PEDOMETER

Group: D9

Saurav Mohanty 05d07019, Ameya Joshi 05d07020, Bamdev Mishra 05d07037

Abstract:

The aim of the project was to design a *pedometer*, a device that can calculate the number of steps and the approximate distance covered by a person wearing it on taking steps. Design of an efficient and *simple* algorithm for distance algorithm was an important requirement. We have been able to implement the algorithm on MATLAB.

Design approach:

A pedometer or a *step-counter* is a portable device that counts each step a person takes by *detecting* the motion of a body part. In our case the *body part* of a person was kept a variable option. The aim was to come about a good (in terms of technical requirements) and user friendly position. On the other hand because the distance of each person's step varies, for the distance calculation some kind of *calibration* approach was necessary.

The following were the main sub-aims of the project.

1) Need of an *accelerometer* that should be able to output the variations of the accelerations in x, y and z axes with analog values.
2) The next step was to convert the analog output from the accelerometer to digital and then transfer the values to a computer *serially* and continuously.
3) Once the data had been successfully transferred, some tests were carried out to obtain the approximate graph or rather the variations of the digital values from the accelerometer using MATLAB. This also helped us to select suitable positions on the body from where the steps *looked like* more prominent.
4) Design of an algorithm that could compute the number of steps and approximate distance.

Hardware portion of the circuit included the following:

1) Accelerometer to output analog data of acceleration.
2) Accelerometer – Microcontroller interface via ADC.
3) Microcontroller (included an ADC) - personal computer (pc) serial data transfer using USART.
4) LCD-microcontroller interface.

Software included the following programs:

1) Continuous data transfer microcontroller code using RS-232.
2) MATLAB code for analyzing the digital data from the microcontroller.
3) Implementation of accelerometer – microcontroller, microcontroller – LCD interface and final algorithm for steps and distance measurement.

Design Approach:

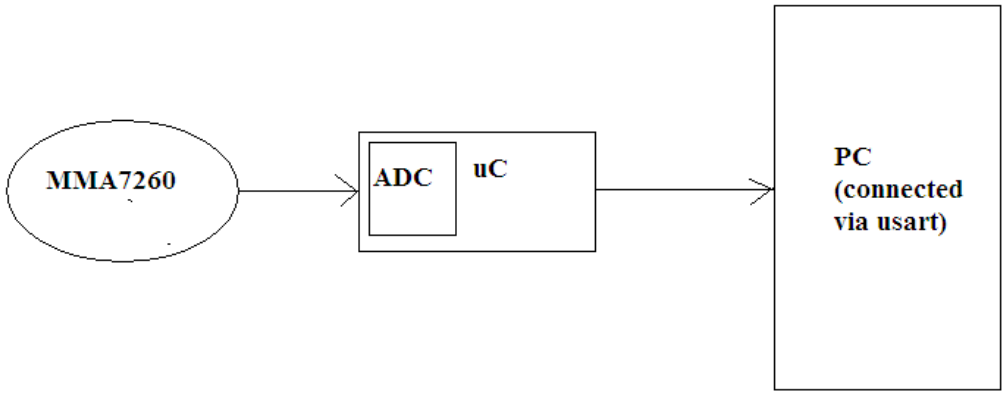
Our first aim was to choose the location of the accelerometer on the body. We ultimately decided on the waist. There were two reasons for this:
1. The horizontal acceleration of the waist would give us the best idea of the distance.
2. It gave us the option of including running measurements in the project.

The next circuit hardware chosen was the microcontroller. We decided to use ATMEGA16 as it had an internal ADC and satisfied the memory requirements. The accelerometer had already been chosen: MMA 7260 it had 3 – axes, allowing flexibility in the algorithm when needed.

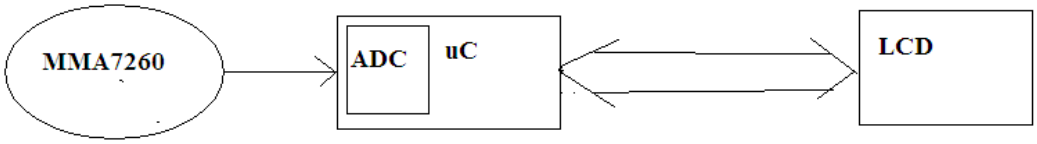
First we decided to make the board for the accelerometer-microcontroller interface. The microcontroller-PC interface was implemented separately on the bread board. After this had been completed, the data from the accelerometer was analyzed on the PC using MATLAB. Several cases were considered, and an algorithm to measure steps and distance was formulated. At the same time, the lcd-microcontroller interface was implemented on the bread board. The code was tested, and then transferred to the final PCB, which included the microcontroller-LCD interface as well.

Block Diagrams:

1. Block Diagram for testing and algorithm development



2. Block Diagram for algorithm development



Algorithm for counting the number of steps:

Counting of the number of steps : The algorithm used during the last time was implemented for the pedometer being worn on the feet and had to be implemented for the waist. Data was collected by the microprocessor from the accelerometer and transmitted serially to the computer via the RS232 port.
We divided the motion of walking into the following regions
a) The initial flat part during which the user has not started to walk yet and is in a resting position.
1) The first initial acceleration in the forward direction, indicating the start of the steps taken by the user, which is given by the first acceleration minima in our test cases(forward direction was the negative axis of our device).
2) The next maxima, which occurs immediately after the minima, followed by another minima after a long time indicating the beginning of the next step
3) The minima and maxima are repeated for each of the subsequent steps
4) The number of steps is given by the number of such acceleration minimums.

A typical simplified graph of the walking pattern is shown below

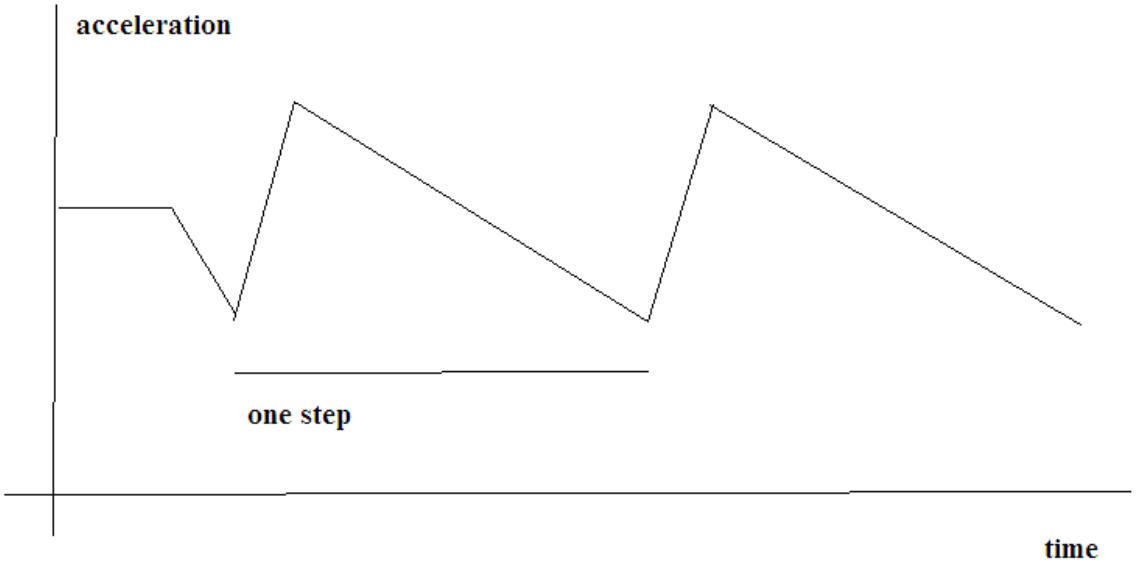


Fig. 1

In practice a sinusoidal wave of higher frequency is super imposed on the above curve, giving a graph which looked something like this

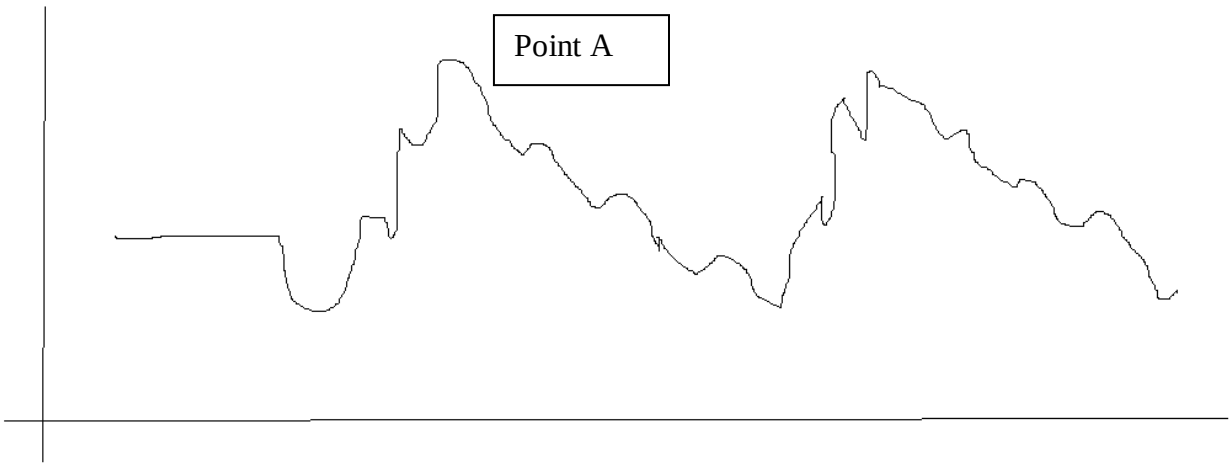


Fig.2

These oscillations were due to the vibrations that occurred in the person while walking which were related to movement in general but did not depend on the speed, frequency of steps and stride distance of the user

Furthermore on top of these high frequency oscillations of the accelerometer were super-imposed random fluctuations of the data about a local mean value. It looked something like this.

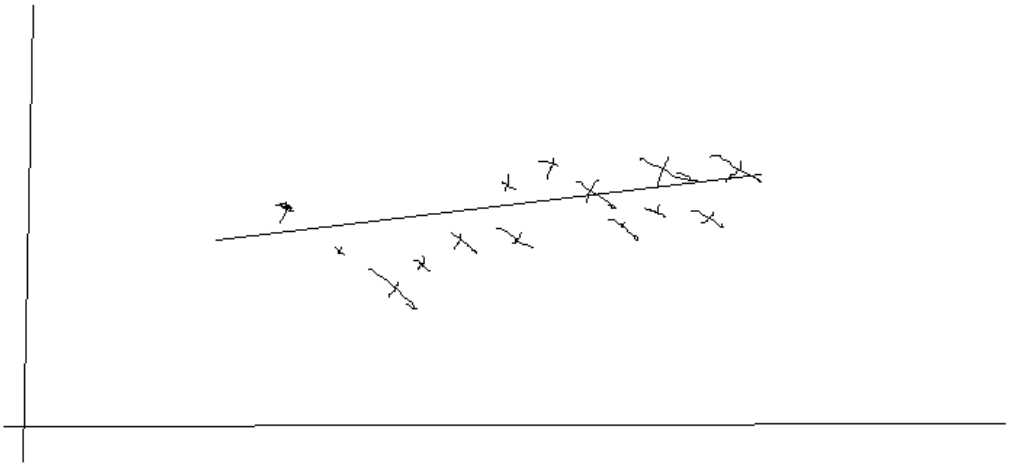


Fig. 3

To resolve this issue, we took a 40 point moving average of the data acquired from the accelerometer and under sampled the data to get the smooth figure of Fig 2. This removed the random errors of the ADC used to sample the data from the accelerometer.

Next, we found the local maxima of Fig 2, by comparing the present value from the accelerometer with the previous 2 values. However to remove the unwanted oscillations of fig 2 and to pin point the maximum point A in Fig.2 we plotted the envelop of the local maxima and found the maxima point of the envelop

This gave the point A's value. A similar procedure was repeated for the minima. Thus the main maxima and minima peak of a step were isolated as indicated by Fig.1

The number of steps was given by counting such minima and maxima

Algorithm for computing the distance

Method 1:
Initially we tried the method of double integration. The user is requested to start from rest so that the initial acceleration and speed is zero and from the subsequent acceleration data finding the distance would be possible by integration twice and using the 2 initial conditions.

The flat part of the output during rest, of the accelerometer subtracted from the actual value gave a value proportional to the instantaneous acceleration. This integrated twice was expected to give the distance covered.

However there were 2 basic problems to this method

1) The data from the accelerometer is discrete and thus does not contain all the information of acceleration. This means that some intermediate acceleration value would be missed, giving inaccuracies in the speed over a period of time.
2) These errors accumulated over time to give an offset to the speed resulting in speed calculated being non-zero even after the person came to rest

Method 2:

In this method, the basic assumption was that the difference in the maxima and the minima of the acceleration multiplied by the time of each step was proportional to the speed of the user. The distance travelled is then proportional to at^2 where a= (Amax – Amin) as calculated from Fig.1. This technique also makes sense dimensionally which is the same as the distance.

This was more robust technique. Although not 100% accurate, it gave pretty consistent results for different speeds, stride lengths and stride frequency. It gave approximately the same calculated distance for the same travelled distance. The drawback of this method was that it did not work well for ridiculously small and fast steps but worked accurately for most other forms of normal walking with significant consistency. One other problem was that the constant of proportionality varied slightly from person to person which required calibration. It was constant for a particular person independent of the nature of the step.
The advantage is that the errors in the distance after calibration did not accumulate over time and when the person stopped, the acceleration difference between maxima and minima was zero giving zero distance covered which is as expected.

Conclusions and future improvements

1. Dynamic calibration should be included
2. Size and frequency of steps should not affect the algorithm.
3. Power down modes should be included.