Accelerometer based wireless mouse

EE389: Electronic Design Lab Project Report

By

Group 09

Mohit Dandekar	06D07006
Neha Rambhia	06D07011
Prachi Pohekar	06D07014
Sai Priya Mahajan	06D07033

Department of Electrical Engineering
Indian Institute of Technology Bombay

November, 2009

Abstract

The aim of this project is to work with accelerometers and translate the motion of the hand into various applications in a virtual interface. It is most intuitive for us to use things based on our hand motions, as they form a very basic form of communication, signaling and gesturing.

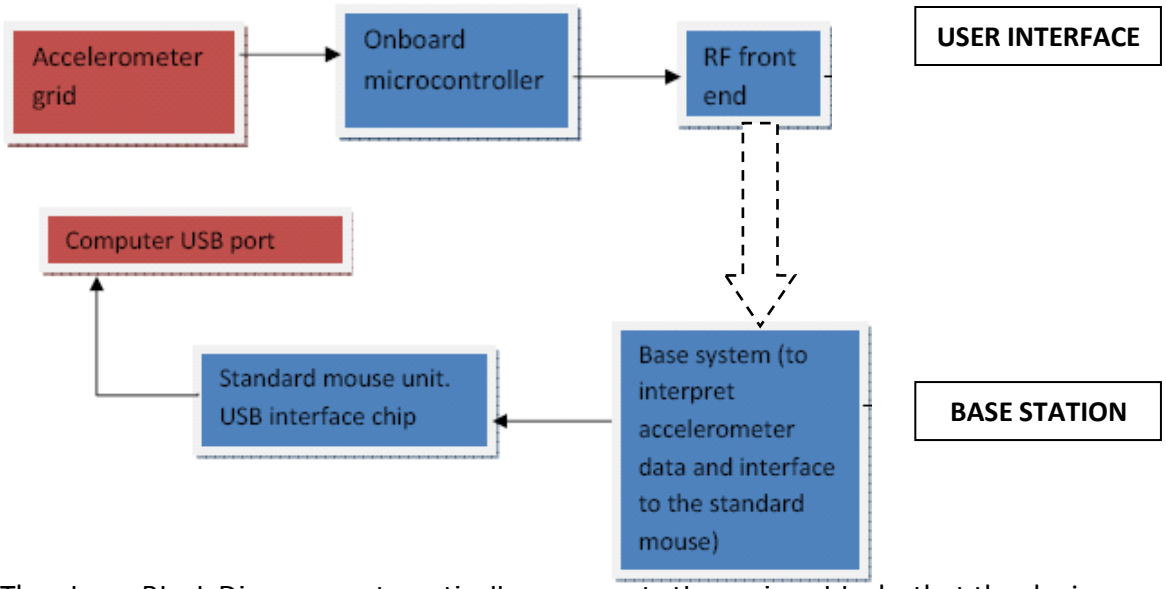
In order to translate these motions into the virtual world we use the accelerometer sensors. The orientation of accelerometers with the ground tells us the orientation our hand is in, because of the components of gravity.

The most commonly used hand motion control in a computer interface is that of a mouse. We have translated very normal gesturing of the hand into the motion of the pointer of the mouse. We have also traced the motion of the hand onto a virtual interface of matlab to show the 3-D variation in motion being captured virtually.

The mouse will be a hand mounted device that maps the movement of the user's hand onto the computer's mouse pointer, having all the standard functionalities as that of a computer mouse, left, right, middle click and scroll. Unique combination of finger movements that are very similar to the current mouse usage behaviorally would represent each of these.

Table of Contents

1.1 Block Diagram



The above Block Diagram systematically represents the various blocks that the device would have to contain.

1.2 Device Design

The device will be in two parts, the USB interface and the hand mounted sensor network. Of how we can implement the given final product that comes out of the systems that have been simulated and implemented as a part of the project.

Hand mounted sensor grid

Deviced such that there are 4 accelerometers MMA 7361L on the network. One at the central palm location and one each on the first, middle and ring finger at locations as shown in the image.

The finger placement is done such that the jerks of the fingers for specific clicks can be recognized with respect to each other as well as the central board.

There will be a single accelerometer on each of the white boards. The central board will have a RF transmitter whose antenna will be followed along the little finger and the processing unit.

This is the first prototype design we wish to implement. Further on, we could create pockets on the glove for each pcb such that the glove itself looks like a finished product without the boards of wires being visible to the user.

Finally we would like to incorporate the finger accelerometers on rings to be pulled onto the respective fingers and the central board on a clip that can be clipped on to the palm.

USB Interface Design

To avoid the hassels of long wires like most USB devices, this is going to be a small PCB that can be directly connected to the USB port (that can even look as compact as a pen drive!)

It will mount the mouse interface circuit, the processing unit and a ASK RF Receiving unit.

The eventual product could look just like a pendrive inserted into the USB port that interfaces with your mouse.

User Interfacing Signals

The algorithm and theory of the device highly depends on the user gestures for each motion. In order to make it easy for any user to use, the system finalized is such:

The motion of the mouse depends on the common motion of all four of the accelerometers and will be calculated by the central accelerometer readings.

The left click will be calculated from the movement of a jerk of the first finger (somewhat like clicking the left button on a mouse interface.) This will be done by the vector difference of readings of the finger and the central accelerometer.

The right click similarly will be difference of the ring finger from the central as well as other finger readings.

The middle click too will be middle finger difference from the other finger readings.

ICs and Parts used

MMA7361L – accelerometers used, 3 axis manufactured by freescale. Work on low voltage (3V) devices.

ASK transmitter and receiver- ideal for the short range applications that we need and economical.

Optical Mouse interface Protocol

ATMEGA 16 – used for processing on the transmitter and receiver end.

Computer interface.

Theoretical Models and Algorithms

Transmission channels

The accelerometer in place is a 3 axis accelerometer with analog output. The accelerometer output channels are multiplexed and sampled by the controller's inbuilt ADC successively at a rate of 1 KHz.

The accelerometer analog output channel has an inbuilt low pass output stage with effective cutoff at 500 Hz hence the sampling frequency of 1 KHz. The controller has 1 KB of memory available onboard also the RF link has a max data rate of 2400 baud, both these factors limit the amount of data that can be dealt with. Hence there is a need of down-sampling the sampled data to rate of 8 samples/sec/channel. To achieve this, the signal must be passed through a LPF with effective cutoff at 4 Hz.

2400 baud => 300 bytes / sec.

8 samples /sec/channel => 96 samples /sec

Every sample is spread to 3 bytes to be sent over the RF link.

Final byte rate = 96 x 3 = 288 bytes /sec < 300 bytes/sec

Note: the sampled data is 8 bit wide

The RF link is a transparent ASK transmitter receiver pair. The transmitter and receiver have been joined to the USART port on each side. Hence the transmission is under USART protocol at 2400 baud, 2 stop bits, no parity. The RF link being transparent merely acts as a RF front end.

The down-sampler:

To get 8 samples/sec from 1000 samples/sec down-sampling by a factor of 125 is required. The signal has bandwidth of 500 Hz and must be cutoff to avoid aliasing.

The filter chosen to do the down-sampling is a causal 125 point FIR filter. The method chosen to perform the convolution is overlap and save method.

The filter length is chosen to be 125 for a reason. When convolution is done is done by overlap and save method with data chunk of 125 samples, the circular convolution produces 125 samples out of which the first 124 samples are invalid. This is a trick we exploit. As we ultimately need to down-sample by a factor of 125 so while performing the circular convolution we don't calculate the N-1 samples (N=125) as they are anyway invalid and we are not going to use them. We calculate the Nth term that corresponds to every 125^{th} point of the filtered original sequence.

$$Y[n] = \Sigma\, x[k]\, h[(n-k)_N] \quad ; k = \{(0) - (N-1)\}$$

As we are interested in every 125^{th} sample => only Y[N-1] is of relevance.

$$Y[N-1] = \Sigma\, x[k]\, g[k] \quad ; k = \{(0) - (N-1)\}$$

Here g[k] is the sequence h[k] reversed.

Re-writing the equation to show downsampling

$$Y[n] = \Sigma\, x[n*N+k]\, g[k] \quad ; k = \{(0) - (N-1)\}$$

This operation has to be performed on each channel: the Pseudo code of a 'MAC' operator will look like:

```
void MAC(g[],x[])
    {
     float sum;
      for(k=0;k<N;k++)
       {
      sum +=  x[k] * g[k] ;
      }
        return (unsigned char)sum;
    }
```

On an AVR microcontroller running at 8 MHz such an operation (if performed independently)can take 10 cycles /iteration => 1250 cycles

For 12 channels the number is 15000 cycles => 1.875 ms to generate 1 sample for each channel. Hence be the time we have done this computation 2 more samples have come in.

To do it in this manner we will need a storage space of 125 * 12 = 1500 bytes + 125 point FIR filter that is float type = 500 bytes.

The controller does not have enough space to adopt this approach. Hence we adopt a different approach.

The above is a maximum sampling rate. We are transmitting in the demo 4 channels at the rate of 10 samples per second, with the first 3 channels with x,y,z accelerometers and the last channel with a compiled information of the clicks.

Rather than first storing and then performing the MAC operation, the operation can be performed as the samples are brought in by the ADC.

This will considerably save the storage space as now I only have to maintain 12 intermediate sums and declare them valid every 125^{th} processing instant.

The samples have been obtained after filtering, are pushed into the transmission buffer. The transmission as mentioned earlier is handled by the USART port. The USART is running on an interrupt routine, interrupt generated by the even of the transmission buffer being empty. If there is any data in the transmit buffer it is pushed out by the USART port to the RF front end.

Serial transmission Scheme:

The data byte obtained from the down-sampling process is spread into 3 bytes according to following packet format.

Byte structure:

Data_valid	data/ device	packet	number	DATA/ INFO NIBBLE

Byte 1: if data valid => the transmit buffer is not empty

1	0	0	1	accelerometer index	axis index

Byte 2: if data valid

1	1	1	0	x	x	x	x

x are the first 4 bits of data

Byte 3:

1	1	1	1	x	x	x	x

x are the last 4 bits of data

if transmit buffer empty the transmitted byte is:

0	x	x	x	x	x	x	x

data not valid => the receiver ignores

Receiver Scheme

Receive complete interrupt

Invoked every time a byte is received over the RF line. Every time a byte is received, it is decodes, depending on the start byte that encompasses the data.

The receiver is in default state zero waiting for say channel 1 to get received. If the header of the channel is the same as that of the state 1, and the data is valid then the data is accepted, and then the code waits for channel 2 to come in.

In case there is faulty data in channel 1, then the code still keeps waiting for a correct data for the channel 1 to come up. In case there is a fault on the channel 2 or channel 3 signals then you still increment the channel number you are waiting for and you would get correct data coming in from the next cycle of channels.

There is a receiver buffer that is circular in nature, used for each channel separately in which they bytes are pushed into. This is a front end primary buffer.

Main Function

After initializing,

Mainfun in infinite loop, waits for global variable called permit which is set after channel 4 is received successfully in the receive complete.

If this is set, then the filter channel function is called which is used for the high pass filtering of the x,y,z channels.

The median filtering is done of the 9 values that are stored in the buffer, this is used for the robustness of the data, where in the outlying data points can be filtered out.

The next function called is the IR filter which is a high pass filter cutting off DC and .1Pie digital frequency. It uses a data structure called the channel buffer.

It is a single input single output IR function.

The accelerometer measures both static and dynamic acceleration. We need to deal with only the dynamic accelerations to measure the movement in the constrained space that we have. Dynamic accelerations will have a much higher frequency and the static components appear as a Dc distributed amongst the 3 channels of the accelerometer.

To filter out this static reading we put the reading through low pass filtering.

The filter was designed through the digital filter requirement, converting it to a first order butterworth filter, whose output was converted to digital again, whose output was a z transform, and the difference equation was directly implemented.

The output of the channel is integrated to get the velocity in x and y which is mapped directly into the rate of flipping of the channels of the mouse which with a scaling factor is mapped into the motion of the pointer of the computer we interface with.

Timer Interrupt

Data structures

Large 64 bit variables

Global Time

Whenever our timer interrupt is invoked, it is incremented with the difference in value between the last time and current time that it was called on. Each integer increase is equivalent to 32 microseconds increase in time.

There are 2 flip channels with independent flipping rate.

Every time the timer interrupt comes in, it calculates when in the global time the particular channel needs to be flipped and is stored and at that global time the channel is flipped.

Every time the channel is flipped, the next flip time is calculated and stored.

The timer interrupt is invoked at instances that are multiples of a maximum set resolution till it would exceed and with minimum resolution reaches next flip time for any of the channels.

Each time the flip occurs a 2 bit grey code counter increments or decrements depending on the direction of the variable.

The interface between the protocol for the optical mouse and our system is coded such that our code generates codes like an optical mouse signals would.

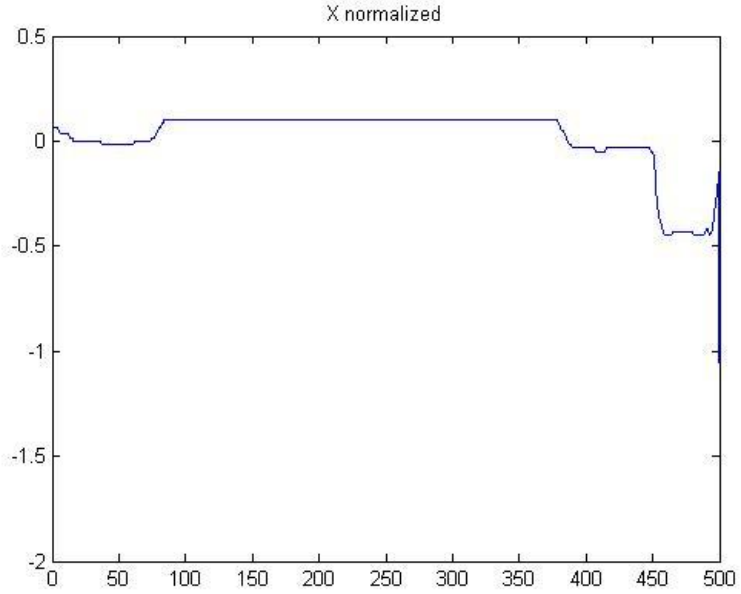
On research, the codes are such that there are 4 channels, 2 of 2 pairs. Each pair represents a direction. This bit pair flips in grey code. And direction of flip determines the direction of movement of the mouse pointer and the rate of flip determines the speed of the motion of the pointer.

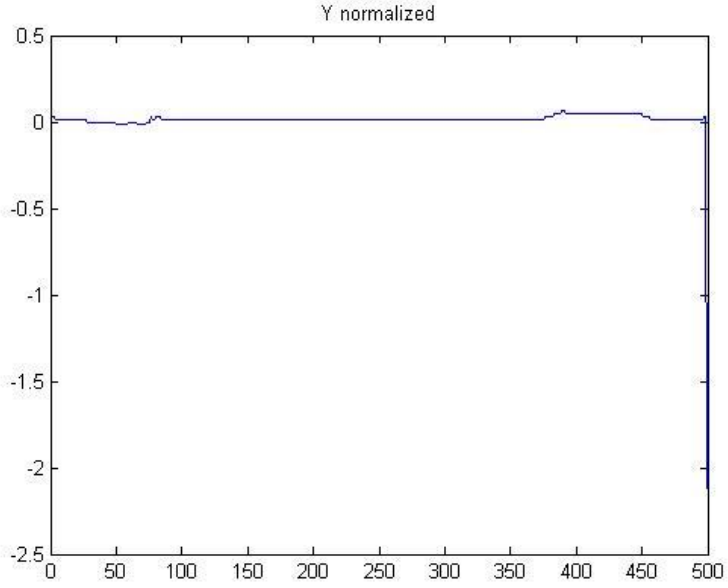
This is simulated by the timer interrupt control described above and the mouse interface connects with the USB interface plugged into the computer system.

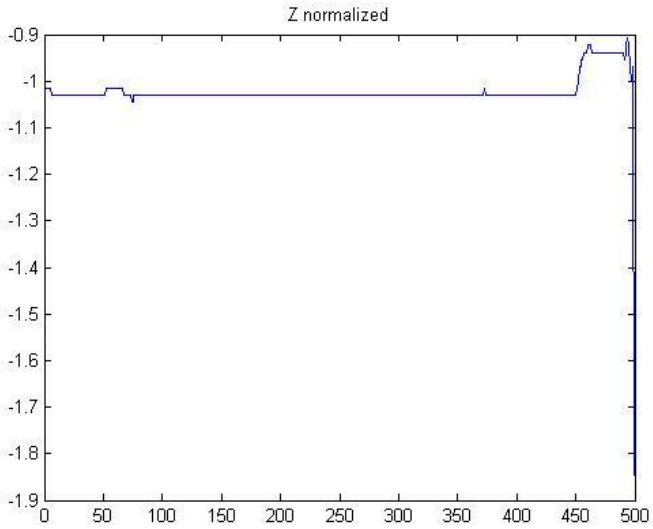
Implementation Results

The PCB made for the various boards look like follows.

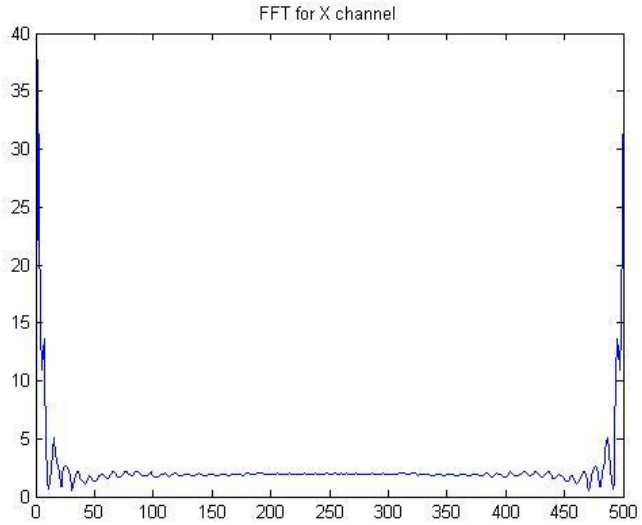
When the accelerometer has been held steady, the following graphs are obtained

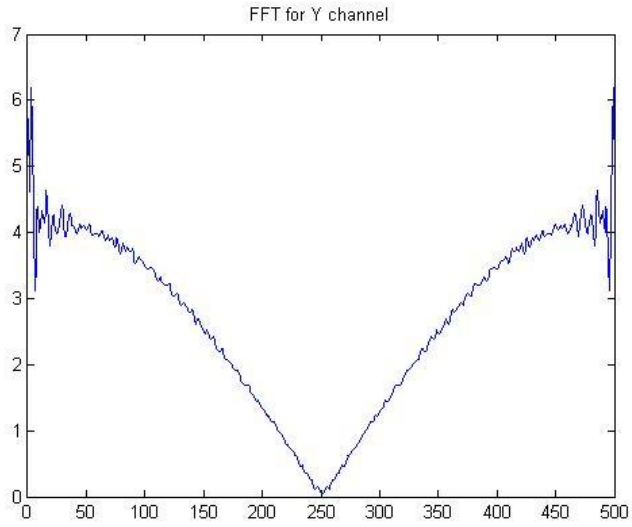


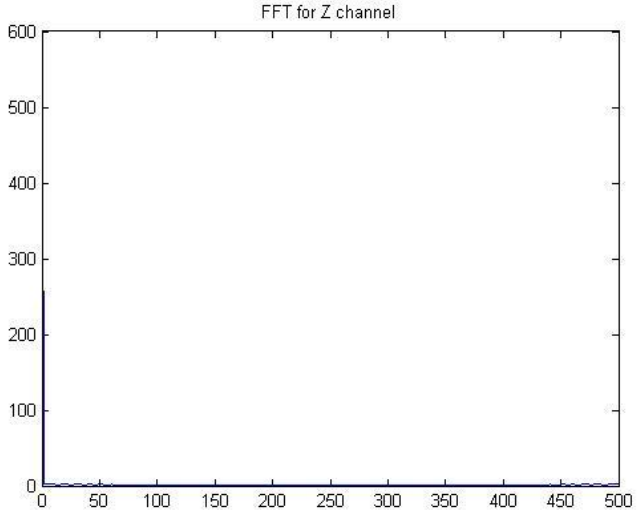


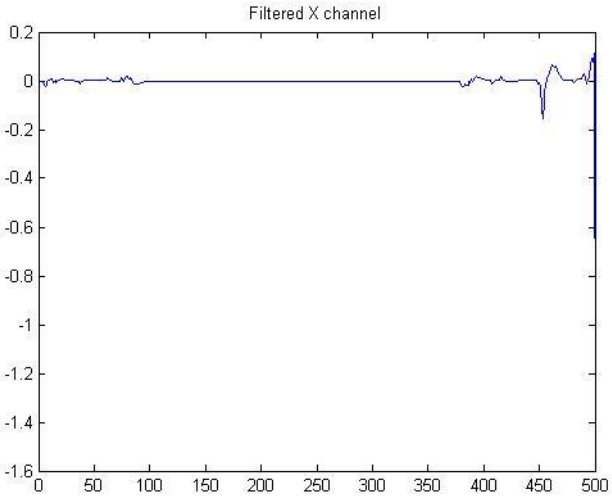


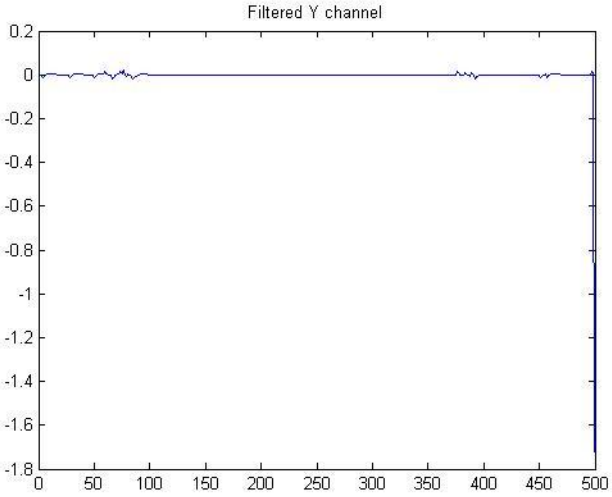
The value of Z=-1 corresponds to –g acceleration.

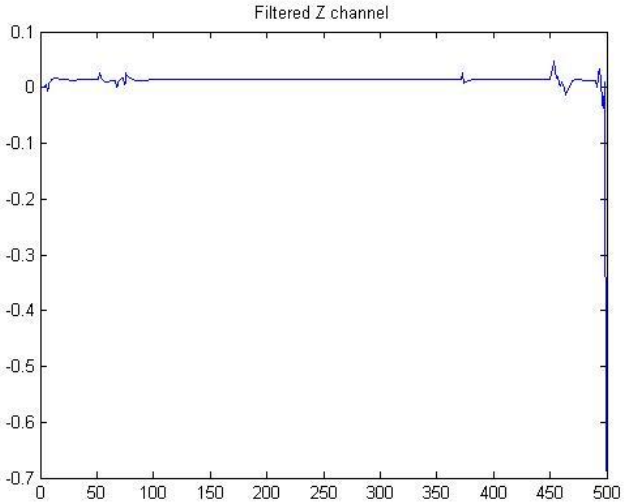



FFT for Y channel
0
1
2
3
4
5
6
7
0
50
100
150
200
250
300
350
400
450
500


FFT for Z channel
0
100
200
300
400
500
600
0
50
100
150
200
250
300
350
400
450
500

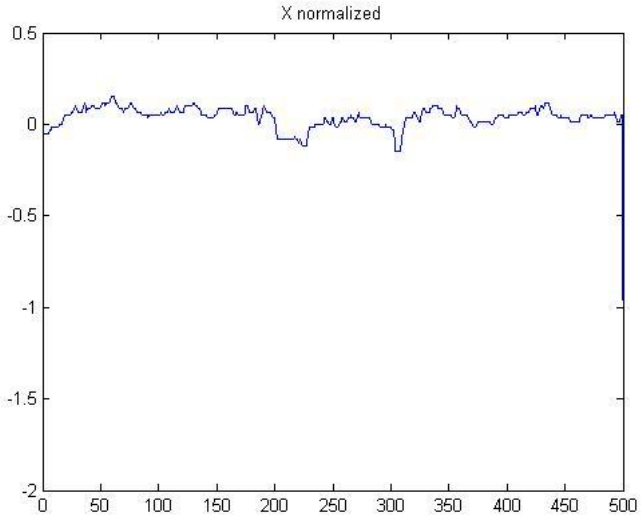

Filtered X channel
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
0
50
100
150
200
250
300
350
400
450
500

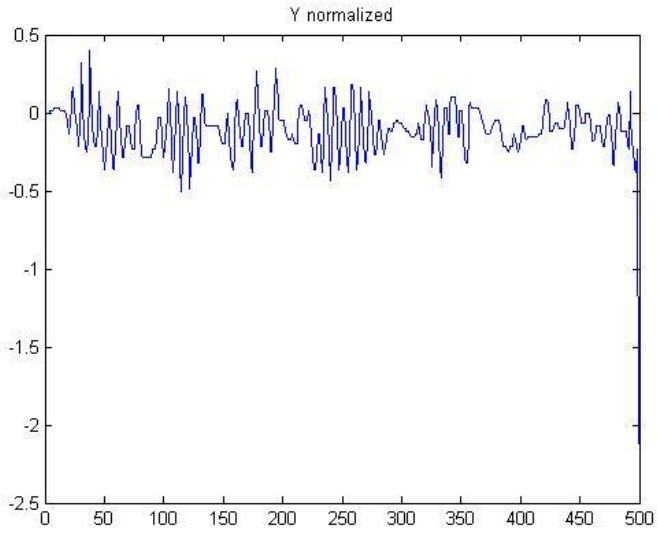

Filtered Y channel
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
0
50
100
150
200
250
300
350
400
450
500

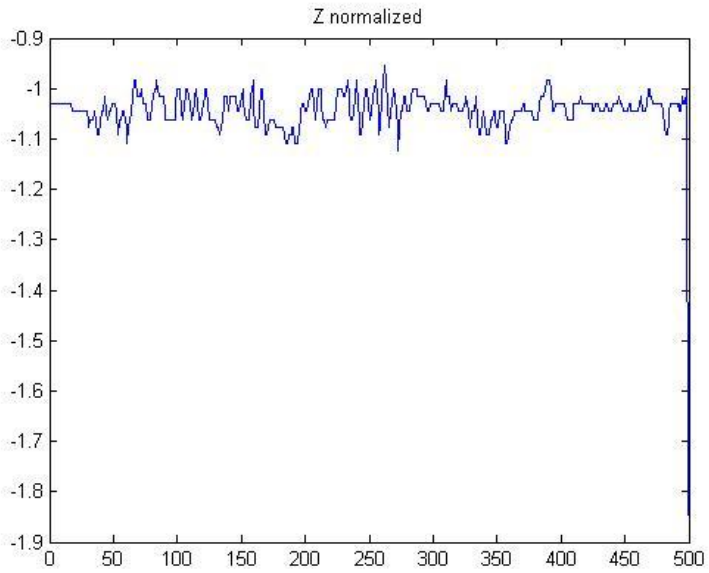

Filtered Z channel
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
0
50
100
150
200
250
300
350
400
450
500

The following was when the mouse pointer was moved around to show the simulation results of x,y,z accelerometers.

X, Y normalized show variation about 0 and Z about -1 which shows that the constant acceleration of X,Y is 0 and Z is -g

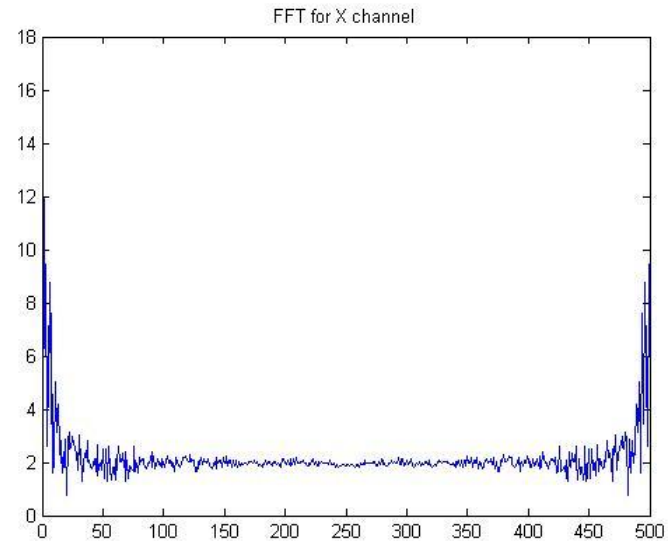


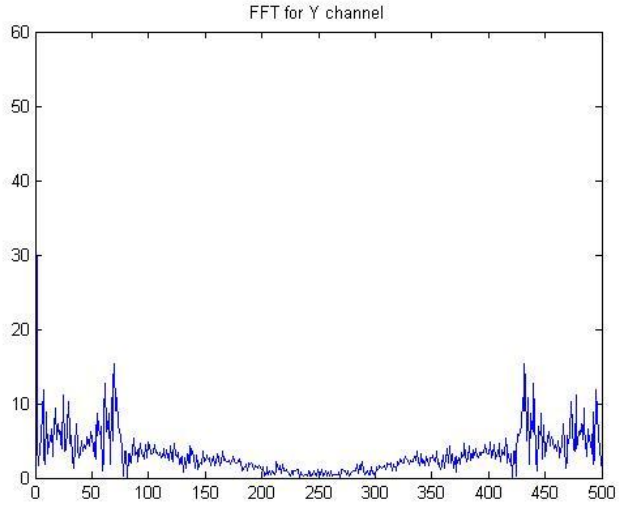


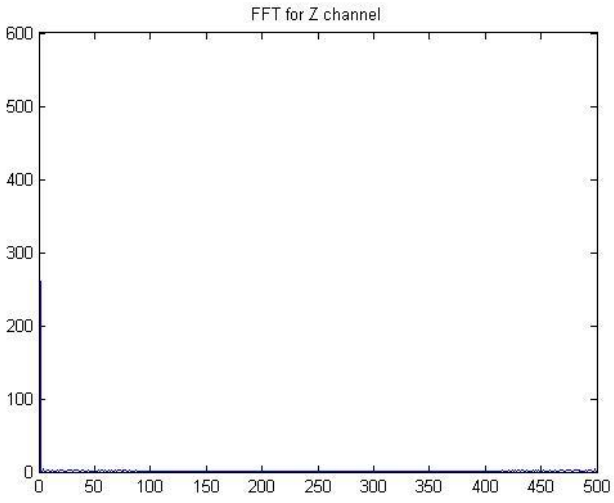


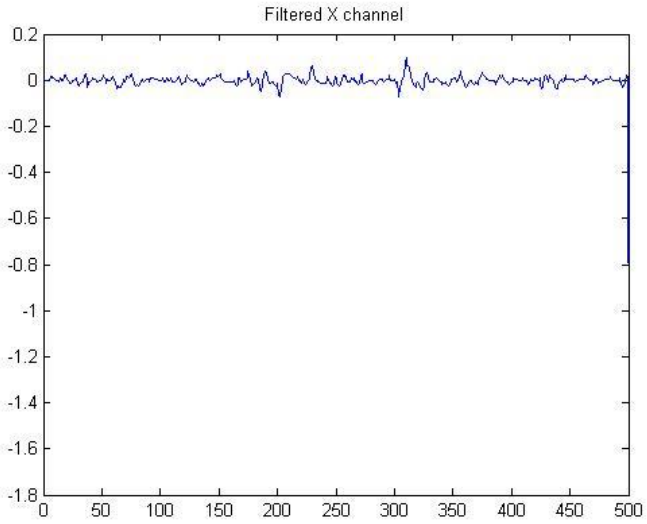
The fourier transformed X,Y,Z are as shown. There are peaks at higher as well as lower frequencies. The peaks at the lower frequencies correspond to the DC part which we filter out.

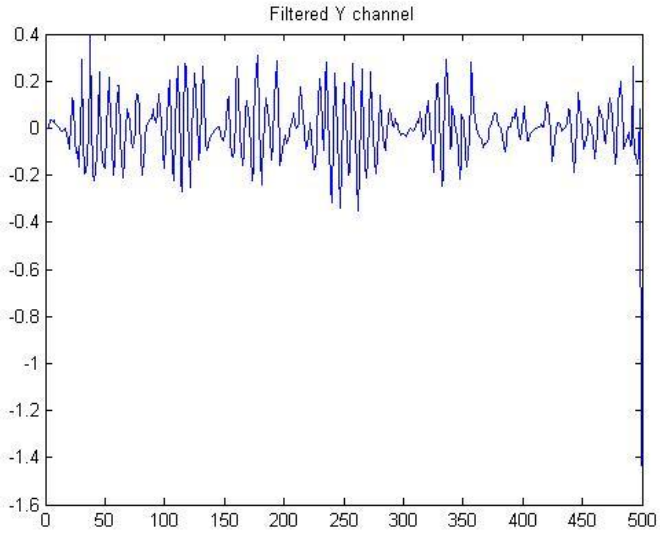
We can see in the filtered X,Y,Z graphs that all 3 show motion, variation about 0 hence the DC part has been Filtered out.

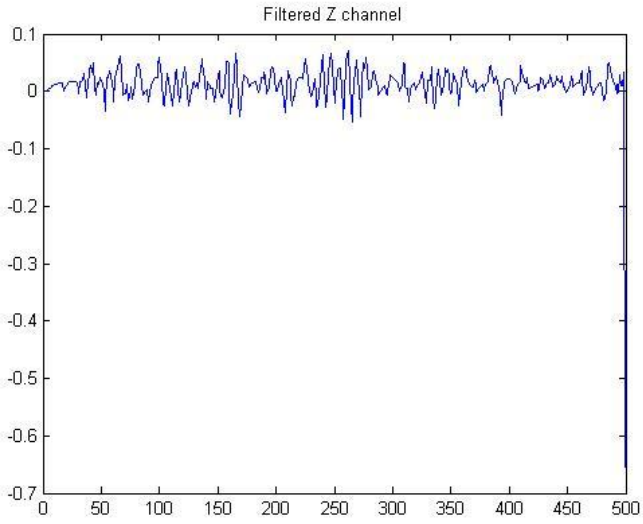

FFT for X channel
18
16
14
12
10
8
6
4
2
0
0
50
100
150
200
250
300
350
400
450
500


FFT for Y channel
60
50
40
30
20
10
0
0
50
100
150
200
250
300
350
400
450
500

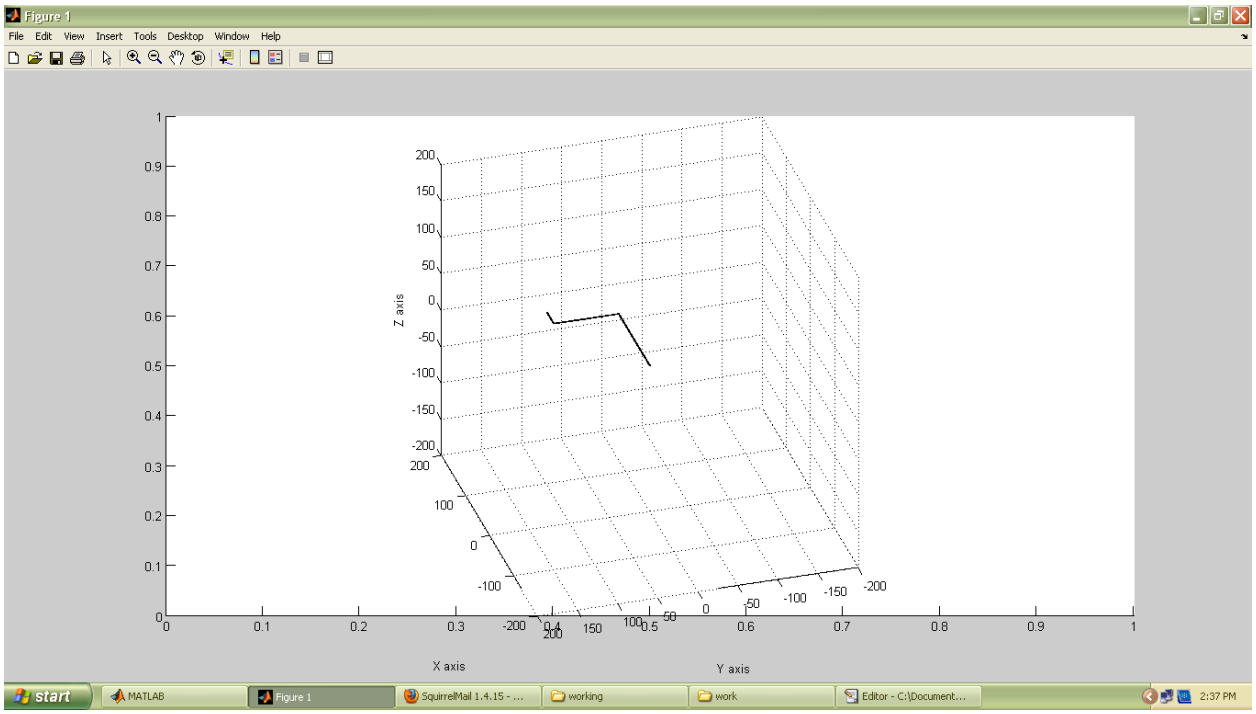

FFT for Z channel
600
500
400
300
200
100
0
0
50
100
150
200
250
300
350
400
450
500


Filtered X channel
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
0
50
100
150
200
250
300
350
400
450
500


Filtered Y channel
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
0
50
100
150
200
250
300
350
400
450
500


Filtered Z channel
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
0
50
100
150
200
250
300
350
400
450
500

Simulation results of moving and tilting the sensor board to show how it varies in 3-D space produce the following results.



Limitations

The High Pass filter implemented and simulated using MATLAB, although identically implemented in the ATMEGA did not function the same way. Hence though the design of the filter worked, the filter on the ATMEGA did not work and there was a major influence of the DC component which couldn't be removed.

This inhibited the implementation of a perfectly moving pointer.

Because of the data transfer delays, there was an inherent delay between motion of the sensor and the mouse, which could be dealt with better transmission channels.

The clicks and the scroll was not tested. Theoretically they were coded. The buttons and scroll button wasn't implemented.

Multiple accelerometers were not interfaced.

Conclusion

The road map ahead for this kind of implementation has a spectrum of improvement channels and applications that can be worked. The effective working and simulation of the same would help in the application in the virtual world interfacing of human gestures as more and more things are going intuitive making user interfaces simple to use and complicated to code!