Hochschule Bremerhaven

Acceleration

Lab report

Submitted to: Prof. Dr. rer.nat. Axel bochert.

Submission date	04-04-2016
Submitted By	Matriculation Number
ALI HASAN TALHA	33433
MUDASSIR NAWAZ	33419

CONTENTS

1 ACKNOWLEDGEMENT

We would like to thank Professor Dr. rer.nat. Axel Bochert for his continuous support and guidance. We are highly indebted for giving us his time and advice. We would also like to thank everyone working at Maritime Laboratory Hochschule Bremerhaven for helping us carry out the experiments.

2 INTRODUCTION

In this experiment a mobile circuit is assembled to measure acceleration using sensor module [1], and with this measured acceleration calculate the distance travelled. The mobile unit consists of a microcontroller unit (MCU) based on Atmel Atmega32 [2] and sensor interface board. The sensor interface is composed of low pass filters, amplifiers, power unit and the dynamic acceleration sensor module GY-61. The data flow diagram below explains it further:

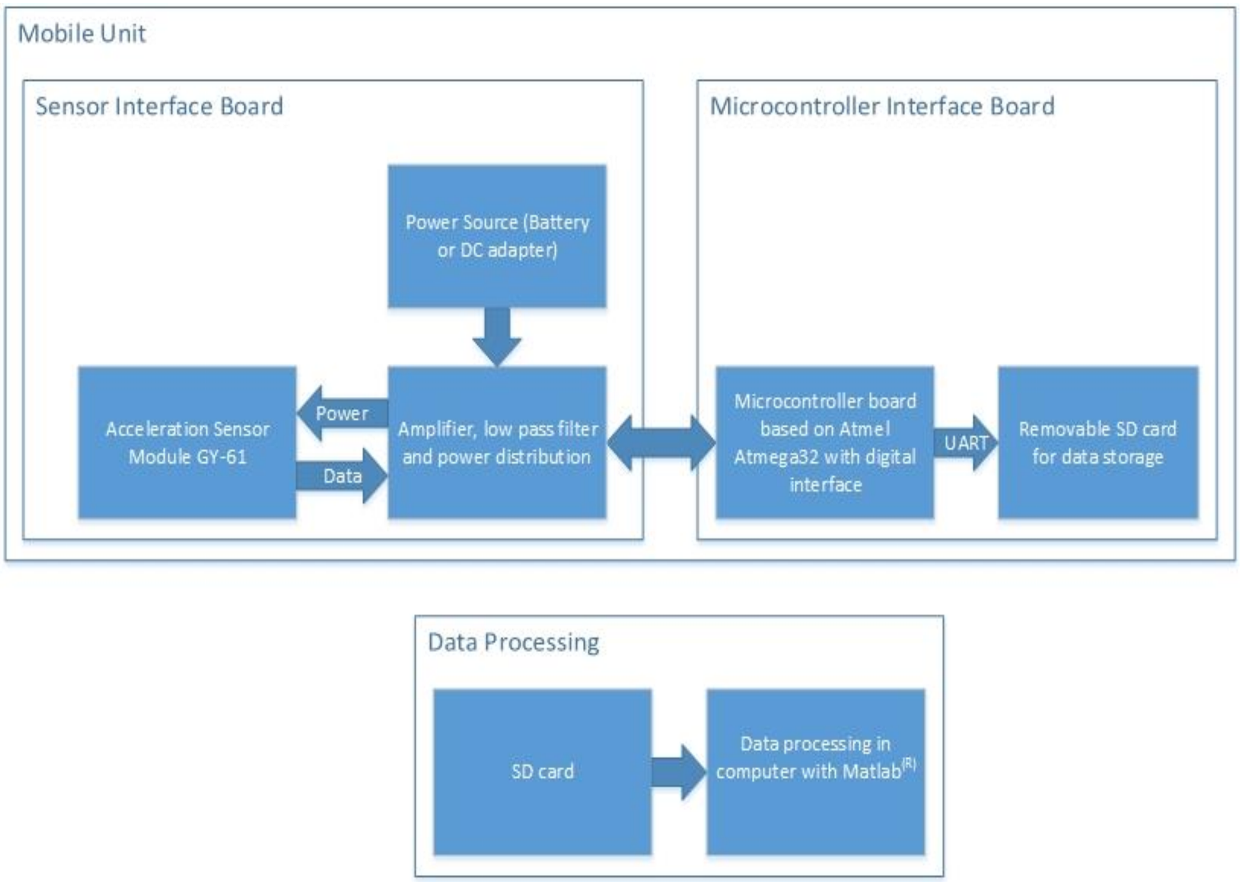


Figure 1 Power and data flow diagram

The main purpose of this experiment is to perform acceleration measurements in an elevator and calculate acceleration, velocity and distance travelled by the elevator. Multiple trips are performed with calibrations and digital data is stored in an SD card. This data is then processed on a standalone computer.

3 SENSOR INTERFACE BOARD:

The sensor interface board consists of amplifier, low pass filter power control unit and acceleration sensor. The sensor module is based on 'Analog devices ADXL335' [3] and operates in a voltage range of +1.8 V to +3.6 V and outputs a voltage according to the following formula [1]:

$$U_s = U_{off} + (g + a_d) \times S \qquad (2.1)$$

Where

U_s is the output voltage

U_{off} is output voltage at 0g.

g is static acceleration, g = 9.8 m/s^2

a_d is dynamic acceleration.

S is Sensitivity of sensor typical value = 300 mV/g

The value of 'U_s' needs to be mapped within the range of 0 to 5V so that it can be accessed by the analog to digital convertor on the MCU board, for this purpose an amplifier or attenuator is designed using simple op-amp configuration. The adjusted output is then filtered out using low pass filter to remove noise and get a stable output.

Schematic below gives a graphical view of the sensor interface [1]:

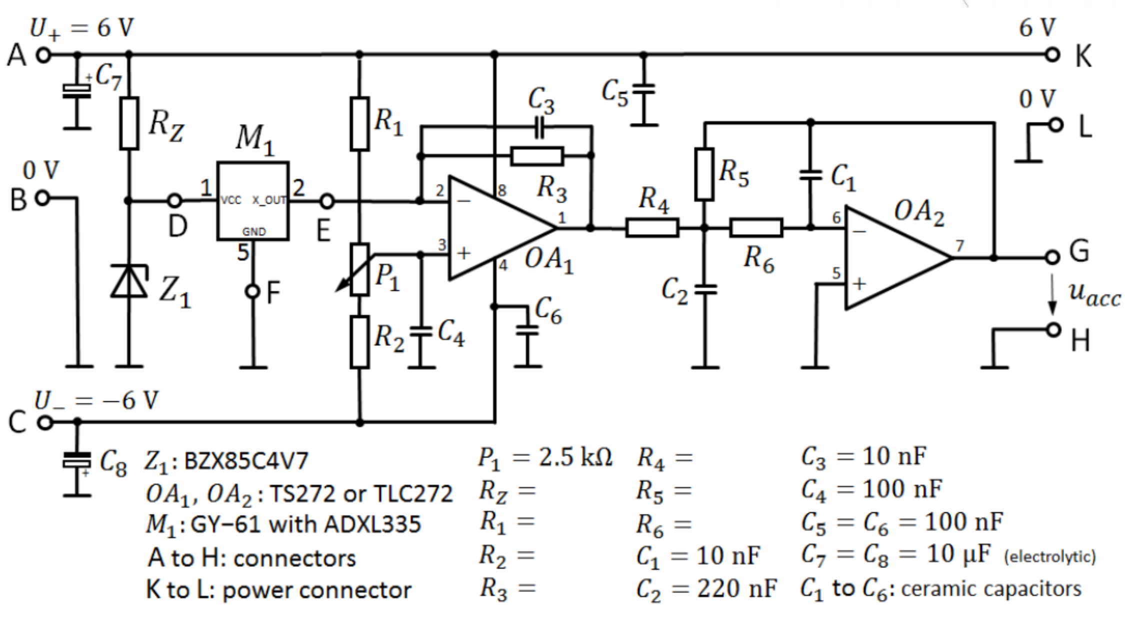


Figure 2.1 Schematics of the sensor interface module and operational amplifiers [1]

R_1 = 12 kΩ	R_2 = 15 kΩ	R_Z = 33 Ω
R_3 = 47 kΩ	R_4 = 68 kΩ	
R_5 = 390 kΩ	R_6 = 270 kΩ	

4 LOW-PASS FILTER:

Low pass filter is used to limit noise and obtain a smooth output. The cut-off frequency is set at 10Hz. Two methods were used to calculate the cut-off frequency. In the first method rise time was measured for a DC signal (square wave) and cut-off frequency was calculated by using equation from [1] :

$$f_c = \frac{1}{3 \times t_s} \tag{3.1}$$

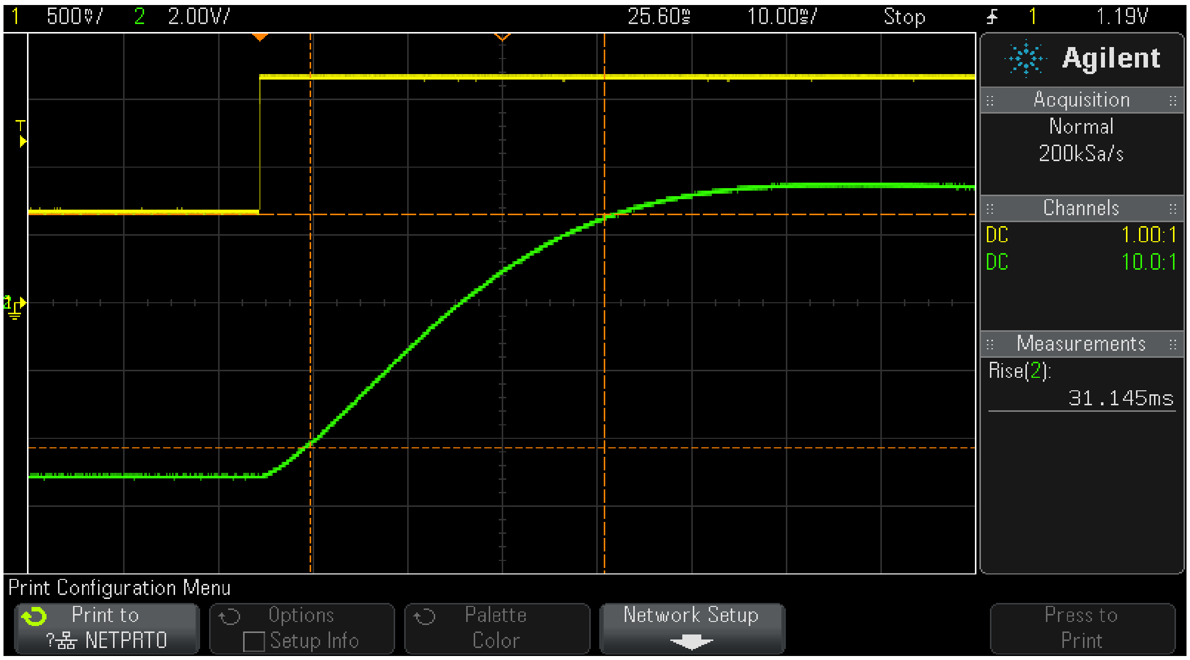


Figure 3.1 Step Response of Low pass filter as displayed on the digital oscilloscope

The rise time is measured directly by the oscilloscope and given as 31.145ms, using equation (3.1) cut-off frequency of 10.7043Hz is calculated.

In second method a range of frequencies were applied as input signal to the filter and the response was recorded. Cut-off frequency is the frequency at which the output amplitude is reduced to 0.707^{th} of the applied input signal. The frequency is calculated to be 13.1Hz. In this particular experiment a sine wave of 1 volts amplitude was applied. As seen in the following figure output voltage reaches 0.707 volts at a frequency of 13.1 Hz, which is the cut – off frequency (f_c).

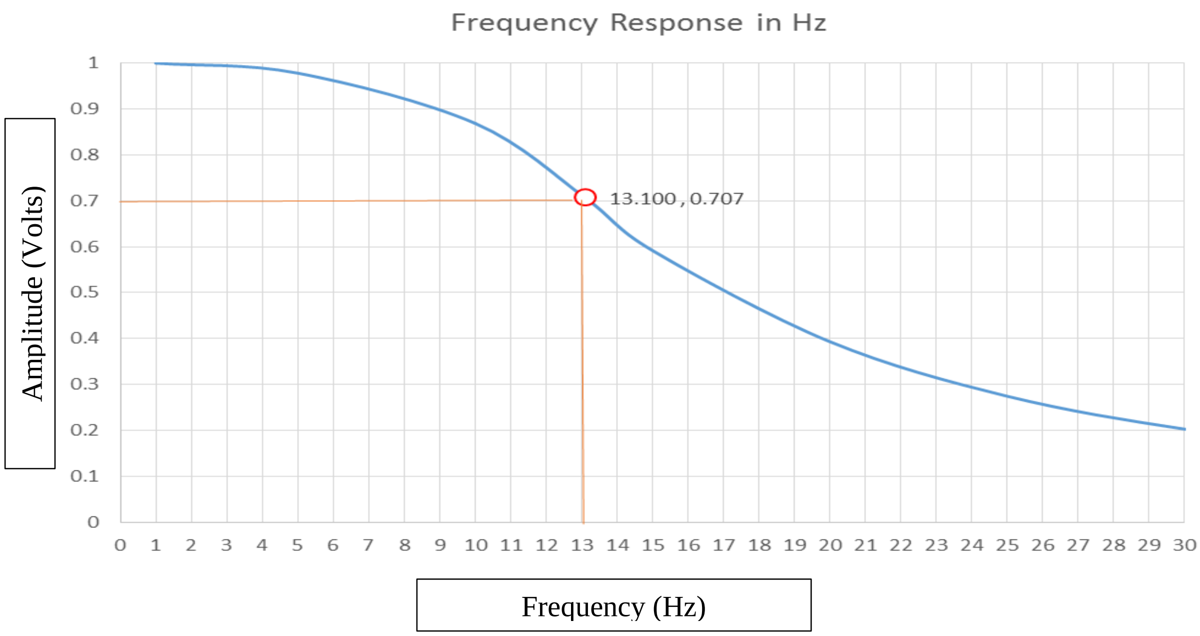


Figure 3.2 Frequency response of low pass filter

5 EXPERIMENT:

The mobile unit powered up by batteries is taken in elevator to perform multiple trips between the same floors and acceleration is measured. The mobile unit is calibrated before every round trip. The values of acceleration are recorded in a text file format in a removable storage which is then post processed in MATLAB® [4] and Simulink® [4] to calculate acceleration, velocity of the elevator and distance travelled by the elevator between the two floors. Raw data from the mobile unit is as follows:

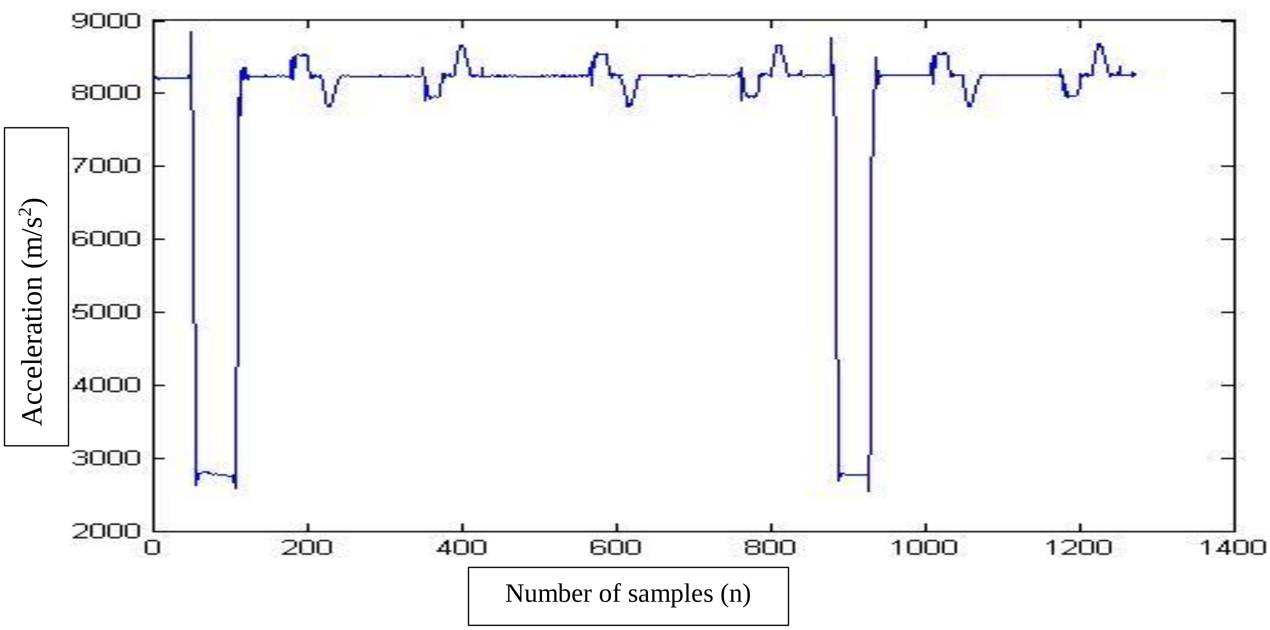


Figure 4.1 Raw data from the SD card

The high peaks and low depressions are the result of calibrations where the mobile unit measures data against a value of no acceleration 0*g* and 1*g*. In practice the acceleration sensor is placed perpendicular to the direction of gravitational acceleration for 0 g or 0 m/s^2 acceleration. Similarly to get the value at 1 g or 9.8 m/s^2 the sensor is placed parallel to the surface of elevator. Both these calibrations are done when the elevator is at a standstill.

Acceleration is given by the following equation [1]

$$a = k_1 \times d_{acc} + k_0 \qquad (4.1)$$

d_{acc0} = 2776 m/s^2 d_{acc1} = 8221 m/s^2

K_1= 0.0018 unit less K_0= -4.9968 unit less

The acceleration calculated from the formula above includes a static acceleration of 1*g* (9.8m/s^2). This static acceleration is removed from the result to obtain dynamic acceleration. Acceleration is shown in the plot below:

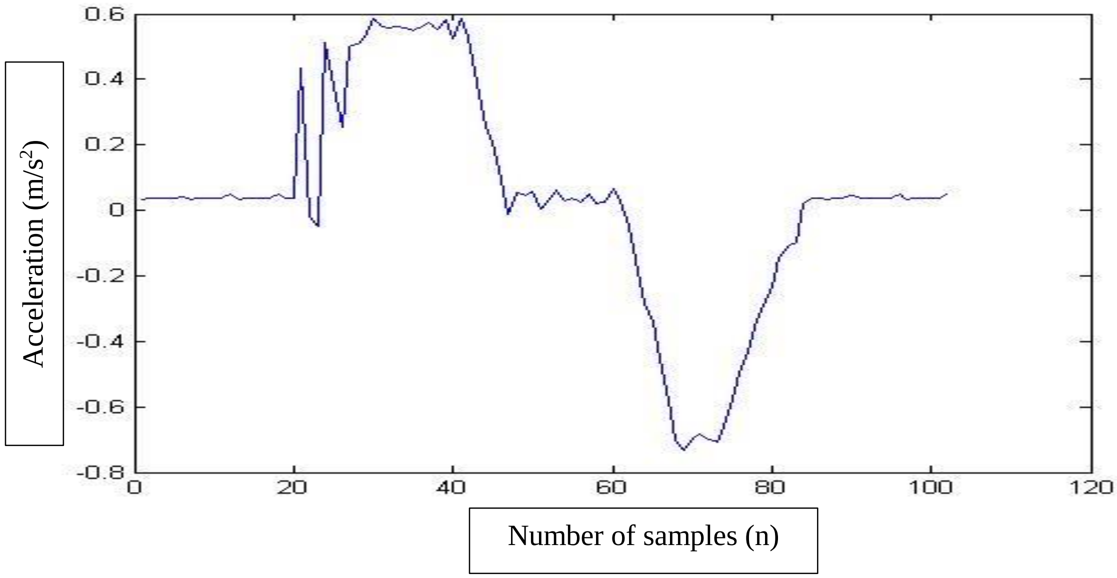


Figure 4.2 Acceleration of one trip is calculated

The calculated acceleration can be used to find out the velocity of elevator by integration but integration in digital signal processing equals to a summation. By definition acceleration is the rate of change of velocity, mathematically

$$a = dv/dt \tag{4.2}$$

Or

$$V = \int_0^T a\,dt \tag{4.3}$$

In continuous time processing integrating acceleration over a period of time (area under the curve) gives us the velocity. In digital domain we convert this using Fourier transforms and according to Fourier transforms an integration in time domain becomes a summation over n intervals. The equation (4.3) changes in Fourier domain as follows

$$V = \sum_{n=0}^{n=N-1} a[n] \tag{4.4}$$

Where n is the sample number and N represents the total number of samples. In theory the intervals can be unlimited but in practice we are limited by the sampling limit; which in our case is 10 Hz.

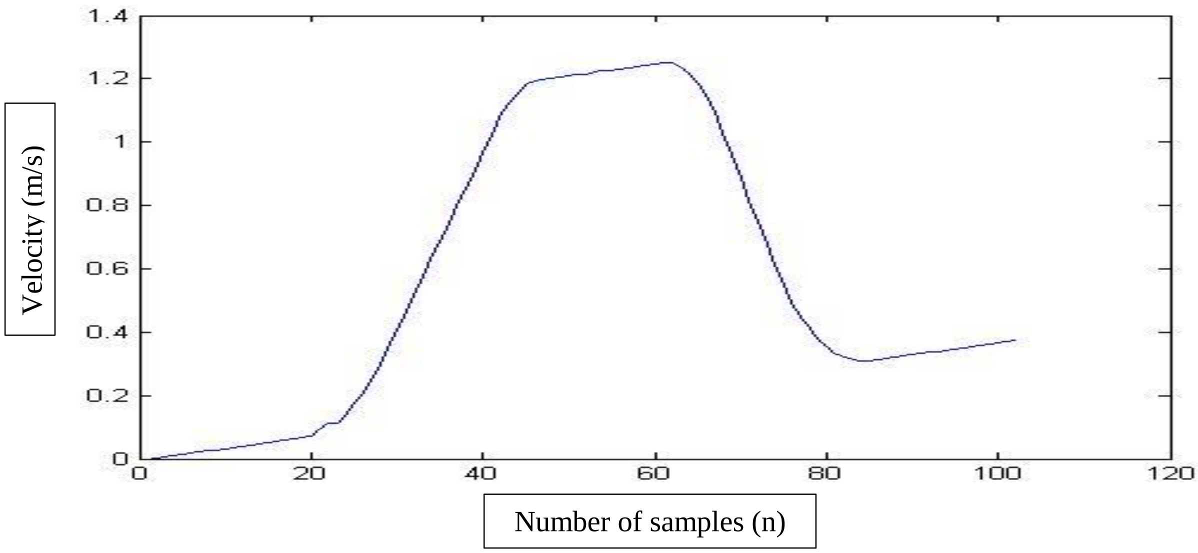


Figure 4.3 Velocity is calculated by integrating acceleration

As visible in the graph the final velocity of elevator does not return to ‘0 level’ on x-axis which is a summation error. Errors are accumulated when acceleration is integrated. This can be removed easily by calculating slope of error and subtracting from the plot.

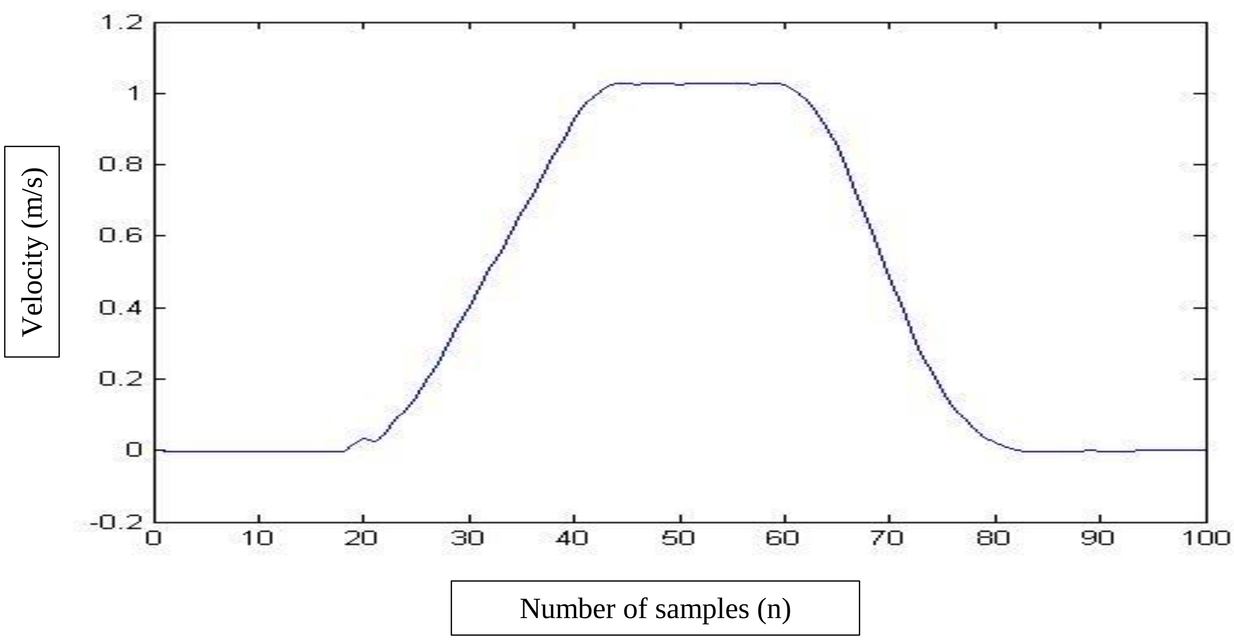


Figure 4.4 Integration error is removed from velocity

Distance is calculated in the same way velocity is calculated from acceleration i.e. by integration

$$S = \int_0^T V dt \tag{4.5}$$

Equation (4.5) is valid for continuous time domain and for digital processing we use the Fourier counterpart.

$$S = \sum_{n=1}^{N} V[n] \tag{4.6}$$

Where n is the sample number and N is the total number of samples.

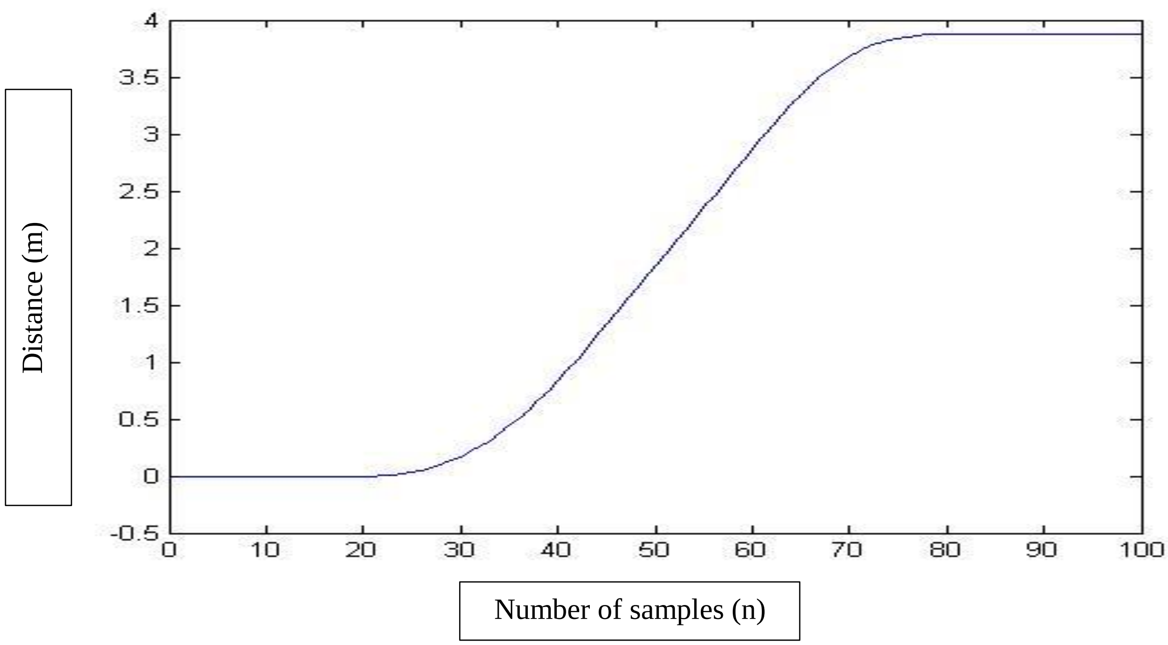
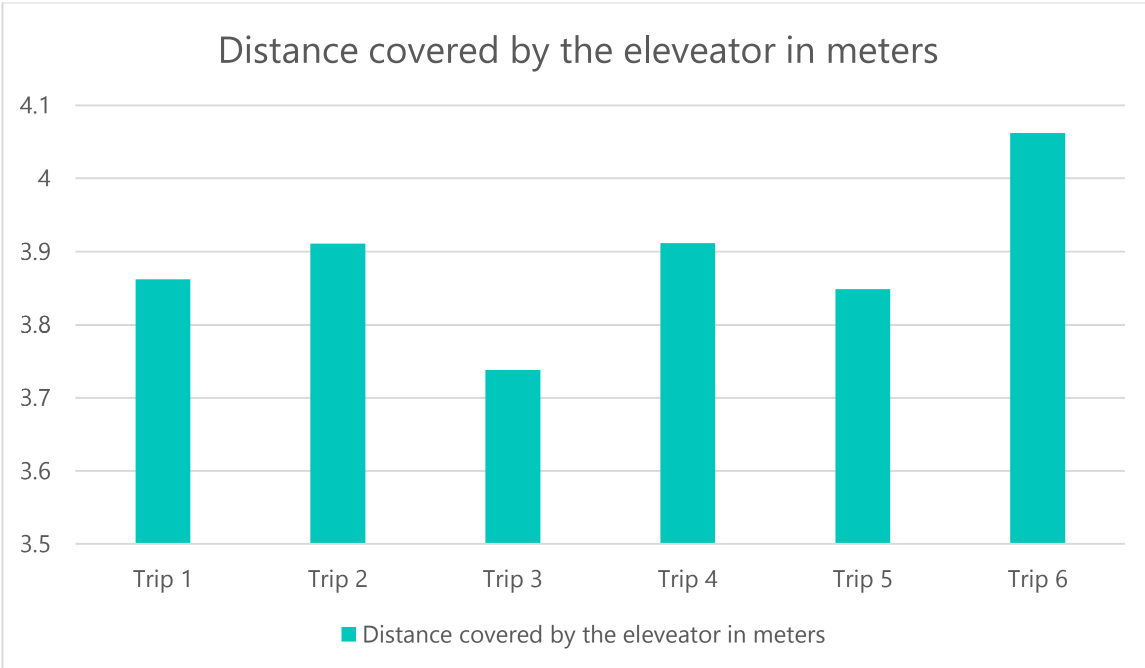


Figure 4.5 Distance is calculated by integrating velocity

6 RESULTS AND CONCLUSION:

After performing six round trips between two floors following values of distance are calculated:

Trips	Distances
1	3.8621 m
2	3.9108 m
3	3.7378 m
4	3.9115 m
5	3.8485 m
6	4.0621 m



Standard deviation is **<u>0.1060 meters</u>**.
Average distance travelled by the elevator between two floors is **<u>3.8888 meters</u>**.

In our opinion the elevator travels the same distance in every trip, the difference in values calculated are due to the accuracy limitation of the equipment used. Different factors can affect the measurements, some of which are: temperature, altitude, humidity and noise.

7 REFERENCES

[1] A. Bochert, "Laboratory Experiment No. 2: Acceleration," HS Bremerhaven, 2015.

[2] Atmel Corporation, Datasheet Atmega32; 8-bit AVR Microcontroller with 32Kbyte InSystem Programmable Flash, Rev 2503Q–AVR–02/11 ed., San Jose: Atmel Corporation, 2011.

[3] A. Devices, "Data Specification ADXL335: Small, Low Power,3-Axis ±3 g, Accelerometer, RevB," Analog Devices, Norwood, 2010.

[4] Mathworks, "MATLAB and SIMULINK". United States of America Patent www.mathworks.com/patents, 2013.

8 APPENDIX

8.1 MATLAB® CODE M-FILE

```
% Maritime engineering LAB 2 Acceleration
% read data from log file, extract data and post process to calculate the
% acceleration velocity and distance

% to be placed in the same folder as Aufzug.txt file
close all
clc
clear all
dat =dlmread('Aufzug.txt');                  %reads contents of LOG file

figure
plot(dat)                                    %plots raw data
xlabel ({'time (s)'})
ylabel ({'acceleration (m/s/s)'})

                                             % A section of graph taken for
sens(1,:) = dat(160:260);                    % post processing
sens(2,:) = dat(320:420);
sens(3,:) = dat(540:640);
sens(4,:) = dat(730:830);
sens(5,:) = dat(990:1090);
sens(6,:) = dat(1154:1254);
distance =[0,0,0,0,0,0].';

for n = 1:6

    for k = 1:length(sens(n,:))
    acc(n,k) = 0.0018*sens(n,k)-4.9968;      % Acceleration is calculated
    end

                                             %Dynamic Acceleration is
    for k = 1:length(acc(n,:))               %calculated by subtracting 9.8
    acc_corr(n,k) = acc(n,k) - 9.8;
    end

    vel(n,:) = cumtrapz(acc_corr(n,:));      %velocity calculation
    vel(n,:) =vel(n,:)*0.1;                  %calculated velocity divided



    m = (vel(n,101)-vel(n,2))/(100-1);           %Integration error slope
    x = 2:101;                                   %is calculated and removed
    y = m*x;                                     %from velocity to get
    vel2(n,:)= vel(n,2:101)-y;                   %correct velocity
```

```
    dist(n,:) =cumtrapz(vel2(n,:));             %Distance calculation
    dist(n,:) =dist(n,:) *0.1;


    distance(n,:) = abs(trapz(vel2(n,:))*0.1)
end

figure
plot(acc_corr(1,:))                             % Acceleration plot
xlabel ({'time (s)'})
ylabel ({'acceleration (m/s/s)'})

figure

plot(vel(1,:))                                 .
xlabel ({'time (s)'})
ylabel ({'velocity (m/s)'})

figure
plot(vel2(1,:))                                 %velocity without error plot
xlabel ({'time (s)'})
ylabel ({'velocity (m/s)'})

figure
plot(dist(1,:))
xlabel ({'time (s)'})
ylabel ({'distance (m)'})

average =mean(distance)
standard_deviation = std(distance)
display('thank you for using Acceleration.m')
```