SISTEM DETEKSI MUSIK DENGAN METODE *BEAT-DETECTION* PADA ARM ROBOT 4-DOF BERBASIS TMS320VC5402

Achmad Fausi[1], Bima Sena Bayu D, S.ST, M.T[2], Fernando Ardilla, S.ST, M.T [3]
[1] Mahasiswa Program Studi Teknik Komputer, [2,3] Dosen Program Studi Teknik Komputer
Politeknik Elektronika Negeri Surabaya – Institut Teknologi Sepuluh Nopember (ITS) Surabaya
Kampus ITS, Sukolilo, Surabaya 60111

[1] owgee@student.eepis-its.edu
[2] bima@eepis-its.edu
[3] nando@eepis-its.edu

ABSTRAK

Pada penelitian ini akan dibuat sistem pendeteksi musik menggunakan metode *beat detection* pada *Arm Robot 4-DOF* berbasis *TMS320VC5402*. Sistem ini mengembangkan teknologi pengenalan bunyi (*Sound Recognition*). Sound recognition adalah sebuah teknologi terapan yang mengubah sinyal-sinyal elektrik yang berasal dari bunyi menjadi instruksi atau aksi tertentu. Sinyal input bunyi akan diolah pada *TMS320VC5402* menggunakan *filter digital*. Terdapat 2 *filter digital* yang semuanya diuji pada penelitian ini. *Filter IIR* terbukti lebih baik daripada *filter FIR* karena membutuhkan koefisien yang lebih sedikit untuk respon frekuensi yang curam. Banyak sedikitnya koefisien yang bertipe float sangat berpengaruh pada cepat lambat pengolahan suara di *TMS320VC5402*. *Filter IIR lowpass* dipilih oleh peneliti setelah diuji, *filter IIR bandpass* tidak bisa bekerja dengan frekuensi cutoff dibawah 1 KHz. Sedangkan *filter FIR* (*lowpass, bandpass dan highpass*) bisa bekerja dengan respon kecuraman yang baik pada orde di atas 64. Masalah utama yang sering dihadapi pengolahan sinyal suara adalah noise sekitar dan jarak microphone terhadap sumber suara. Pengujian noise pada penelitian ini dilakukan dengan meminta beberapa orang pada radius 1 meter dan diatasnya untuk bersuara dengan volume wajar. Tingkat keberhasilan proyek akhir ini sekitar 60-90%.

Kata kunci : **Beat Detection, Filter Digital, Filter IIR, Filter FIR, TMS320VC5402, Arm Robot 4-DOF**

1. PENDAHULUAN

Beberapa tahun terakhir, perkembangan dunia teknologi semakain pesat. Banyak ditemukan dan dikembangkan inovasi-inovasi terbaru di berbagai bidang. Termasuk di bidang pengenalan suara (*voice recognition*). Dimana sinyal informasi yang dikirimkan tidak hanya berupa data teks tetapi juga suara.

Hal lain yang tidak kalah menariknya adalah penggunaan teknologi dalam bidang suara ini untuk mengakses atau memberikan perintah. Sinyal suara yang masuk akan diproses dan dikenali oleh suatu mesin untuk kemudian digunakan sebagai perintah.

Robot 'Putu Ayu' berhasil meraih Juara I Nasional pada Kontes Robot Seni Indonesia (KRSI) di Universitas Muhammadiyah Malang tahun 2010 lalu. Inilah salah satu aplikasi dari teknologi voice recognition. Saat musik terdengar, robot akan bergerak melenggak-lenggok. Saat musik berhenti, maka robotpun akan berhenti bergerak.

Mengenai robot 'Putu Ayu', ada satu kekurangan pada robot ini. Yaitu robot hanya mendeteksi ada tidaknya suara sebagai parameter robot akan bergerak atau diam. Bukan bergerak sesuai irama musik sebagaimana seorang penari. Atau mengenali salah satu bagian dari musik itu sendiri misalnya beat, suara seruling atau suara gendang.

Penelitian ini mencoba memberikan metode alternatif untuk menyelesaikan masalah pada kasus KRSI yaitu pengenalan suara musik dalam hal ini beat atau irama. Sehingga nantinya robot bisa bergerak melenggak-lenggok harmonis sesuai irama musik yang dimainkan.

2. KONSEP FILTER IIR

Filter adalah adalah sebuah rangkaian yang dirancang agar melewatkan suatu pita frekuensi. Pengertian lain dari filter adalah rangkaian pemilih frekuensi agar dapat melewatkan frekuensi yang diinginkan dan menahan (couple) atau membuang (by pass) frekuensi lainnya.

Filter IIR adalah salah satu tipe dari filter digital yang dipakai pada aplikasi Digital Signal Processing (DSP). IIR kepanjangan dari Infinite Impulse Response. Mengapa disebut respons impulsnya tak terbatas (infinite)? Karena *adanya feedback* didalam filter, jika anda memasukkan sebuah impulse (yaitu sebuah sinyal '1' diikuti dengan banyak sinyal '0'), maka pada outputnya akan terus menerus berosilasi karena adanya umpan balik, walaupun pada

prakteknya akan hilang pada suatu saat. Fungsi transfer filter IIR adalah

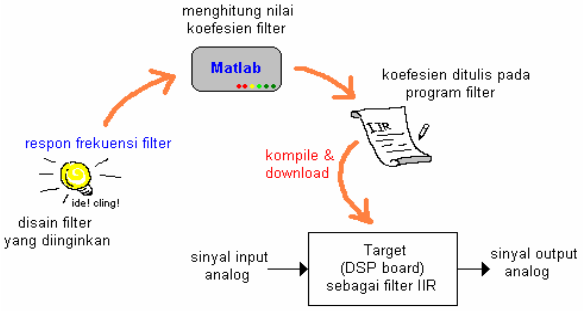
$$H[z] = \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-m}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}}$$

Keuntungan filter IIR antara lain adalah membutuhkan koefesien yang lebih sedikit untuk respon frekuensi yang curam sehingga dapat mengurangi jumlah waktu komputasi.

Yang perlu diingat disini bahwa Infinite Impulse Response (IIR) dalam hal ini bukan berarti filter yang bekerja dari nilai negatif tak hingga sampai positif tak hingga. Pengertian sederhana untuk infinite impulse respon filter disini adalah bahwa output filter merupakan fungsi dari kondisi input sekarang, input sebelumnya dan output di waktu sebelumnya. Konsep ini kemudian lebih kita kenal sebagai *recursive filter*, yang mana melibatkan proses feedback dan feed forward.

Secara singkat, tahapan-tahapan untuk membuat filter digital IIR antara lain :

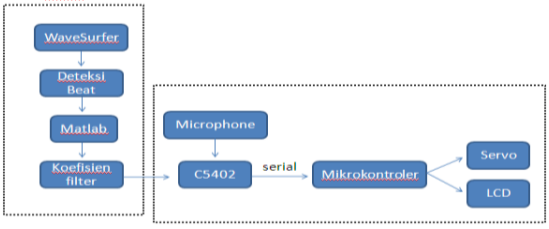
1. Menentukan respon frekuensi filter yang diinginkan.
2. Menghitung nilai koefesien filter dengan Matlab.
3. Menuliskan koefesien filter kedalam program filter.
4. Kompilasi program dan download kode mesin ke DSP.
5. Menguji sistem dengan memberikan sinyal input dari audio-out komputer dan mendengarkan hasilnya melalui speaker.



Gambar 1. Ilustrasi alur implementasi Filter IIR

3. PERANCANGAN SISTEM

Secara garis besar, penelitian ini terdiri dari 2 proses. Yang pertama yaitu mendeteksi beat di Wavesurfer sekaligus desain filter di Matlab. Yang kedua yaitu desain filter di TMS320C5402. Berikut gambaran umum dari sistem ini :



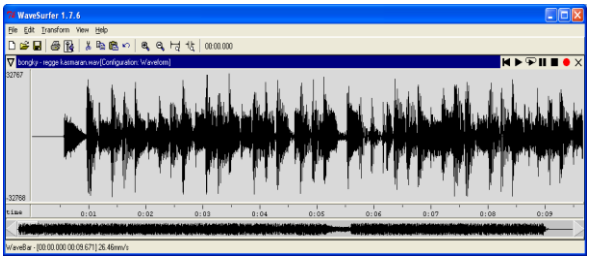
Gambar 2. Gambaran umum sistem

3.1 Proses di Wavesurfer

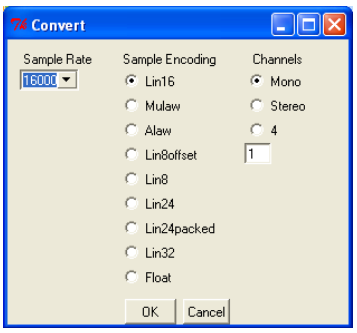
Sistematika pengerjaan tahap deteksi beat adalah sebagai berikut :

Gambar 3. Diagram blok proses di Wavesurfer

Tahap awal pada proses ini adalah membuka file lagu yang sudah ditentukan dengan WaveSurfer. Langkah kedua yaitu meng-convert lagu dari frekuensi sampling 44100 Hz stereo menjadi mono dengan frekuensi sampling 16000 Hz.



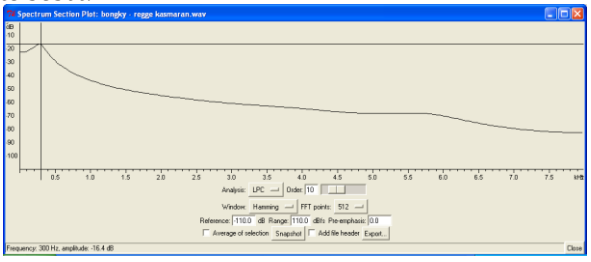
Gambar 4. Wavesurfer



Gambar 5. Convert Lagu

Langkah selanjutnya adalah memainkan lagu tersebut di wavesurfer dan mengamati frekuensinya di Spectrum Section Plot. Untuk memudahkan pengamatan, atur Analysis yang semula FFT menjadi LPC dengan Order 10.

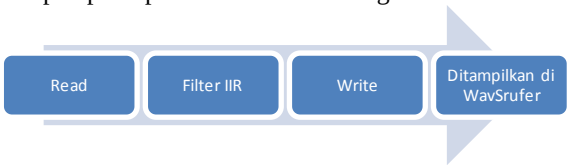
Mainkan lagu tersebut apabila pengaturan sudah selesai dan amatilah frekuensinya. Dengan Order 10, kita dapat dengan mudah menemukan frekuensi manakah yang muncul secara periodik. Kita boleh mengira-ngira karena setelah mendapatkan frekuensi, lagu tersebut akan di filter band-pass pada frekuensi tersebut.



Gambar 6. Spectrum Section Plot

3.2 Proses di Matlab

Akhir dari proses di WaveSurfer diatas adalah menemukan frekuensi yang diprediksi sebagai beat. Untuk membuktikan apakah frekuensi tersebut beat atau bukan maka perlu diproses di Matlab. Tahapan-tahapan pada proses ini adalah sebagai berikut :



Gambar 7. Proses di Matlab

Proses read yaitu proses untuk membaca frekuensi sampling (dalam Hertz) yang digunakan untuk mengkodekan data pada lagu tersebut ke dalam sebuah variabel. Contoh codenya :

```
[x,fs]=wavread('lagu.wav');
```

Proses selanjutnya yaitu filter IIR. Ada beberapa keunggulan dari filter IIR yang mendasari penulis memilih filter ini, salah satunya yaitu kecuraman. Di Matlab disediakan fungsi *butter* untuk filter IIR dan fungsi *fir2* untuk filter FIR. Fungsi tersebut akan menghasilkan koefisien yang nantinya digunakan untuk mengimplementasikan filter yang sudah didesain ke dalam board DSP.

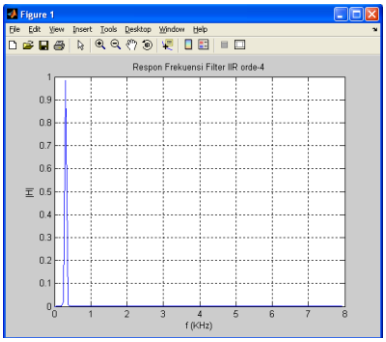
Beat yang berhasil dideteksi pada proses sebelumnya berada pada 300 Hz. Oleh karena itu pada proses ini akan dilakukan band-pass filter dengan puncak 300 Hz (cutoff 280 – 320 Hz). Grafik respon frekuensi juga ditampilkan untuk mengetahui seberapa bagus filter yang sudah didesain. Contoh codenya :

```
n=4;     %orde
f=8;     %setengah frekuensi sampling
w1=0.035;    %280
w2=0.04;     %320
wn=[w1 w2];
[b,a]=butter(n,wn);
[y,fs]=filter(b,a,x);
```

Lagu hasil filter harus dibuka lagi dengan WaveSurfer untuk membuktikan apakah frekuensi 300 Hz benar-benar beat atau bukan. Apabila benar beat, akan terlihat amplitudo dari frekuensi tersebut yang muncul secara periodik. Contoh codenya :

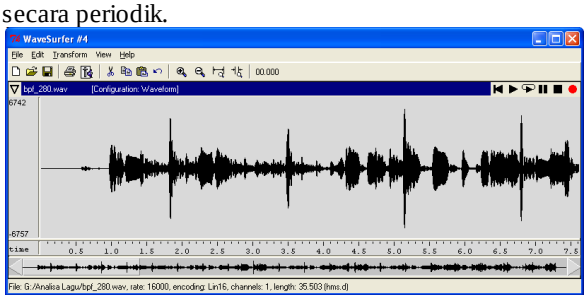
```
wavwrite(y,16000,'lagu_bpf.wav');
```

Selain itu, grafik respon frekuensi perlu ditampilkan untuk mengetahui seberapa bagus filter yang sudah didesain. Sehingga apabila respon frekuensinya tidak sesuai dengan apa yang kita inginkan, kita bisa memperbaikinya. Respon frekuensi filter dengan frekuensi cutoff 280 – 320 Hz adalah sebagai berikut :



Gambar 8. Respon Frekuensi Filter

Seperti yang telah dijelaskan diatas, untuk membuktikan apakah frekuensi 300 Hz benar-benar beat atau bukan kita harus membuka lagu yang sudah difilter dengan Wavesurfer. Apabila benar beat, akan terlihat amplitudo dari frekuensi tersebut yang muncul secara periodik.



Gambar 9. Lagu yang sudah difilter

3.3 Desain Filter di Matlab

Pada tahap ini akan digunakan fungsi *butter* yang disediakan oleh Matlab untuk mendapatkan koefesien filter dengan metode butterworth.

Perintah `[b,a] = butter(N,Wn);` pada Matlab digunakan untuk mendisain filter IIR lowpass menggunakan metode butterworth **orde N** dengan **frekuensi cut-off pada Wn**, dan menghasilkan koefesien filter pada vektor B (numerator) dan vektor A (denumerator) sebanyak N+1. Nilai dari frekuensi cut-off Wn haruslah bernilai antara 0.0 < Wn < 1.0, dimana *1.0 menunjukkan setengah dari frekuensi sampling*.

Langkah-langkah mendesain filter IIR menggunakan fungsi BUTTER adalah sebagai berikut :

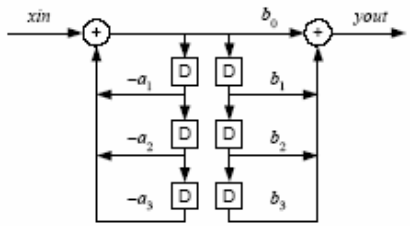
1. Pilih orde filter, misal N=5
2. Menentukan frekuensi cut-off
3. Koefesien filter dapat dihitung menggunakan perintah BUTTER pada Matlab.
4. Simpan nilai koefesien pada file, yang nantinya digunakan untuk mengimplementasikan filter IIR dengan konvolusi pada pemrograman DSP.

3. 4 Implementasi Filter IIR ke TMS320C5402

Fungsi transfer filter IIR adalah

$$H[z] = \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-m}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}}$$

Seperti diagram flow Direct Form II yang ditunjukkan oleh gambar 1, maka langkah yang harus dilakukan ditunjukkan seperti pada gambar 5 berikut.



Gambar 10. Diagram flow Direct Form II filter IIR

Dari diagram flow diatas, didapat rumus untuk menghitung nilai output yaitu :

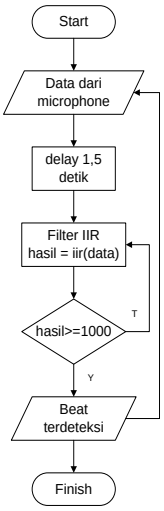
```
w=xin-(a1*d[0])-(a2*d[1])-(a3*d[2]);
yout=b0*w+(b1*d[0])+(b2*d[1])+(b3*d[3]);
```

Variabel ‘a’ dan ‘b’ adalah koefisien yang dihasilkan dari proses di Matlab. Agar terjadi feedback, maka input sekarang harus digeser.

```
d[2] = d[1];
d[1] = d[0];
d[0] = w;
```

3.5 Deteksi Beat di TMS320C5402

Pada tahap ini, peneliti memanfaatkan fungsi delay. Fungsi delay tidak akan memberatkan tugas DSK. Justru lebih meringankan karena DSK tidak akan terus-menerus memproses sinyal. DSK akan memproses sinyal input hanya pada waktu delay selesai sampai beat terdeteksi. Untuk lebih jelas berikut flowchart dari metode ini :



3. 5 Komunikasi Serial

Pada penelitian ini, komunikasi serial dibutuhkan untuk menghubungkan DSK dengan mikrokontroller. DSK akan mengirimkan karakter sebagai penanda bahwa beat terdeteksi. Kemudian mikrokontroller akan menerima karakter tersebut sebagai penanda untuk menggerakkan servo.

Sebenarnya tahap ini cukup mudah karena hanya membutuhkan header `uart.h` dan fungsi untuk mengirim karakter ke uart yaitu `uart_fputc()`.

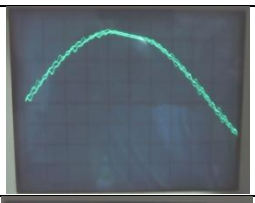
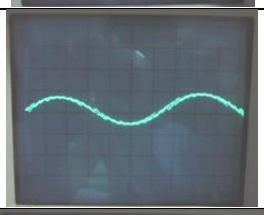
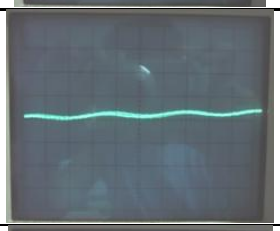
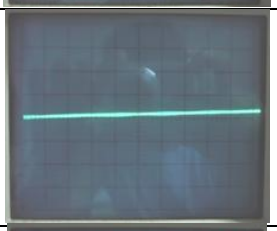
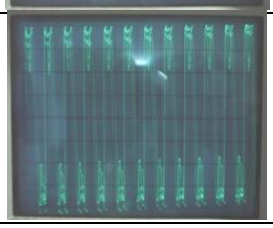
Sebelum memulai program, inisialisasi terlebih dahulu pengaturan standar UART seperti di bawah ini :

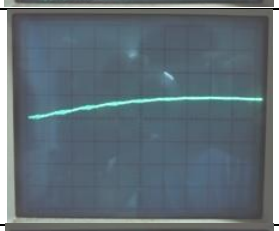
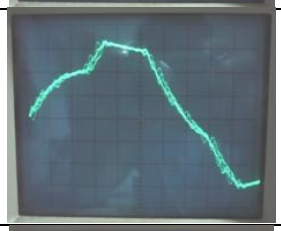
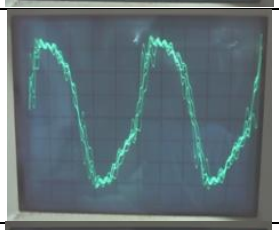
```
UartBaud          baud = UART_BAUD_9600;
UartWordLen       wordLength = UART_WORD8;
UartStopBits      stopBits = UART_STOP1;
UartParity        parity = UART_EVEN_PARITY;
UartFifoControl   fifo = UART_FIFO_DISABLE;
UartLoop          loop = UART_NO_LOOPBACK;
```

4. Pengujian Sistem

4.1 Pengujian Filter

Dibuat filter IIR 3 jenis yaitu lowpass, bandpass dan highpass. Semuanya bisa digunakan untuk mendeteksi beat. Pada tahap ini, masing-masing jenis filter tersebut akan diuji dengan memberi input frekuensi dibawah, diatas atau tepat pada frekuensi cutoff-nya.

Filter	Input (Hz)	Hasil
Lowpass (cutoff 400 Hz)	100	
	300	
	600	
	1000	
Bandpass (cutoff 250 – 350 Hz)	100	

	300	
	600	
Highpass (200 Hz)	50	
	150	
	400	
	1000	

Dari hasil pengujian bisa dilihat dengan jelas kualitas dari masing-masing filter. Filter lowpass menunjukkan respon yang baik. Frekuensi 100 dan 300 Hz masih dilewatkan. Frekuensi 600 diredam karena melebihi cutoff.

Filter bandpass menunjukkan respon yang tidak baik. Karena diberi input berapapun hasilnya sama saja. Ini menunjukkan bahwa filter IIR bandpass tidak bisa digunakan apabila frekuensi cutoff nya dibawah 1 KHz.

Sedangkan filter highpass menunjukkan respon yang kurang baik. Filter 50 dan 100 Hz yang seharusnya diredam akan tetapi ternyata tetap diloloskan.

4.2 Pengujian Jarak Microphone Terhadap Sumber Suara

Suara adalah input dalam sistem ini. Jauh-dekatnya microphone terhadap sumber suara tentulah sangat berpengaruh. Semakin jauh dari sumber suara, suara akan semakin mengecil dan akan semakin rentan terhadap noise karena posisi noise dan suara yang diterima akan seimbang. Tetapi semakin dekat, maka noise akan tertutup oleh sumber suara sehingga sistem dapat bekerja maksimal. Berikut hasil pengujian jarak :

- Lagu yang dipilih penulis untuk penelitian ini berdurasi 4 menit 15 detik. Karena terlalu lama maka untuk pengujian ini lagu tersebut dipotong menjadi 10 detik (6 beat).
- Terdapat 2 parameter untuk pengujian ini :
- Beat yang terdeteksi berjumlah 6.
- Saat lagu berhenti, informasi jumlah beat di LCD juga akan berhenti (tidak bertambah).
- Pengujian dikatakan berhasil (√) apabila kedua kondisi diatas terpenuhi. Apabila salah satu kondisi saja tidak terpenuhi maka tetap dikatakan tidak berhasil (x).

Jarak (cm)	**Percobaan ke-**				
	1	**2**	**3**	**4**	**5**
5	√	√	√	√	√
10	√	√	√	√	√
15	√	**x**	**x**	√	√
20	**x**	√	√	**x**	√
25	**x**	**x**	√	**x**	√
30	√	**x**	**x**	**x**	**x**

Dari tabel diatas dapat dilihat bahwa semakin jauh dari sumber suara, program semakin berjalan tidak maksimal. Apabila microphone jauh dari sumber suara, maka noise-noise disekitar akan semakin terdengar. Tetapi semakin dekat, maka noise akan tertutup oleh sumber suara sehingga sistem dapat bekerja maksimal.

4.2 Pengujian Noise

Pengujian selanjutnya yaitu pengujian noise. Noise disini bersumber dari suara-suara obrolan beberapa orang dengan radius lebih dari satu meter dari microphone.

- Lagu yang dipilih penulis untuk penelitian ini berdurasi 4 menit 15 detik. Karena terlalu lama maka untuk pengujian ini lagu tersebut dipotong menjadi 10 detik (6 beat).
- Terdapat 2 parameter untuk pengujian ini :
- Beat yang terdeteksi berjumlah 6.
- Saat lagu berhenti, informasi jumlah beat di LCD juga akan berhenti (tidak bertambah).

- Pengujian dikatakan berhasil (√) apabila kedua kondisi diatas terpenuhi. Apabila salah satu kondisi saja tidak terpenuhi maka tetap dikatakan tidak berhasil (x).

Jumlah (orang)	Percobaan ke-				
	1	2	3	4	5
1	√	√	√	√	√
2	√	√	√	√	√
3	**x**	√	√	√	√
4	√	√	**x**	**x**	√
5	**x**	√	√	√	**x**
6	**x**	√	**x**	√	√
7	**x**	**x**	√	√	**x**
8	√	**x**	**x**	**x**	√
9	**x**	**x**	**x**	√	√
10	**x**	**x**	**x**	**x**	**x**

Dari data pengujian diatas bisa dilihat bahwa sistem tidak bisa bekerja maksimal jika ada 8 orang berbicara secara bersamaan. Karena noise yang ditimbulkan melebihi ambang batas. Suara musik dan suara manusia (noise) bercampur jadi satu. Sehingga apabila music berhenti, sistem akan terus mendeteksi beat karena menganggap suara manusia memenuhi target frekuensi yang ditetapkan.

5. Kesimpulan

Berdasarkan hasil pengujian dan analisa, dapat disimpulkan :

1. Filter IIR bandpass tidak bisa bekerja dengan frekuensi cutoff di bawah 1 KHz.
2. Deteksi beat bisa dilakukan dengan melihat sinyal lagu dan membuktikannya dengan memfilter lagu memakai filter IIR bandpass.
3. Jarak ideal microphone terhadap sumber suara adalah 5-15 cm.
4. Metode beat detection kurang baik apabila diterapkan pada robot KRSI.

6. Daftar Pustaka

1. TMS320C54x Chip Support Library API Reference Guide.
2. Frederich Patin, "*Beat Detection Algorithms*", 2003.
3. Tri Budi Santoso, dkk, "*Pengenalan Prosesor DSP*", PENS-ITS Surabaya.
4. Tri Budi Santoso, dkk, "*Implementasi Filter IIR secara Real Time pada TMS 32C5402*", PENS-ITS Surabaya.
5. Hary Oktavianto, "*Codec dan Sampling*", Modul Praktikum Pengolahan Sinyal PENS ITS, 2000.
6. Hary Oktavianto, "*4-Preset Equalizer menggunakan filter IIR*", Modul Praktikum Pengolahan Sinyal PENS ITS, 2000.
7. Hary Oktavianto, "*Real-Time Filter FIR*", Modul Praktikum Pengolahan Sinyal PENS ITS, 2000.
8. Ahmad Bahtiar, "*Efek Gitar Digital dengan Parameter yang Dapat Diubah Menggunakan TMS320VC5402*", PENS-ITS Surabaya, 2007.
9. Daniel P.W. Ellis, "*Beat Tracking by Dynamic Programming*", New York, 2007.
10. Erick D.S, "*Tempo and Beat Analysis of Acoustic Musical Signal*", Cambridge, 1997.
11. DSP/BIOS Hardware and Software UART Device Drivers.
12. Implementing a Software UART on the TMS320C54x with the McBSP and DMA.