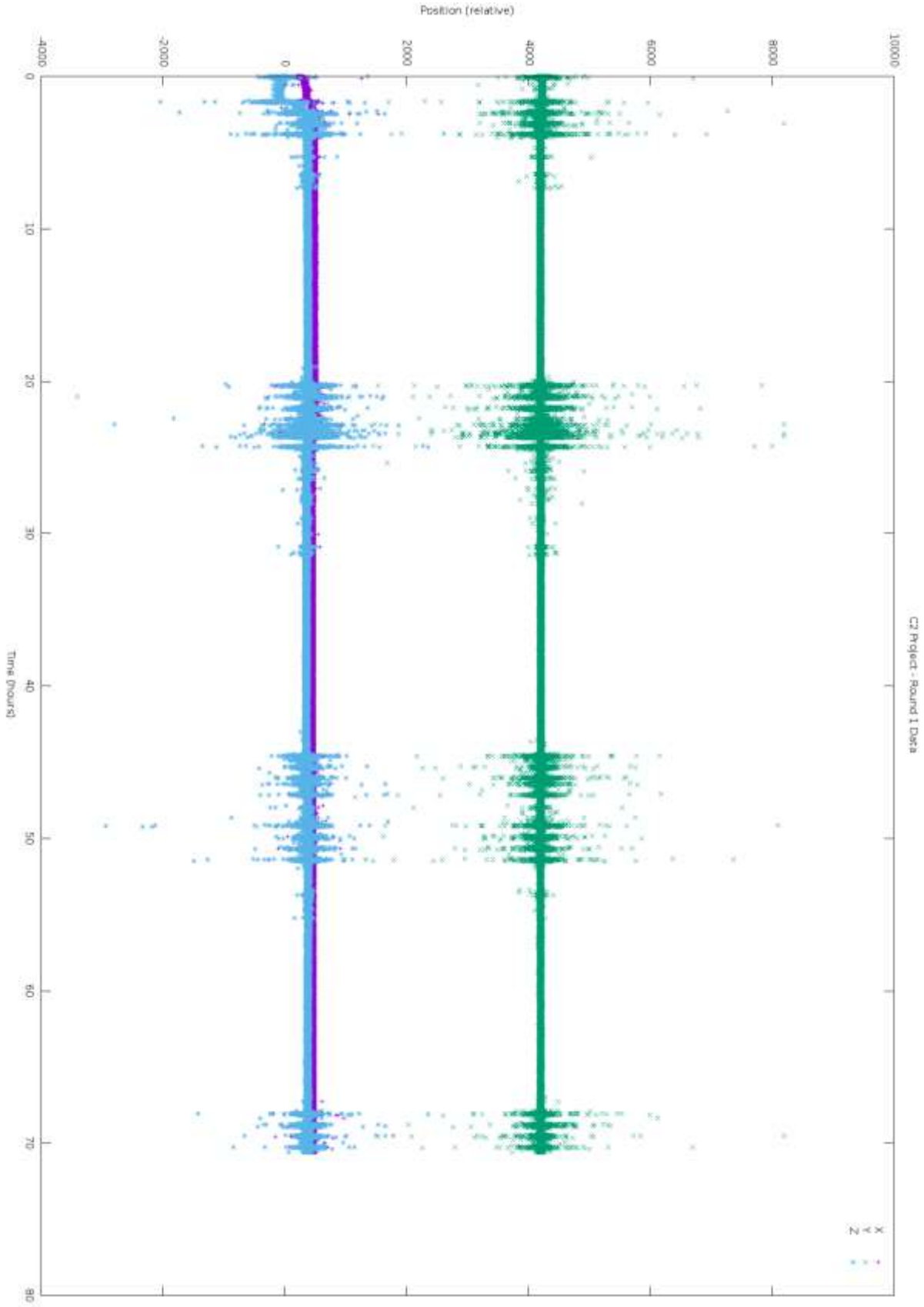

Figure 8 - Round 1 Data (in full)

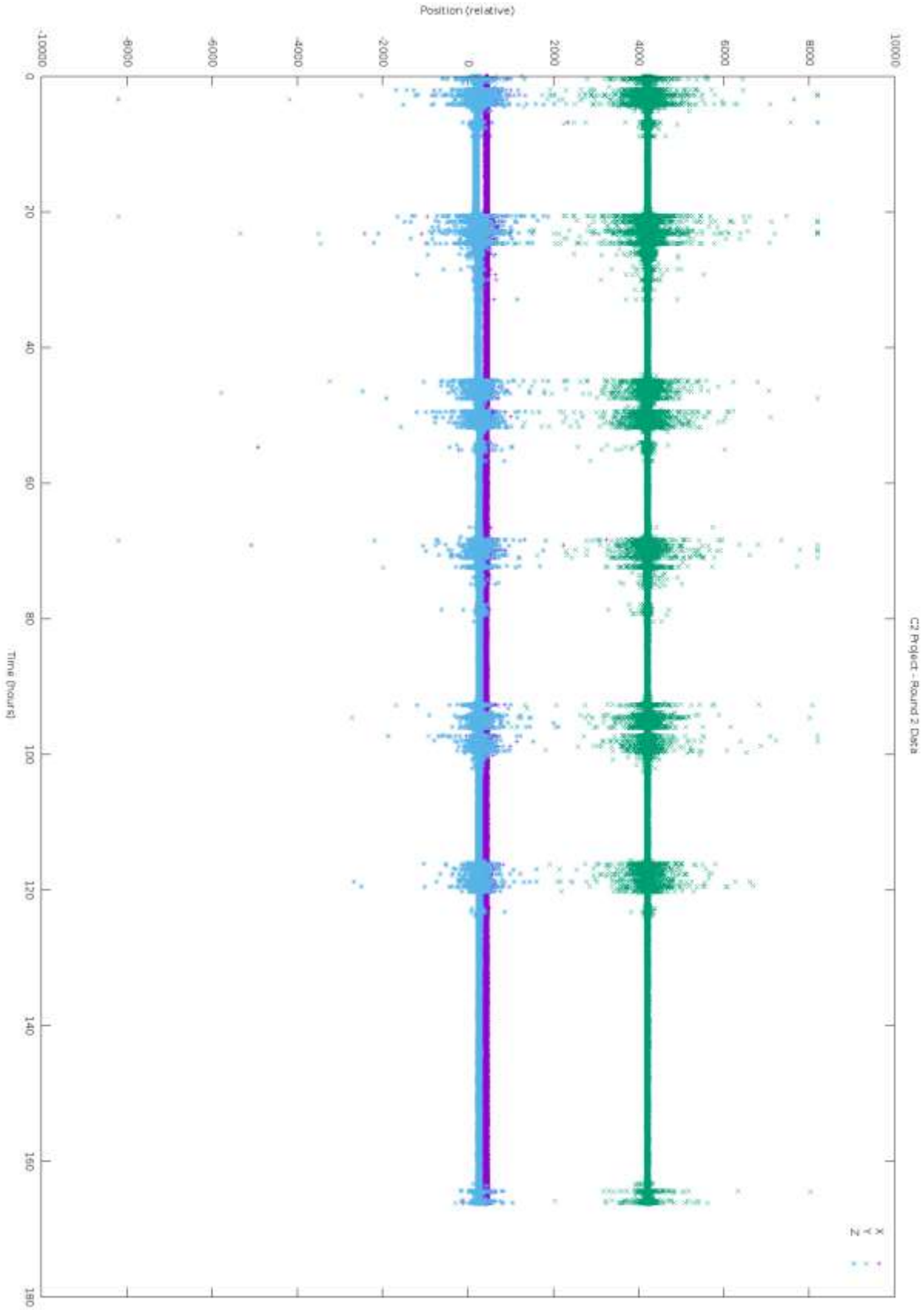

Figure 9 - Round 2 data (in full)

Figure 8 and Figure 9 show the complete data we gathered during the whole experiment, with 11.4million points between them. These graphs are useful, as they show us that during the first hour of operation in round 1, the accelerometer recorded some unusual values while it was being secured in place, and settling into its tape based mounting – but overall they are less useful because the sheer volume of points makes it difficult to drill down into the data, and examine the details of one day for example.

Figure 10 and Figure 11 show the complete data, though this time the "computed version", which is the same data, just with a function I created run against it, in attempt to simplify the data:

$$X/Y/Z\ (Constant) = Average\ (mean)\ of\ relevant\ series\ (x/y/z)\ for\ relevant\ round\ (1/2)\ between\ 23{:}00\ and\ 04{:}00\ during\ the\ first\ night\ of\ data\ collection\ in\ that\ round$$

$$f(x,y,z) = abs(X - x\ ) + abs(Y - y) + abs(Z - z) - 200$$

Only $f($ values greater than zero are plotted

Equation 1 - "Computed Values" equation

The $abs($ function ensures that any value under zero is converted to a positive number. The -200 ensures that values below 200 are not plotted. The idea behind these graphs was that they would be more useful in analysis, because less data is easier to handle from a graphing perspective – though the reality has been that they are just as difficult to use, because the high values are still grouped around each other, and appear as stripes.

Figure 10 - Round 2 - Complete Computed Data

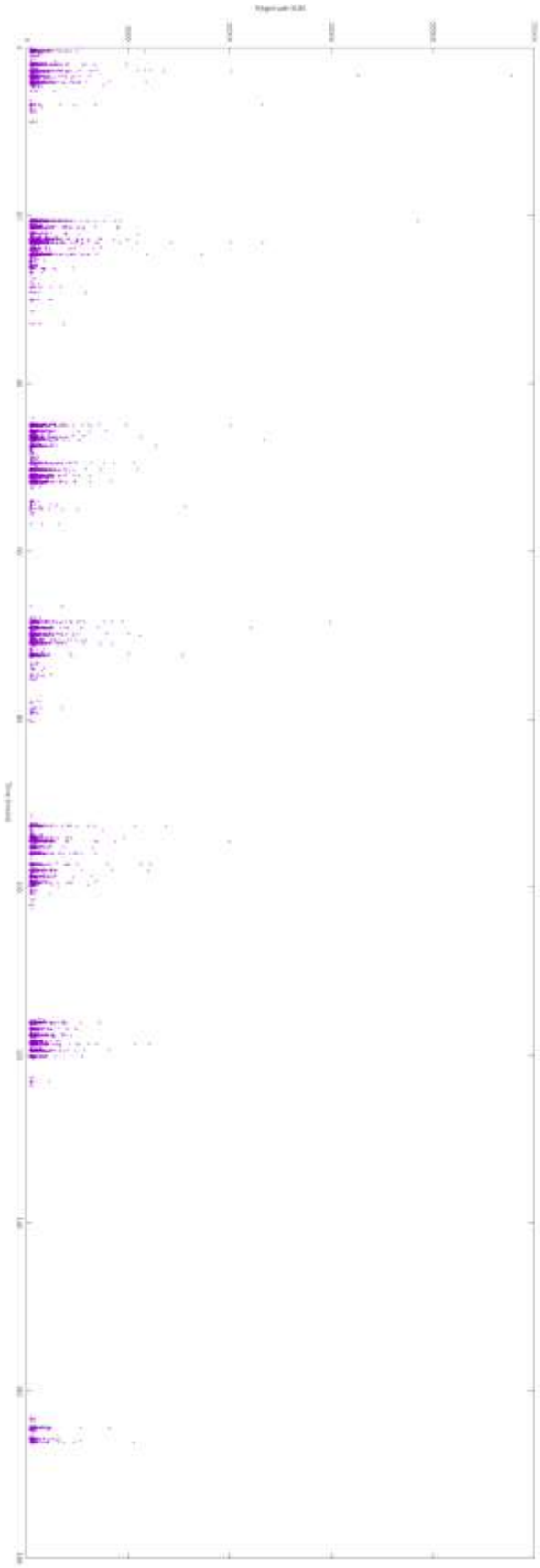

Figure 11 - Round 1 - Complete Computed Data

Figure 12 shows data for Tuesday 24th January (which is an extract from round 1's data). It uses the same data as the graphs for round 1, except it only displays the Y axis data from the accelerometer (as for these purposes, looking at magnitude of change, X/Y/Z are all very similar), and the time in hours has been converted to time of day before the graph was generated. Tuesday 24th January was a "Saturday timetable" which means the period numbers and timings were that of a normal Tuesday (5 periods), but the lessons in those periods were those of a Saturday. With kind support from the Deputy Director of Studies, Mr Tolley, we were able to obtain a copy of the timetable for Saturday in that building, from which I was able to estimate the number of pupils in classes (from the second floor upwards, due to the location of our apparatus). As accurate pupil numbers were not available, I estimated that there are 18 pupils in average lower school sets, and 12 in average upper school sets. The data (shown on the graph), shows a clear correlation between the magnitude of change in position and the number of pupils in lessons, though some of these lessons could be doubles, and therefore wouldn't be using the staircase – though the overall trend is that the larger the number of pupils using the staircase, the greater the magnitude of change in position readings. This graph is particularly exciting because it demonstrates that our accelerometer was successfully gathering the data we wanted, and the magnitude of movement data captured is comparable to what we would expect. Focusing in on the gaps between lessons for example – we can see that the majority of staircase users were on time or early for their period 1 lesson, with a minority being late – which is corroborated by our personal experience. The graph shows staircase users were far more likely to be late to their lesson if it was after another lesson, then they were if it was after break. We know this to be true, as staircase users are able to leave early for lessons after break, or at the beginning of the day, whereas for their arrival for period 3 for example, they may have had a lesson in the Weston Building, which is 525m away, and the respective classrooms could be as many as 9 flights[1] of stairs away.

The anomalous data around 12noon highlighted is most likely due to departures for sports matches, or a particular class moving to another location or conducting experiments around the building - or at a local park for example.

[1] Used here as a set of stairs between two floors.

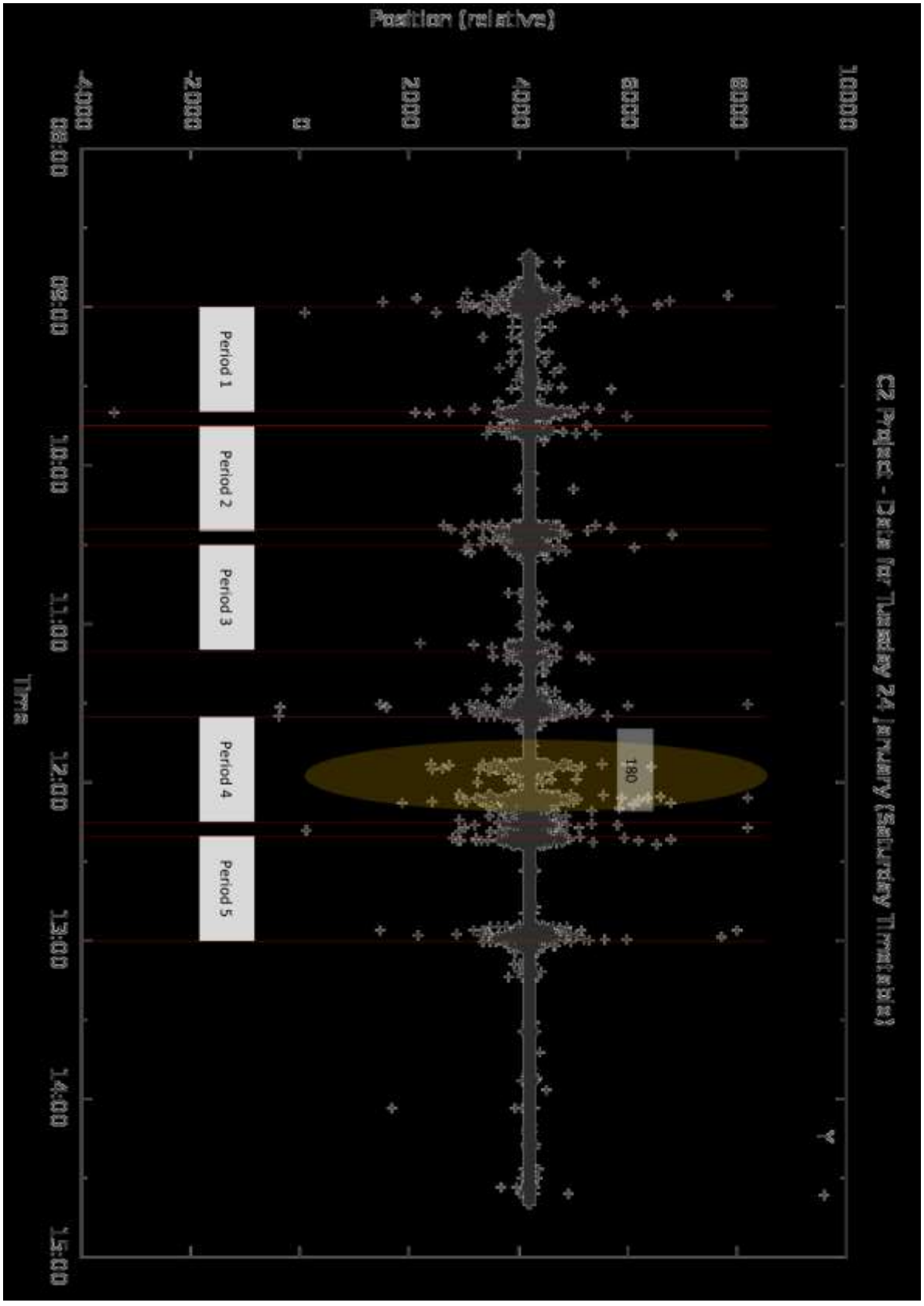

Figure 12 - Data for Tuesday 24 January (Y Axis only)

Evaluation

Data Collection

Overall our data collection strategy was broadly successful, and we were able to collect a large amount of data using the Arduino, though perhaps the greatest improvement we could have made would have been to have focusing on obtaining data in a known unit, such as g or N which would enable us to quantify the amount of movement resulting from someone going up the stairs, and therefore we could quantify the number of people going up the stairs at any given time.

We also faced the significant challenge of getting the BBC Micro bit to work, which, though we believed was the right course of action at the time, in the end left us with less time available to develop the Arduino solution as a plan B, and thus we were unable to implement some features we wanted to, such as a quantifiable measure of movement, as opposed to just a relative scale of movement which is what we ended up with.

If we were doing this project again, I would devote more time to researching SD card and data storage technologies on the chosen device, in this case the BBC Micro bit, before working to produce code for the accelerometer, which was eventually rendered defunct as we were unable to produce an SD card reader library for the device. I would also perhaps try and run the device off mains power, instead of a battery, so that we could gather more data over a number of weeks, but this might have meant putting it in a less ideal location.

Overall though, I conclude that we were able to gather useful and worthwhile data, with the Arduino, in great enough detail to be able to perform analysis.

Data Presentation

Presenting the data has been the most challenging and enjoyable section of this project. Working with such a large number of data points was considerably difficult, and meant learning a new technology, GNUPlot, simply to parse and display the data in the graphs in the preceding sections.

If I was to repeat this project, I would focus on further parsing at data collection level, for example instead of recording the position of each of the three axis at every 10th of a second, I would have instead recorded the magnitude of change in position, whenever it exceeded a certain value – up to every 10th of a second, reducing the volume of data collected, and requiring less parsing and manipulation after collection from the measuring device. I might also try and add in other data points, such as a manual footfall per minute count taken by eye to compare and evaluate our accelerometer device.

Concluding Statement

In conclusion, using a three axis accelerometer, linked to an Arduino and SD card, powered by a battery, we were able to gather 236 hours of data on the relative positions of each of the three axis, which matched the expected magnitude of movement based on timetable data, of the device located under the stair tread on the first floor of our school's science centre. Therefore, we were able to, with some accuracy, measure footfall with under £40 worth of electronic components, a task that would normally require either an IR based data logger, or a manual surveying team at a much higher cost – though our data is influenced by the individual style in which people negotiate the stairs – with those who jump perhaps giving a greater reading than two people who walk softly.

Appendix

Arduino Code

```
/* ###############################################################
 * Arduino Accelerometer Project
 * 06/02/2017
 * ---------------------------------------------------------
 * James Bithell & William Ashton
 * C2 Physics project
 *
 * ----------------------LCD-------------------------------
 * LCD RS pin to digital pin 7
 * LCD Enable pin to digital pin 6
 * LCD D4 pin to digital pin 5
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 3
 * LCD D7 pin to digital pin 2
 * LCD R/W pin to ground
 * 10K variable resistor: ends to +5V and ground and wiper to LCD VO pin (pin 3)
 *
 * ---------------------SD CARD-----------------------------
 * SD card attached to SPI bus as follows:
 * MOSI   -   pin 11
 * MISO   -   pin 12
 * CLK    -   pin 13 - also SD operation LED
 * CS     -   pin 10
 *
 * -------------------ACCELEROMTER--------------------------
 * SCL    -   analogue pin 5
 * SDA    -   analogue pin 4
 *
*/


#include <Wire.h>
#include <Adafruit_MMA8451.h>
#include <Adafruit_Sensor.h>
#include <LiquidCrystal.h>
#include <SD.h>

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
// initialize the LCD library with the numbers of the interface pins
// float V = 0; // Computed voltage from input pin 0 AD converter
unsigned long Time = 0; // Becomes number of seconds since power up

String dataString = "";
Adafruit_MMA8451 mma = Adafruit_MMA8451();

void setup(void) {
  //Setup Accelerometer
  Serial.begin(9600);

  Serial.println("Adafruit MMA8451 test!");

  if (! mma.begin()) {
    Serial.println("Couldnt start");
    while (1);
  }
  Serial.println("MMA8451 found!");

  mma.setRange(MMA8451_RANGE_2_G);

  Serial.print("Range = "); Serial.print(2 << mma.getRange());
  Serial.println("G");
```

```
62.
63.
64.   //Setup Screen
65.   pinMode(10,OUTPUT); //CS pin on the SD
66.   Serial.begin(9600);
67.   lcd.begin(16, 2);
68.   lcd.print("      JBWA");
69.   lcd.setCursor(0,1);
70.   lcd.print("     Logger");
71.   delay(2000); //wait two seconds
72.   lcd.clear(); //clear LCD
73.
74.   if (SD.begin(10)) {    // see if the card is present initialize
75.       dataString = "START,"; // make a string for assembling the data to log:
76.       File dataFile = SD.open("data.txt", FILE_WRITE); // open the file.
77.         if (dataFile) { // if the file is available, write to it:
78.         dataFile.println(dataString);
79.         dataFile.close();
80.         lcd.print("Micro SD Card"); //and display a message
81.         lcd.setCursor(0,1);
82.         lcd.print("initialised");
83.         delay(2000); //wait two seconds
84.         lcd.clear(); //clear LCD
85.       }
86.   }
87. }
88.
89. void loop() {
90.   mma.read();
91.   sensors_event_t event;
92.   mma.getEvent(&event);
93.
94.   if (millis()/100  > Time) { //Do everything below once a 1/10th of a second
95.     Time = millis()/100; //Time in tenths of seconds since power up
96.     dataString = "" ;
97.
98.     lcd.clear();
99.     lcd.setCursor(0,0);
100.          lcd.print("Logging | ");
101.          lcd.print(mma.x);
102.          lcd.setCursor(0,1);
103.          lcd.print(mma.y);
104.          lcd.print(" | ");
105.          lcd.print(mma.z);
106.          //It's best to plot the sum of the three values as it gives you the most
      sensible data with some jitter
107.          dataString += millis();
108.          dataString += ",";
109.          dataString += mma.x;
110.          dataString += ",";
111.          dataString += mma.y;
112.          dataString += ",";
113.          dataString += mma.z;
114.          dataString += "\r\n";
115.
116.          File dataFile = SD.open("data.txt", FILE_WRITE);
117.          if (dataFile) {
118.            dataFile.print(dataString);
119.            dataFile.close();
120.          }
121.        }
122.
123.      }
```

Table of Figures

Table of Tables

Table of Equations