1. **Reduksi Error Pada Pengukuran Dan Perhitungan**

Jika hanya menggunakan metode pengukuran dan perhitungan diatas, dapat diamati bahwa tingkat kesalahan dari pengukuran dan perhitungan cukup tinggi jika dibandingkan dengan nilai sesungguhnya.

Untuk meminimalisir tingkat kesalahan dilakukan perbaikan proses pengolahan sinyal lebih lanjut. Dimaulai dari sinyal yang didapat dari sensor akselerometer.

Sinyal dari akselerometer terdapat banyak derau untuk itu diperlukan filter. Terdapat banyak macam filter digital yang dapat diaplikasikan untuk sinyal dari akselerometer ini. Pemilihan filter didasarkan pada sinyal yang didapat.

Data percepatan dari akselerometer merupakan sinyal yang informasinya direpresentasikan dalam runtun waktu diskret. Sehingga dipilih filter IIR (*Infinite Impulse Response*) dan *prototype* filter analog yang dipilih adalah LPF *Bessel Filter*.

Langkah selanjutnya adalah menentukan parameter – parameter fiter yaitu frekuensi sampling, frekuensi *cutt off*, dan orde filter.

Pengambilan data akselerometer dalam 1 detik sebanyak 125 data sehingga frekuensi sampling adalah 125 Hz, sedangkan frekuensi *cutt off* dibuat 5 Hz dan orde filter 10.

Respon frekuensi dari filter digital adalah pada [0,1] sedangkan 1 merepresentasikan π yang merupakan frekuensi *Nyquist* sehingga π = 62.5 Hz. Frekuensi *cutt off* dari filter digital adalah ωc = *fc / fs* = 0.04 rad/s.

Untuk mencari koefisien rekursi dari fungsi alih filter yang akan dibuat pada pengujian ini dilakukan dengan menggunakan program *Filter* *helper* yang tersedia pada website <http://www-users.cs.york.ac.uk/~fisher/mkfilter> dengan memasukan parameter frekuensi sampling, frekuensi *cut off* (*corner frequency*) dan orde filter.

Hasil yang didapat adalah sebagai berikut.

**Filter Design Results**

Generated by:   <http://www-users.cs.york.ac.uk/~fisher/mkfilter>

**Summary**

You specified the following parameters:

|  |  |  |
| --- | --- | --- |
| filtertype | = | Bessel |
| passtype | = | Lowpass |
| ripple | = |  |
| order | = | 10 |
| samplerate | = | 125 |
| corner1 | = | 5 |
| corner2 | = |  |
| adzero | = |  |
| logmin | = |  |

**Results**

Command line: /www/usr/fisher/helpers/mkfilter -Be -Lp -o 10 -a 4.0000000000e-02 0.0000000000e+00

raw alpha1 = 0.0400000000

raw alpha2 = 0.0400000000

warped alpha1 = 0.0402118901

warped alpha2 = 0.0402118901

gain at dc : mag = 3.457215320e+06 phase = 0.0000000000 pi

gain at centre: mag = 2.444620397e+06 phase = 0.8569553843 pi

gain at hf : mag = 0.000000000e+00

S-plane zeros:

S-plane poles:

-0.4870299950 + j 0.0610482858

-0.4870299950 + j -0.0610482858

-0.4654470131 + j 0.1837480006

-0.4654470131 + j -0.1837480006

-0.4198709098 + j 0.3085216633

-0.4198709098 + j -0.3085216633

-0.3437908196 + j 0.4379854060

-0.3437908196 + j -0.4379854060

-0.2187410626 + j 0.5792466866

-0.2187410626 + j -0.5792466866

Z-plane zeros:

-1.0000000000 + j 0.0000000000 10 times

Z-plane poles:

0.6073755865 + j 0.0394557060

0.6073755865 + j -0.0394557060

0.6134616787 + j 0.1202501436

0.6134616787 + j -0.1202501436

0.6265413107 + j 0.2073760334

0.6265413107 + j -0.2073760334

0.6490511046 + j 0.3081590352

0.6490511046 + j -0.3081590352

0.6877886497 + j 0.4406309504

0.6877886497 + j -0.4406309504

Recurrence relation:

y[n] = ( 1 \* x[n-10])

+ ( 10 \* x[n- 9])

+ ( 45 \* x[n- 8])

+ (120 \* x[n- 7])

+ (210 \* x[n- 6])

+ (252 \* x[n- 5])

+ (210 \* x[n- 4])

+ (120 \* x[n- 3])

+ ( 45 \* x[n- 2])

+ ( 10 \* x[n- 1])

+ ( 1 \* x[n- 0])

+ ( -0.0217192171 \* y[n-10])

+ ( 0.3012846437 \* y[n- 9])

+ ( -1.8998620952 \* y[n- 8])

+ ( 7.1765621146 \* y[n- 7])

+ (-17.9969646061 \* y[n- 6])

+ ( 31.3334899657 \* y[n- 5])

+ (-38.3927239217 \* y[n- 4])

+ ( 32.7257173947 \* y[n- 3])

+ (-18.5945171308 \* y[n- 2])

+ ( 6.3684366602 \* y[n- 1])

Pada bagian *Recurent relation* menunjukan persamaan relasi dari fungsi alih filter yang dihasilkan dengan x[n] adalah sebagai dan y[n] sebagai output. Langkah selanjutnya adalah menerapkanya kedalam program arduino.

Berikut fungsi filter IIR pada arduino yang diigunakan.

// LPF Bessel, fc = 5 Hz Fs = 125 Hz Orde=10

#define GAIN 3.457215320e+06

**float** IIR(**float** value, **float** xv[], **float** yv[]) {

xv[0] = xv[1];

xv[1] = xv[2];

xv[2] = xv[3];

xv[3] = xv[4];

xv[4] = xv[5];

xv[5] = xv[6];

xv[6] = xv[7];

xv[7] = xv[8];

xv[8] = xv[9];

xv[9] = xv[10];

xv[10] = value / GAIN;

yv[0] = yv[1];

yv[1] = yv[2];

yv[2] = yv[3];

yv[3] = yv[4];

yv[4] = yv[5];

yv[5] = yv[6];

yv[6] = yv[7];

yv[7] = yv[8];

yv[8] = yv[9];

yv[9] = yv[10];

yv[10] = (xv[0]+xv[10]) + 10\*(xv[1]+xv[9]) + 45\*(xv[2]+xv[8]) + 120\*(xv[3]+xv[7]) + 210\*(xv[4]+xv[6]) + 252\*xv[5] + (-0.0217192171\*yv[0]) + (0.3012846437\*yv[1]) + (-1.8998620952\*yv[2]) + (7.1765621146\*yv[3]) + (-17.9969646061\*yv[4]) + (31.3334899657\*yv[5]) + (-38.3927239217\*yv[6]) + (32.7257173947\*yv[7]) + (-18.5945171308\*yv[8]) + (6.3684366602\*yv[9]);

**return** yv[10];

}

Untuk memanggil fungsi IIR diatas dilakukan dengan cara sebagai berikut.

//============ Filtering data akselerometer==========//

**float** accelXIn[11], accelXIn[11], accelXIn[11];

**float** accelXOut[11], accelXOut[11], accelXOut[11];

**float** accelX = IIR(Ax, accelXIn, accelXOut);

**float** accelY = IIR(Ay, accelYIn, accelYOut);

**float** accelZ = IIR(Az, accelZIn, accelZOut);