Maritime Systems

Laboratory Experiment 2 : Acceleration Measurement

Under the guidance of Prof. Dr. Axel Bochert

Submitted by

Mahesh Baldaniya 39392

20.04.2023

Table of Contents

List of Figures

Summary

The purpose of a recent experiment was to develop a novel approach to determine the height of floors in T building at Hochschule Bremerhaven by measuring the acceleration of an elevator. To accomplish this objective, a specialized circuit was designed to capture the acceleration data, which was then collected using a microcontroller and SD card. Multiple trips were taken in the elevator to obtain a sufficient amount of data. Analysis of the recorded data was performed using MATLAB, with the floor height being calculated based on the standard deviation of all collected values. This approach provides a reliable and efficient method to determine floor height using elevator acceleration data in a non-invasive way.

1. Introduction

The aim of this laboratory task is to measure acceleration using ADXL355 [2] accelerometer. The interfacing circuit for the sensor module was implemented using TLC272 [8] dual precision operational amplifier package and the sensor data is recorded using a micro SD card data logger. After successful acquisition and calibration of the time series sensor data, the acceleration-deceleration data of the elevator was used to calculate the distance between two floors of T building.

2. Assembly of the Circuit

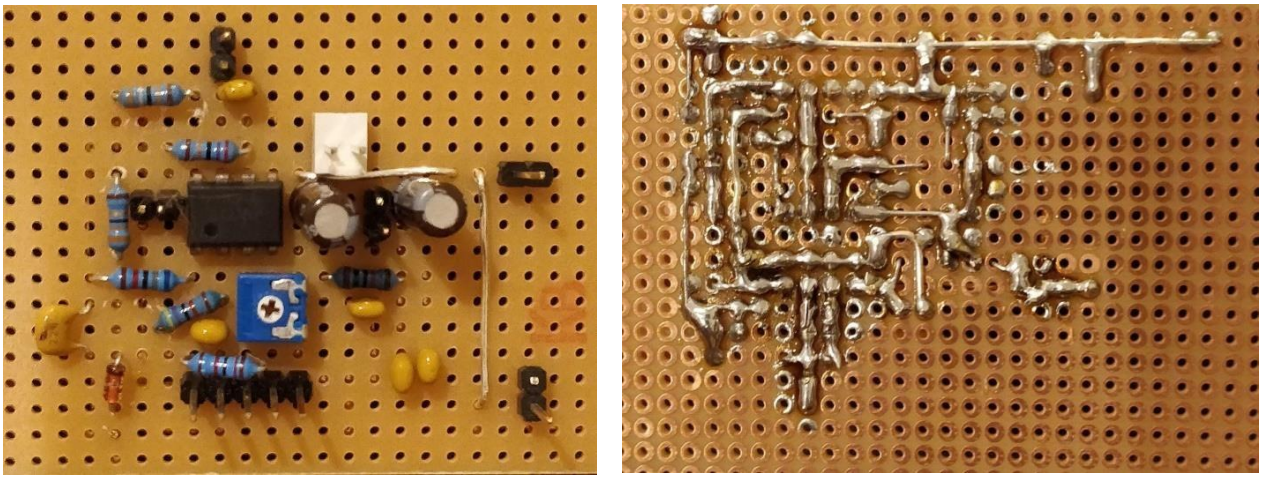

Figure 1 : Position of the components on the perf-board and the respective connections

The Figure 1 shows the constructed circuit based on the provided circuit diagram in Figure 2 as part of the problem statement. The components required to build the circuit are chosen based on calculation results and the corresponding E12 standard values. The circuit is soldered on a perf-board while taking the orientation of the sensor into consideration. After successfully soldering the circuit, it is checked for possible faults which are fixed as part of the debugging process. The circuit is provided with the necessary activation and input voltages in order to verify the output voltage range.

Figure 2 shows the provided circuit [6] used in the experiment, which comprises of two amplification stages. The first stage employs a differential amplifier, while the second stage is equipped with an active second-order low-pass filter. The filter serves to eliminate high frequency noise signals from the accelerometer, which is critical as the analog signal will be transformed into a digital signal. Power supply for the microcontroller experimental board utilized for data acquisition is provided by connectors K and L.

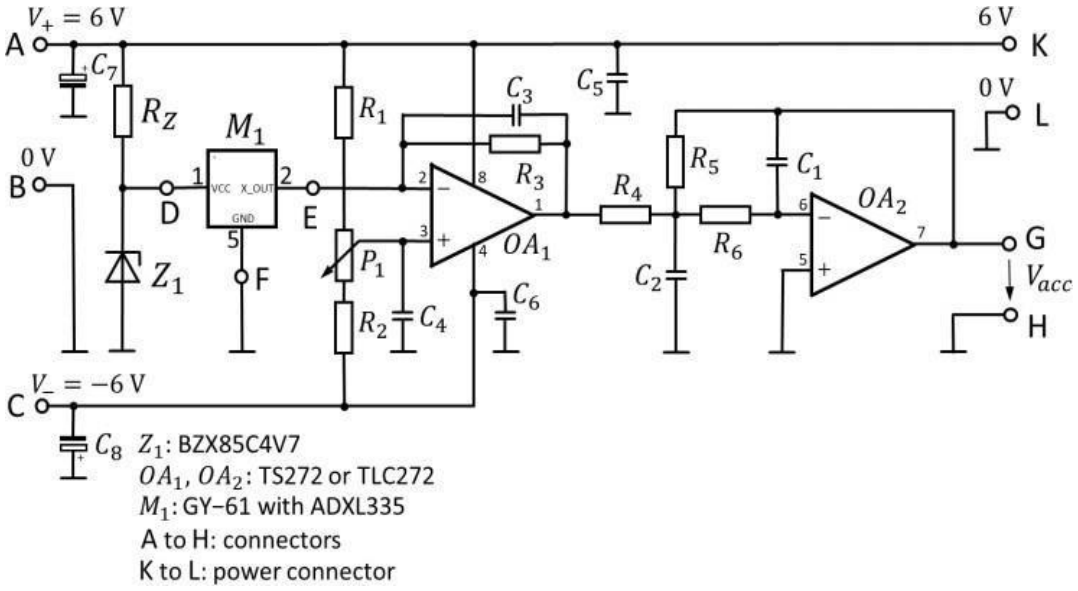

Figure 2 : Circuit diagram provided in the problem statement [6]

3. Measuring the cut-off Frequency

The methodology to measure the cut-off frequency of the filter circuit involves, varying the input signal frequency from 0 Hz to 100 Hz and observing the changes in the amplitude of the output signal. The expected behaviour is that the circuit should allow amplification till the cut-off frequency and after crossing the roll-off frequency the output signal get attenuated significantly. The exact cut-off is measured at -3 dB reduction in the signal amplitude.

3.1. Frequency Response

The following results were obtained after sweeping the input frequency in a range of 0-100 Hz and observing the amplitude values at discrete intervals. A sine wave is provided as input using an Agilent DSOX 2002A oscilloscope [5] as a wave form generator. The output is observed on the input channel 2 of the oscilloscope. The Y-axis cursor measurement mode [5] is used to determine the amplitude.

The data points from Table 2 in the appendix can be represented on a graph where the X-axis is the Frequency in Hz on a log scale and Y-axis is the peak-to-peak Voltage of the output signal on a linear scale.

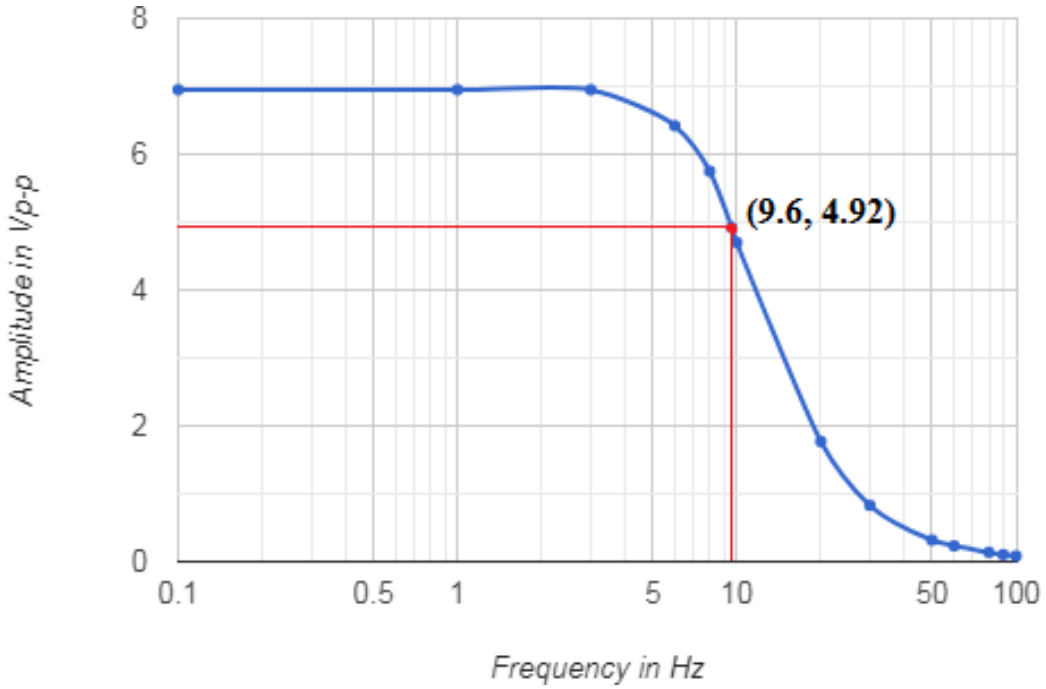

Figure 3 : Frequency response curve of the constructed circuit

In Figure 3, the line graph is used to illustrate the frequency response curve of the circuit. The blue line represents the obtained frequency response and the red line represents the virtual -3 dB amplitude reference. The roll-off frequency is measured at the respective X-coordinate of the intersection point.

The cut-off frequency at -3 dB is the frequency corresponding to the value of the maximum amplitude divided by $\sqrt{2}$ = 1.41 in equation 1.

$$A_{max} = 6.95 \ , \ A_{-3\,dB} = \frac{A_{max}}{\sqrt{2}} = 4.92\ V \quad \ldots(1)$$

Where ,

A_{max} → Maximum amplitude

$A_{-3\,dB}$ → Amplitude at $\frac{Amax}{\sqrt{2}}$

The corresponding cut-off frequency observed in Figure 3 at the -3 dB amplitude is approximately 9.6 Hz. As a result of the tolerances and the E12 standard approximation the actual cut-off frequency of the circuit is slightly lower than the ideal cut-off frequency of 10 Hz.

3.2. Step function response and Slew rate

Slew rate can also be used as another method to estimate the cut-off frequency. The slew rate is determined by observing the step function response of the circuit (Figure 4). Slew rate is defined as the maximum rate at which an amplifier can respond to an abrupt change of input level. The abrupt change at the input is provided using a step function and the resultant output response is observed. The time it takes for the output to reach from 10% of its max value to 90% of its max value is the slew rate.

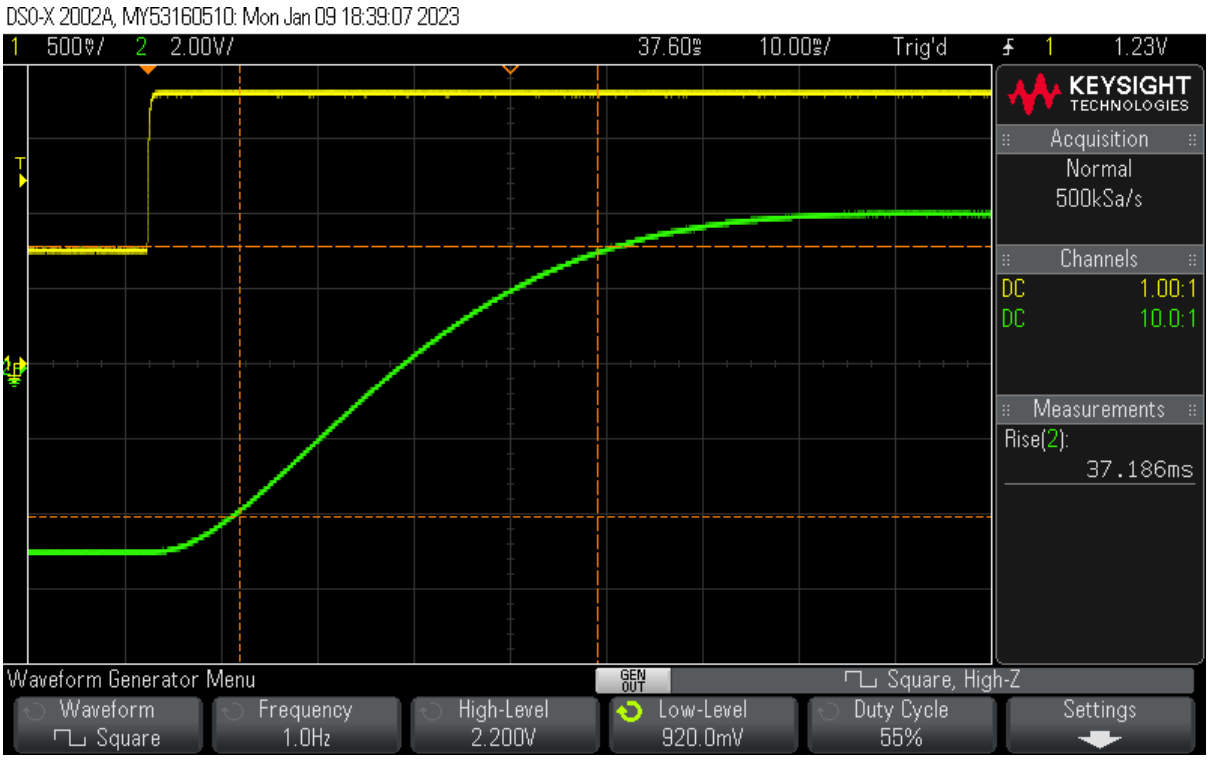

Figure 4 : Oscilloscope screen while measuring step response

t_S = 37.186 ms …(2)

where,

t_S → Slew rate

Figure 4 represents the step response curve of the circuit. The cursor measurement tool is used to measure the values at 10% and 90 % of the max value. It can be observed that the scope displays the measurement in milli-seconds as 37.186 ms.

The cut-off frequency can be approximated using the slew rate by applying the following formula :

$$f_c \approx \frac{1}{3 \cdot t_s} \quad \ldots(3)$$

Where,

f_c → cut-off frequency

Hence, the cut-off frequency calculated using the slew rate in equation 2 and equation 3 is 8.96 Hz. The inaccuracies in the placement of the cursor markers influence the measurement. [5]

4. Data acquisition with ATmega32

The custom-built development board consisting of ATmega32 [1] microcontroller is used to implement the data acquisition from the accelerometer through the interfacing circuit. The timer calculations for 100 Hz sampling frequency were carried out and the respective registers were configured. The analog values are collected using the in-built Analog-to-Digital Converter (ADC) and the output is displayed on the serial monitor via the RS232 serial interface.

Figure 5 shows the circuit schematic of the provided development board [6]. The units which were used are marked with red bounding boxes. Serial Interface port K_3 is used to for observing data on serial monitor as well as for

collection of the data using SD card data-logger. Analog Interface port K_9 is used to connect the circuit output to the ADC of the Atmega32 microcontroller. In the development phase the Digital Interface K_6 ports PD3 and PD4 were used to connect digital push buttons as inputs, in order to test the function of the code step by step.

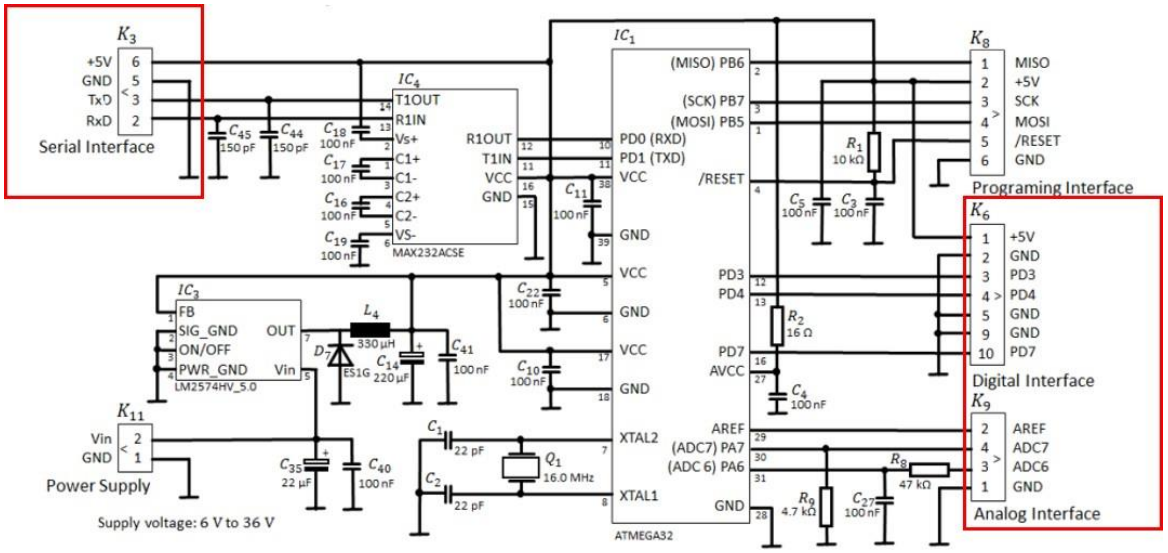

Figure 5 : Circuit Diagram of the provided development board.[6]

5. Acceleration measurements in the elevator

After checking the performance of the test circuit, multiple trips were conducted in an elevator. By means of the data recorded as time series of acceleration the height of the floors was calculated. We travel repeatedly between two floors. Evaluating the several trips we determine the accuracy of the measurement. Prior to each trip the acceleration sensor shall be calibrated. A lead acid battery serves as a portable power supply at $V_+ = 6$ V and $V_- = -6$ V for the mobile assembly. For recording of the acceleration data the RS232 to micro SD converter data logger was used.

5.1. Data recording

Before starting the experiment, the equipment needs to be calibrated. The experiment is conducted in the elevator of building T, and the test process is to move up and down three times between the first floor and the third floor. In order to obtain the data as accurately as possible, we must maintain the stability of the sensor. We fix the circuit board on a rectangular wooden block, so that we can easily rotate the sensor 90 degrees. When we turn on the device, rotate the sensor 90 degrees and then turn it back. When we start the measurement process with the test circuit and the sensor orientation, we keep the current state for roughly 5 seconds to output the data more clearly.

5.2. Calibration and Measurement

After obtaining the data, we use MATLAB to generate plot figures of all the log data. We use the parts of the data which clearly indicate acceleration and deceleration events. The resultant plot can be seen in the Figure 6 below.

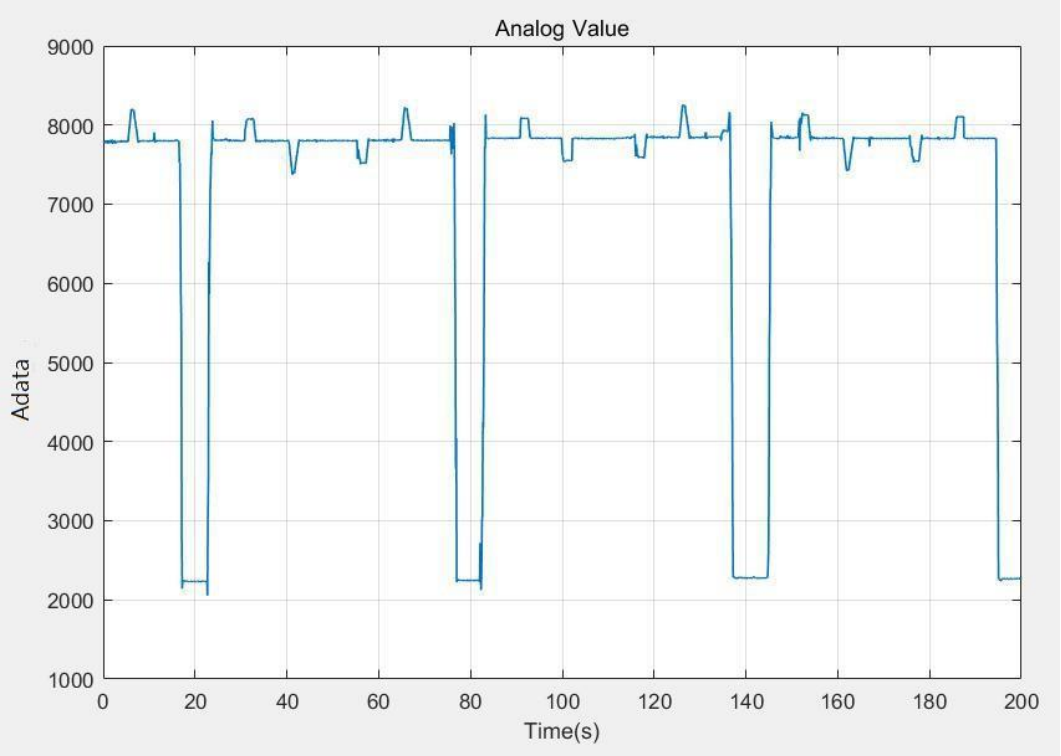

Figure 6 : Accelerometer data (Adata) extracted from the SD card data logger.

Considering the data from first elevator trip as an example to introduce the methodology. First, we choose one of the acceleration-deceleration events i.e. when the elevator goes up shown in Figure 7:

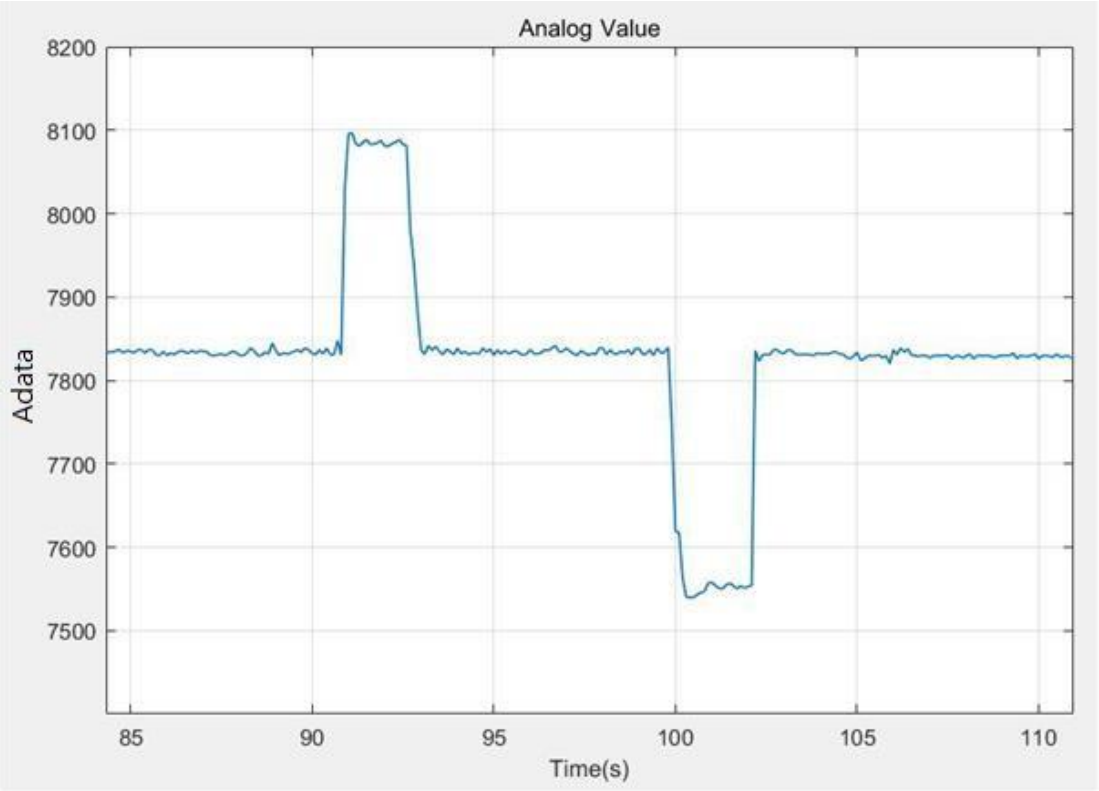

Figure 7 : Data of the third acceleration-deceleration event (elevator moving up)

The acceleration a can be calculated using the ADC values with the equation 4 where k_1 and k_0 are the calibration coefficients :

$$a = k_1 \cdot A_{data} + k_0 \quad \ldots(4)$$

Where,

a → Acceleration in m/s^2

A_{data} → Acceleration data read by ADC and recorded with data-logger

k_1, k_0 → Calibration coefficients

From the Figure 6, the lowest value is at 2234 (A_{data}) which corresponds to 0 g and the value corresponding to 1g is at 7808 (A_{data}).

Then we calculate the calibration coefficients as follows.

When acceleration a is 0 g :

$$0\ \mathrm{m/s^2} = k_1 \cdot 2234 + k_0 \quad \ldots(5)$$

When acceleration a is 1 g :

$$9.81\ \mathrm{m/s^2} = k_1 \cdot 7808 + k_0 \quad \ldots(6)$$

Using the two equations 5 and equation 6 we can get the values for k_1 and k_0 as follows :

$$k_1 = 0.00176\ \mathrm{m/s^2}$$

$$k_0 = -3.93184\ \mathrm{m/s^2}$$

Hence, we calibrate every ADC value with the calibration coefficients.

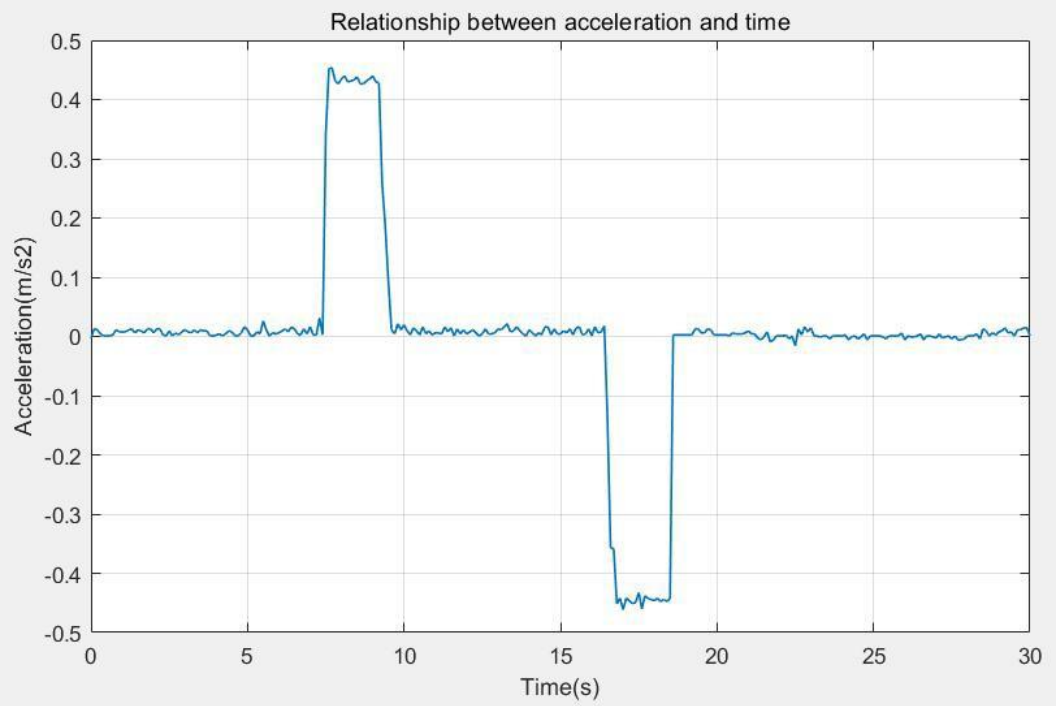

Figure 8 : Adjusting the values using the calibration coefficients k_1 and k_0

Figure 8 represents the calibrated acceleration data. In order to obtain the velocity from acceleration we use the fact that the rate of change in velocity is

equal to acceleration and get equation 7. Since we are dealing with discrete values, the data values occur at discrete time intervals. The required equation 9 is formulated using equation 8 [4] as follows:

$$a = \frac{dv}{dt} \quad \cdots (7)$$

$$v = \int_{0}^{t} a \cdot dt + v_0 \quad \cdots (8)$$

$$v_T = \sum_{i=0}^{i=T} a_i \,.\, \Delta t + v_0 \quad \ldots (9)$$

Here,

v_0 → Initial velocity is 0 m/s

Δt → Sampling interval is 0.1 s

v_T → Velocity at time T

Using the cumulative integral function „cumtrapz“ in Maltab for the calibrated data, we get the velocity curve as shown in Figure 9 below:

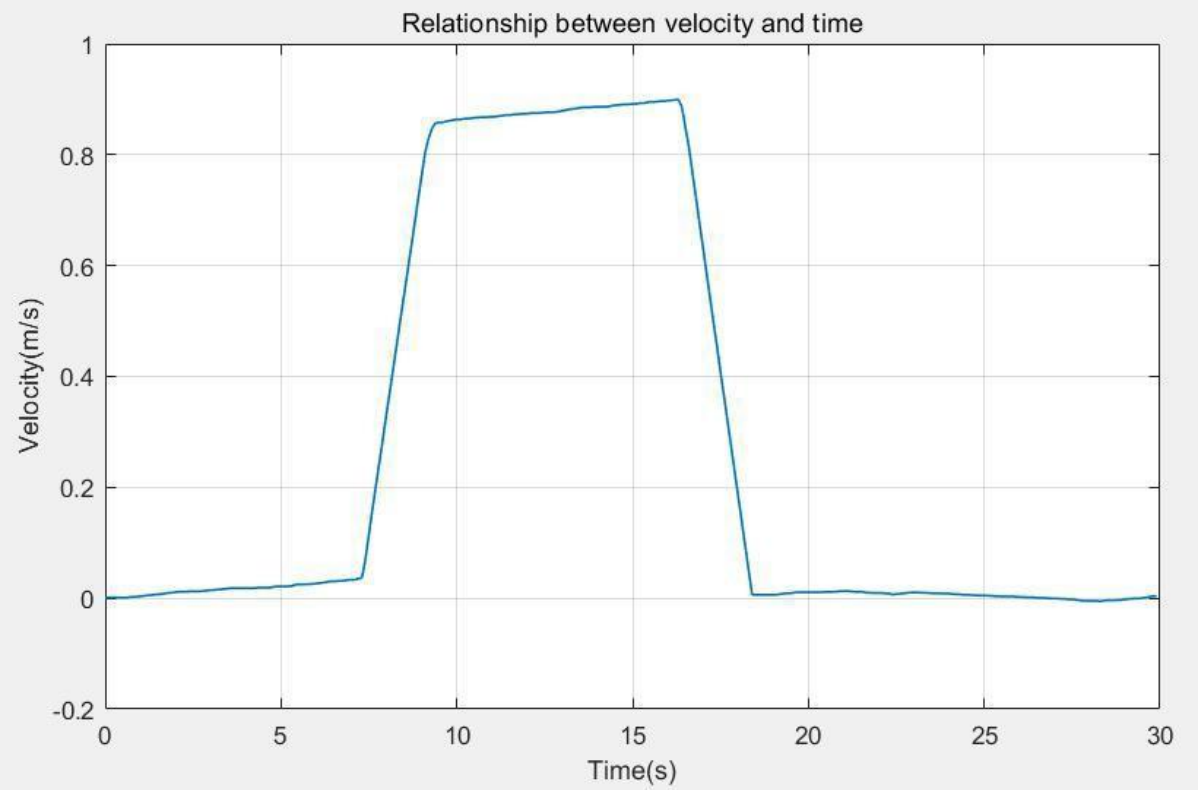

Figure 9 : Velocity curve after performing cumulative integration for discrete data

It can be observed in the above figure that the velocity 0 m/s at t = 0 s, which indicates that the elevator is almost stable. We must integrate these values further to obtain displacement curve. A similar approach is used to realize these equations 10, 11 [4] and 12 along with the fact that velocity corresponds to the rate of change in displacement with respect to time. Hence, the mathematical equations for displacement can be written as:

$$v = \frac{ds}{dt} \quad \cdots (10)$$

$$s = \int_0^t v \cdot dt + s_0 \quad \cdots (11)$$

$$s_T = \sum_{i=0}^{i=T} v_i \,.\, \Delta t + s_0 \quad \ldots (12)$$

Here,

s_0 → Initial displacement is 0 m

Δt → Sampling interval is 0.1 s

s_T → Displacement at time T

The displacement in discrete domain can be calculated using equation 12 where $s_0 = 0$. We use the cumulative integral function „cumtrapz" in Maltab in order to obtain the displacement curve.

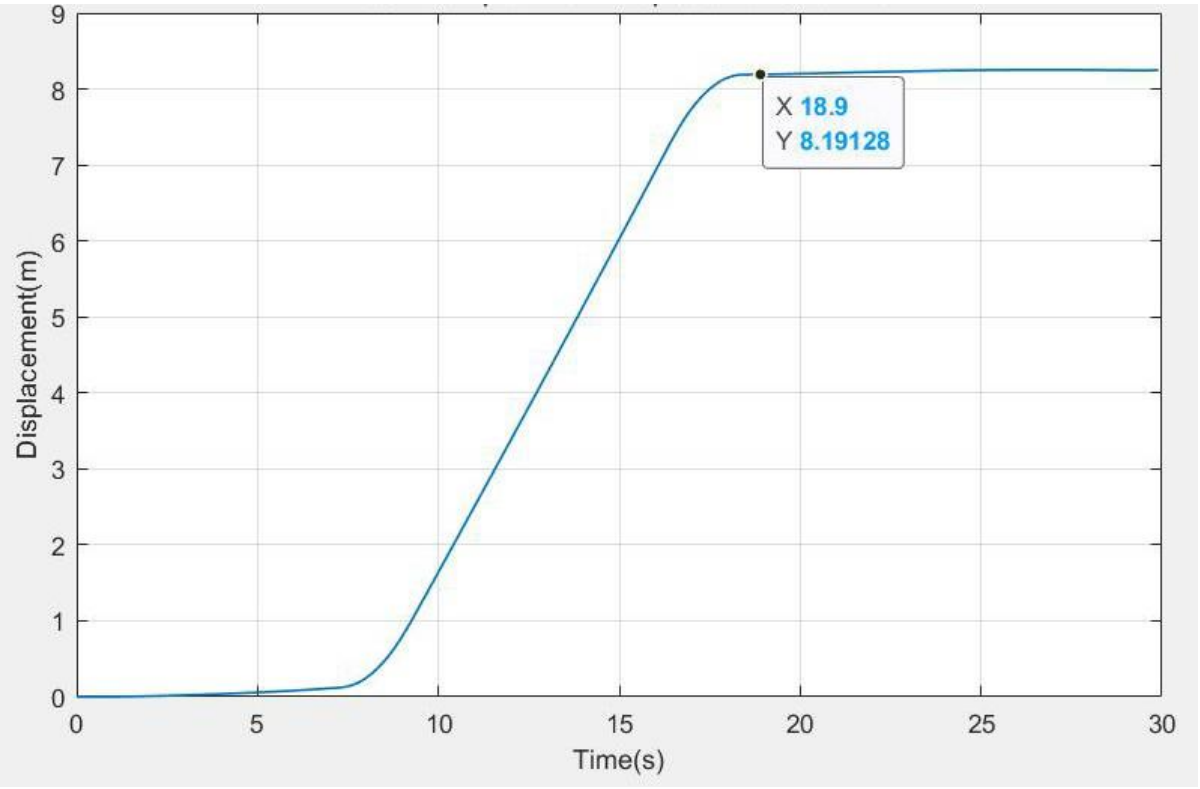

Figure 10 : Displacement curve obtained after performing cumulative integration on the velocity values

By observing the Y value in the Figure 10, it can be determined that the elevator has travelled 8.191 m, which is the distance between two floors.

Repeating the same process for the deceleration-acceleration event viz. the elevator going down. We get the following displacement curve.

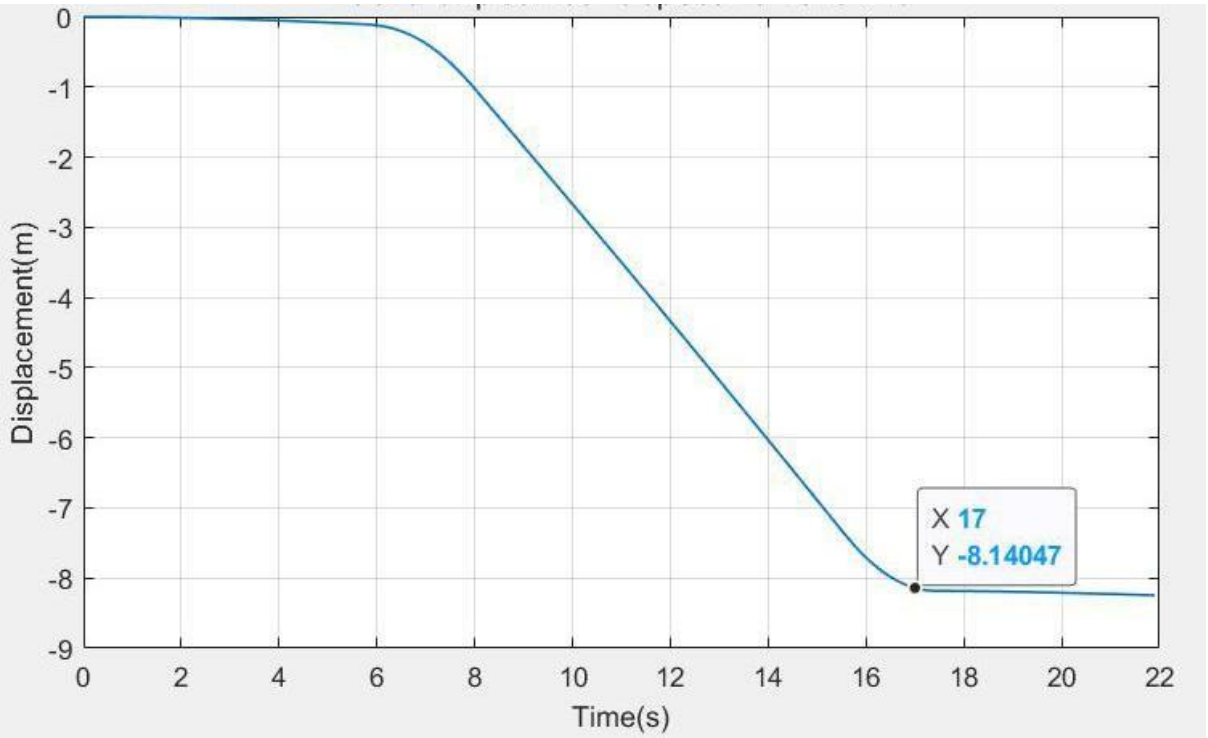

Figure 11 : Displacement curve obtained after performing cumulative integration twice on the discrete values of down trip deceleration-acceleration event.

The Figure 11 shows the displacement as a negative value. This is an indication that the elevator has moved downwards. A displacement of 8.14 m was observed.

In order to get a close estimate of the distance between two floors the average of all the up-down trips is taken and the standard deviation is calculated. The displacement values for each trip are noted in the Table 1 below.

Table 1 :Displacement values calculated for 6 acceleration-deceleration events

N	Up-Down	Displacement (m)
1	Up	8.11
2	Down	8.09
3	Up	8.19
4	Down	8.14
5	Up	8.11
6	Down	8.15
Average	=	8.13

Here N is the number of times displacement was measured. The average displacement for 6 values is 8.13 m.

Now we calculate the standard deviation using the equation :

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

The standard deviation σ is calculated to be 0.03 m.

6. Analysis and Conclusion

In this experiment, we design the circuit for interfacing the acceleration sensor using two configurations of Operational-Amplifier viz. subtracting amplifier and active low pass filter. We calculated the values for each component and a simulation was done in SPICE [7]. By running the simulation, the theoretical voltage output range was noted. Later on, we soldered the selected components on a perf-board and adjusted the output within the desired range by turning the potentiometer. Then we wrote the code for data acquisition and performed the necessary calculations for the timer 0 interrupt to generate a delay of 10 ms. We carried out the task of displaying the data obtained from the micro-SD card Data Logger module using MATLAB. We found that it is important to keep the sensor module stable during the measurement phase. The vibrations and shaking directly affect the acceleration values and can be observed as disturbances in the waveform. As a result, we calculated the average height of the 2 floors and the standard deviation.

7. References

[1] Atmel Corporation: Data Sheet ATmega32; 8-bit AVR Microcontroller with 32Kbyte In- System Programmable Flash; San Jose; Revision 2503Q-AVR-02/11; 2011.

[2] Analog Devices: Data Specification ADXL335; Small, Low Power, 3-Axis ± 3 g Accelerometer; Norwood; Revision B; 2010.

[3] Vishay: Data Specification BZX85-Series; Zener Diodes; Document Number 85607; Rev. 2.1; 2011.

[4] Physics Tutorial 2: Numerical Integration Methods, https://research.ncl.ac.uk/game/mastersdegree/gametechnologies/previousinformation/physics2numericalintegrationmethods/

[5] Agilent Technologies: Agilent InfiniiVision 2000 X-Series Oscilloscopes, Users Guide

[6] Experiment_2_Acceleration: Axel Bochert; Hochschule Bremerhaven; Winter Semester 2021/2022.

[7] Analog Devices : LTspice; "https://www.analog.com/en/search.html?q=LTspice "

[8] Texas Intruments: Data Specification TLC272; Precision Dual Operational Amplifier; August 1994.

8. Appendix

8.1. Filter Response Data Table

The measurements for frequency response are noted in a tabular format as follows :

Table 2 : Frequency response data points Frequency vs Amplitude noted manually.

Frequency (Hz)	Amplitude (V_{p-p})
0.1	6.95
1	6.95
3	6.95
6	6.42
8	5.75
10	4.7
20	1.77
30	0.828
50	0.317
60	0.232
80	0.137
90	0.103
100	0.083

8.2. Embedded C Code

8.2.1 Code : DataAcquisition.c

```
/*
 * DataAcquisition.c
 *
 * Created: Mayank & Kshitij 2022-2023 WiSe
 *
 * Program skeleton for the experiments in maritime
 * systems laboratory of embedded system design.
 * Prior to modify the program, change the name in
 * the "Created:" line.
```

```
 */

#define F_CPU 16000000

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "fifo.h"
#include "uart.h"
#include "dataio.h"
#include "timer0interupt.h"

volatile uint16_t intnum=0;
volatile uint16_t flag =0;
volatile uint8_t *bufcounter;
uint16_t sum = 0;
uint8_t count = 0;

void InitialiseHardware(void)
{
    sei();
// enables interrupts by setting the global interrupt
mask
    AdcInit();
// initializes the a/d converter
    bufcounter = uart_init(19200);
// initializes the UART for the given baudrate
    PortInit();
// initializes the port settings
    StartTimer0Interrupt() // timer 0 interrupt for
10 ms
}

int main(void)
/*
Example program for first experiments.
After initializing the interfaces and "Hello World"
is send to the serial port.
```

```
In a period of a second port pin D7 is toggled and
sample data are send to the
serial port. These sample data contain an index,
analog data input, digital port
inputs and an interrupt counter.
*/
{
    char Text[64];
    uint16_t ADCValue;
    //uint16_t index=0;

    InitialiseHardware();

    while(1)
    {
        TogglePortD(7);
        if (flag == 1)
        {
            ADCValue = ReadChannel(6);
            sum = sum + ADCValue;
            flag = 0;
            count++;

            if (count == 10)
            {
                count =0;
                sprintf(Text,"%d \r\n",sum);
                uart_puts(Text);
                sum = 0;

            }
        }
    }
}

ISR(TIMER0_COMP_vect)
/*
Interrupt service routine for timer 0 interrupt.
*/
{
    flag = 1;
}
```

8.2.2 Code : timer0interupt.c

```
/*
 * timer0interupt.c
 *
 * Created: Aug. 2020, Bochert
 *
 * Routines for the timer 0 for interrupt generation.
 * The interrupt service routine at TIMER0_COMP_vect
will be called up.
 */

#include <avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#include "timer0interupt.h"

void StartTimer0Interrupt(void)
/*
The timer 0 is initialized to generate an interrupt
every 15 ms.
*/
{
  OCR0 = 156;              // 10 ms period
  TCNT0 = 0;           // counter reset
  TCCR0 = 0B00001101; // start timer with prescaler of
1024 in clear timer on compare match mode (CTC)
  TIMSK |= (1<<OCIE0);    // timer 0 output compare
match interrupt enable
}

void StopTimer0Interrupt(void)
/*
The timer 0 is stopped and the interrupt is disabled.
*/
{
  TCCR0 = 0B00000000; // disable timer 0
 TIMSK &= ~(1<<OCIE0);     // disable timer 0 interrupt
```