PROPOSAL PRAKTIKUM
ET3005 PENGOLAHAN SINYAL WAKTU DISKRIT

MODUL 2

"AUDIO ENCODING AND DECODING"

ANGGOTA : BARIQ SUFI FIRMANSYAH (18116004)

: ELLENA STEFANI (18116019)

: NADIA SARASWATI (18116027)

KELOMPOK : 6

HARI : JUMAT

TANGGAL : 23 NOVEMBER 2018

WAKTU : 13.00-15.00

ASISTEN : WAKHIDATI HIDAYAH

LAB. TELEKOMUNIKASI RADIO & GELOMBANG MIKRO

PROGRAM STUDI TEKNIK TELEKOMUNIKASI – STEI – ITB

I. PENDAHULUAN

1.1 Latar Belakang

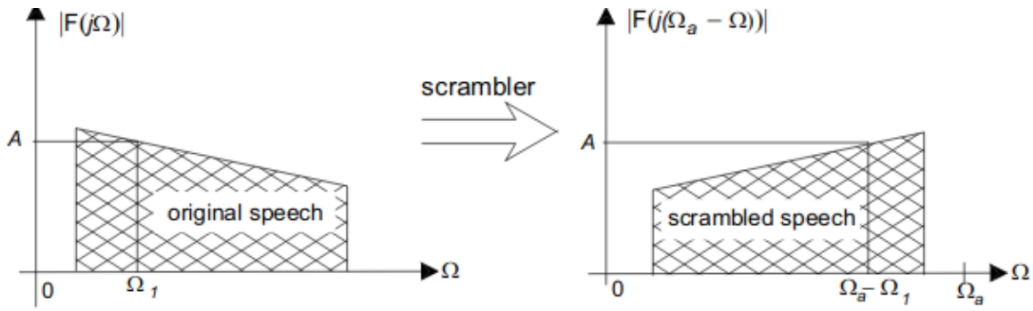
Pada zaman sekarang, teknologi berkembang dengan sangat pesat. Termasuk juga dengan dunia Telekomunikasi. Seiring perkembangan di dunia teknologi maka kejahatan melalui teknologi juga semakin meningkat. Salah satu contoh kejahatan menggunakan teknologi adalah Penyadapan.

Penyadapan di dalam Telekomunikasi adalah Pemantauan percakapan telepon atau alat komunikasi lainnya oleh pihak ketiga yang dilakukan secara rahasia. Dengan adanya penyadapan tersebut, pihak yang bersangkutan sudah tidak mempunyai privasi, sehingga hal ini sangat dilarang oleh negara. Untuk mencegah penyadapan tersebut, maka perlu dibuat sebuah sistem untuk mengacak sinyal suara.

1.2 Praktikum kali ini mempunyai tujuan-tujuan sebagai berikut:

1. Mengimplementasikan skema *Audio Encoding*
2. Mengimplementasikan skema *Audio Decoding*

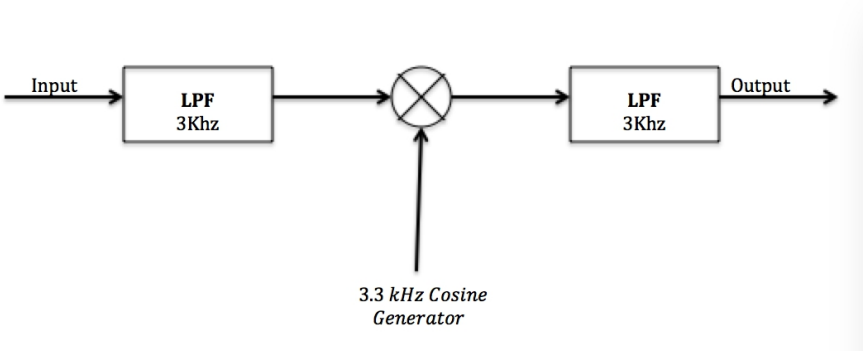
II. SKEMA DESAIN



Gambar 1 Skema *Audio Encoding*

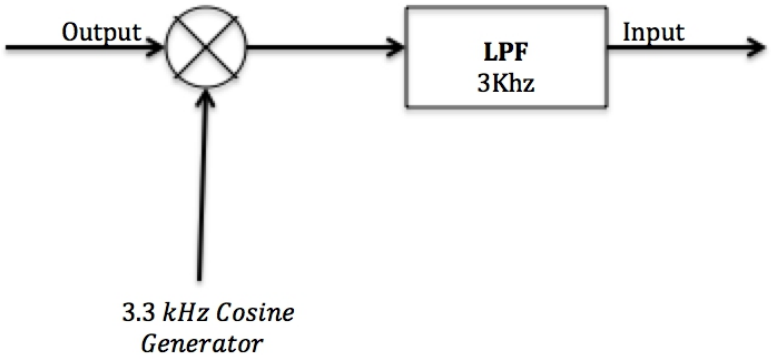
Disini akan digunakan skema pencerminan domain frekuensi, dimana setiap komponen frekuensi akan terbalik. Frekuensi tinggi akan ditranslasi menjadi frekuensi rendah *vice versa*[1]. Komponen frekuensi **Ω1** akan ditranslasi menjadi frekuensi baru yaitu **Ωa-Ω1.** Dimana Ωa = 2π x 3300Hz merupakan komponen frekuensi yang diketahui oleh pengirim dan penerima. Gelombang yang dihasilkan akan berbentuk acak sampai frekuensi kembali seperti semula.

Spektrum pada gambar bagian kiri akan ditranslasi menjadi bentuk spektrum baru seperti pada gambar kanan. Decoder akan mengembalikan hubungan frekuensi seperti semula. Proses *encoding* dan *decoding* pada sistem pengacak suara dapat dilihat dari diagram blok dibawah ini.



Gambar 2 Proses *Encoding*

Dari blok diagram diatas, dapat dilihat bahwa proses *encoding* sinyal masukan yang berupa suara di filter menggunakan LPF. Hasil filter akan dimodulasi dengan sinyal sinusoidal yang berfungsi sebagai *noise.* Setelah dimodulasi sinyal akan difilter kembali menggunakan filter lowpass.



Gambar 3 Proses *Decoding*

Proses *Decoding* dimulai dengan memodulasi sinyal hasil modulasi pada proses *encoding.* Setalh itu sinyal akan difilter kembali menggunakan filter *lowpass.*

Spesifikasi Filter *Lowpass* :

$fa(frekuensi\ pemodulasi) = 3300\ Hz$

$fp(frekuensi\ passband) = 2950\ Hz$

$fs(frekuensi\ stopband) = 3050\ Hz$

$frekuensi\ sampling = 48000\ Hz$

$\Omega a = 2\pi \times 3300\ Hz = 6600\ \pi\ rad/s$

$\Omega p = 2\pi \times 2950\ Hz = 5900\ \pi\ rad/s$

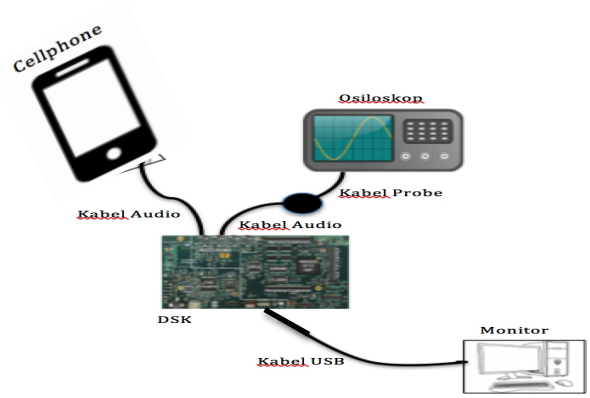
$\Omega s = 2\pi \times 3050\ Hz = 6100\ \pi\ rad/s$

III. PERANGKAT YANG DIPERLUKAN

1. Satu set komputer
2. 2 buah DSK TMS320C6713
3. Osiloskop
4. Kabel USB
5. Kabel Audio
6. Speaker

IV. PROSEDUR PRAKTIKUM

- Percobaan A : Melakukan filtering lowpass pada sinyal audio

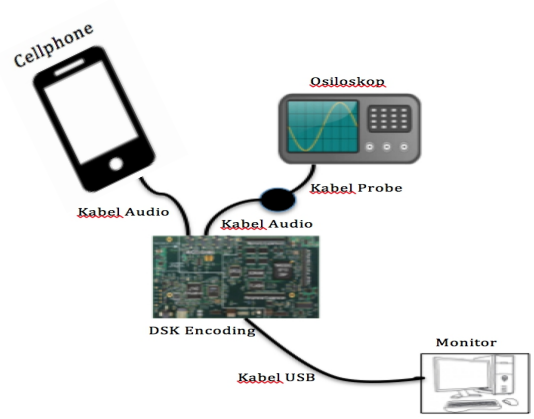


Gambar 4 *Set-Up* Perangkat

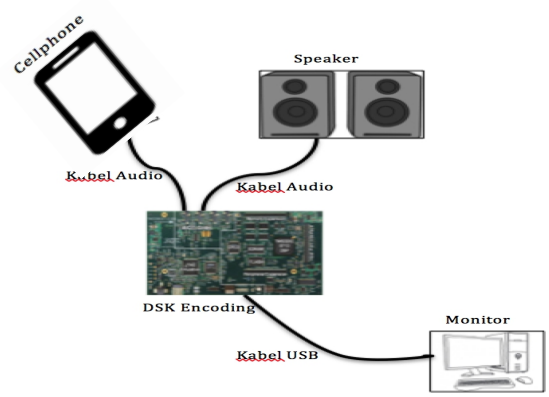
1. Melakukan kalibrasi pada osiloskop.
2. Pastikan DSK masih dalam keadaan mati
3. Perhatikan **Gambar 4**
4. Nyalakan komputer dan hubungkan kabel USB DSK ke konektor USB komputer
5. Hubungkan kabel Audio cellphone menuju port line in DSK
6. Hubungkan kabel Audio port line out menuju osiloskop dengan menggukan kabel probe
7. Pastikan rangkaian perangkat yang terhubung sama dengan **Gambar 4**
8. Nyalakan DSK Encoding dan lakukan prosedur DSK Diagnostic. Klik Start dan tunggu hingga proses diagnostic selesai dan keluar tulisan “PASS” .
9. Buka software CCS
10. Klik menu **Debug>connect** untuk menghubungkan CCS dengan DSK Encoding
11. Buka project Filter FIR
12. Bunyikan sinyal audio pada *cellphone*
13. Ubah koefisien filter sesuai pada hasil simulasi octave pada file coeff.h
14. Jalankan program build project (F7) dan load program ke DSK Encoding dengan mengklik menu **File>load program.**

15. Jalankan program yang telah di load ke DSK dengan menekan F5 sehingga output hasil pemrosesan DSK terlihat di osiloskop
16. Amati dan catat tegangan di osiloskop pada frekuensi 100-3100 Hz dengan stepsize 100 Hz
17. DSK ini selanjutnya akan digunakan sebagai DSK encoding, maka perlu dilakukan reset agar memori yang tersimpan bisa dibersihkan.

- Percobaan B : Proses *Encoding Audio*



Gambar 5 *Set-Up* Perangkat Encoding

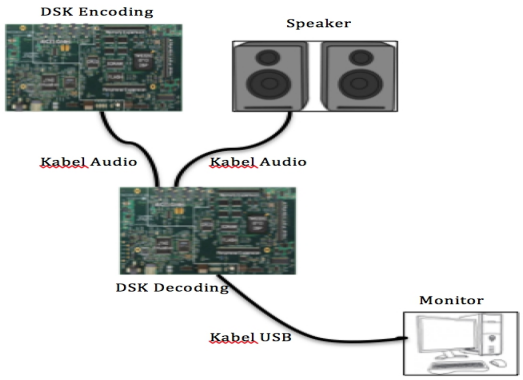


Gambar 6 *Set-Up* Perangkat Encoding

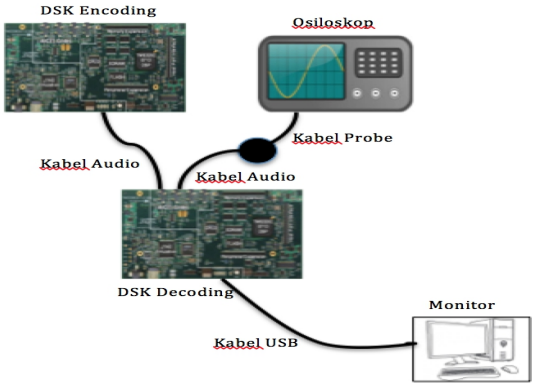
1. Melakukan kalibrasi pada osiloskop.
2. Pastikan DSK Encoding masih dalam keadaan mati
3. Perhatikan **Gambar 5**
4. Nyalakan komputer dan hubungkan kabel USB DSK ke konektor USB komputer
5. Hubungkan kabel Audio cellphone menuju port line in DSK Encoding
6. Hubungkan kabel Audio port line out menuju osiloskop dengan menggukan kabel probe
7. Pastikan rangkaian perangkat yang terhubung sama dengan **Gambar 5**
8. Nyalakan DSK Encoding dan lakukan prosedur DSK Diagnostic. Klik Start dan tunggu hingga proses diagnostic selesai dan keluar tulisan “PASS” .
9. Buka software CCS
10. Klik menu **Debug>connect** untuk menghubungkan CCS dengan DSK Encoding
11. Buka project Audio Encoder dengan mengklik menu **Project>open** dan pilihlah file AudioEncoder.pjt
12. Ubah koefisien filter sesuai pada hasil simulasi octave pada file coeff.h

13. Jalankan program build project (F7) dan load program ke DSK Encoding dengan mengklik menu **File>load program**. Pada folder debug, double klik file AudioEncoder.out
14. Jalankan program yang telah di load ke DSK Encoding dengan menekan F5 sehingga output hasil pemrosesan DSK terlihat di osiloskop
15. Amati dan catat tegangan di osiloskop pada frekuensi 100-3100 Hz dengan stepsize 100 Hz.
16. Cabut kabel probe yang terhubung pada osiloskop lalu hubungkan kabel audio port line out ke speaker
17. Ulangi langkah 14 dan bandingkan keluaran speaker dengan sinyal input audio pada cellphone.

- Percobaan C : Proses *Decoding Audio*



Gambar 7 *Set-Up* Perangkat Proses Decoding



Gambar 8 *Set-Up* Perangkat Proses Decoding

1. Pastikan DSK Decoding masih dalam keadaan mati
2. Perhatikan **Gambar 7**
3. Nyalakan komputer dan hubungkan kabel USB DSK Decoding ke konektor USB komputer
4. Hubungkan kabel Audio line out DSK Encoding menuju port line in DSK Decoding
5. Hubungkan kabel Audio port line out DSK Decoding menuju osiloskop dengan menggukan kabel probe
6. Pastikan rangkaian perangkat yang terhubung sama dengan **Gambar 7**
7. Nyalakan DSK Decoding dan lakukan prosedur DSK Diagnostic. Klik Start dan tunggu hingga proses diagnostic selesai dan keluar tulisan “PASS” .
8. Buka software CCS
9. Klik menu **Debug>connect** untuk menghubungkan CCS dengan DSK Decoding
10. Buka project Audio Decoder dengan mengklik menu **Project>open** dan pilihlah file AudioDecoder.pjt

11. Ubah koefisien filter sesuai pada hasil simulasi octave pada file coeff.h
12. Jalankan program build project (F7) dan load program ke DSK Decoding dengan mengklik menu **File>load program**. Pada folder debug, double klik file AudioDecoder.out
13. Jalankan program yang telah di load ke DSK Decoding dengan menekan F5 sehingga output hasil pemrosesan DSK terlihat di osiloskop
14. Amati dan catat tegangan di osiloskop pada frekuensi 100-3100 Hz dengan stepsize 100 Hz.
15. Cabut kabel probe yang terhubung pada osiloskop lalu hubungkan kabel audio port line out ke speaker
16. Ulangi langkah 13 dan bandingkan keluaran speaker dengan sinyal input audio pada cellphone.

V. SCRIPT DAN SIMULASI OCTAVE

Berikut ini merupakan dokumentasi hasil simulasi octave. Dengan simulasi octave ini, akan dihasilkan spektrum sinyal encoding dan sinyal decoding. Langkah pertama yang dilakukan adalah membaca sinyal audio dengan menggunakan fungsi audioread dengan parameternya adalah 'test3.wav'.

AudioEncoderDecoder.m

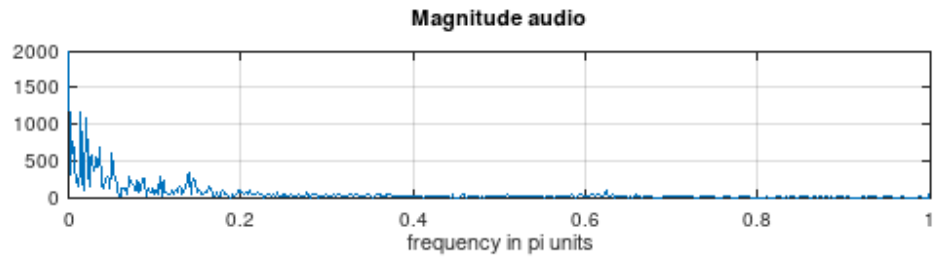
```
1  %Input sinyal Audio
2  [x,fs]=audioread('test3.wav');
```

Gambar 9 Membaca Sinyal Audio

Kemudian script selanjutnya adalah perintah untuk menampilkan spektrum sinyal audio.

```
4  %Plot sinyal audio
5  n = 1:length(x); k = 0:500; w = (pi/500)*k;
6  X=x(:,1)'*(exp(-j*pi/500)) .^ (n'*k);
7  magX = abs(X); angX = angle(X);
8  subplot(4,2,1); plot(k/500,magX);grid
9  xlabel('frequency in pi units'); title('Magnitude audio')
10
```

Gambar 10 Perintah untuk menampilkan spectrum sinyal audio

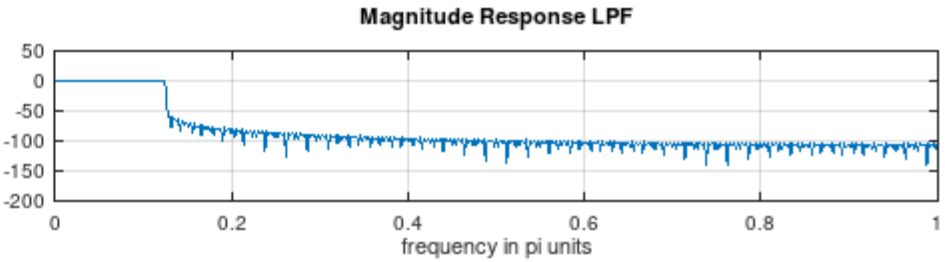


Gambar 11 Spektrum Magnituda Sinyal Audio

Sesuai pada gambar 2. sebelum dilakukan encoding akan dilakukan proses pemfilteran dengan filter low-pass. Filter ini akan melewatkan sinyal audio dibawah 3Khz. Perintah dibawah adalah konfigurasi filter dan plotting filter lowpass FIR dengan window hamming.

```
11  %LowPassFilter
12  wp=2*2950*pi/fs; ws=2*3050*pi/fs;
13  tr_width=ws-wp;
14  M =ceil(6.6*pi/tr_width) + 1;
15  n =[0:1:M-1];
16  wc=(ws+wp)/2;
17  hd=ideal_lp(wc,M);
18  w_ham=(hamming(M))';
19  h=hd.*w_ham;
20  [db,mag,pha,w]=freqz_m(h,[1]);
21  delta_w = 2*pi/1000;
22  subplot(4,2,2); plot(w/pi,db); xlabel('frequency in pi units');
23  title('Magnitude Response LPF'); grid;
24
```

Gambar 12 Perintah untuk membuat lowpass filter FIR dengan window hamming



Gambar 13 Spektrum magnituda filter LPF FIR

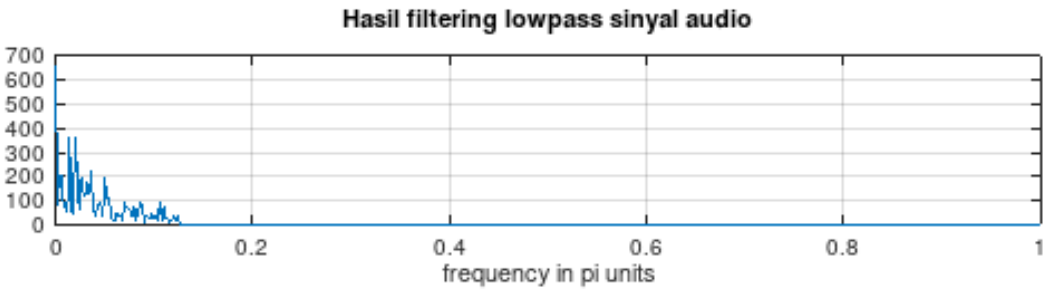
Bisa dilihat bahwa filter digital akan melewatkan frekuensi 3Khz*pi/fs. Dengan fs (frekuensi sampling) bernilai 48000.

Kemudian dilakukan pemfilteran pada sinyal audio dengan menggunakan perintah seperti berikut.

```
24
25  %filtering sinyal audio
26  m=1:length(x);
27  a=[1];
28  x4=filter(h,a,x);
29  X4=x4(:,1)'*(exp(-j*pi/500)) .^ (m'*k);
30  magX4 = abs(X4); angX4 = angle(X4);
31  subplot(4,2,3); plot(k/500,magX4/pi);xlabel('frequency in pi units');
32  title('Hasil filtering lowpass sinyal audio'); grid;
```

Gambar 14. Perintah untuk memfilter sinyal audio

Sehingga didapatkan sinyal audio hasil filtering sebagai berikut.‘



Gambar 15 Sinyal audio hasil filtering low pass

Kemudian sinyal audio hasil filtering lowpass dibunyikan dengan menggunakan perintah soundsc.

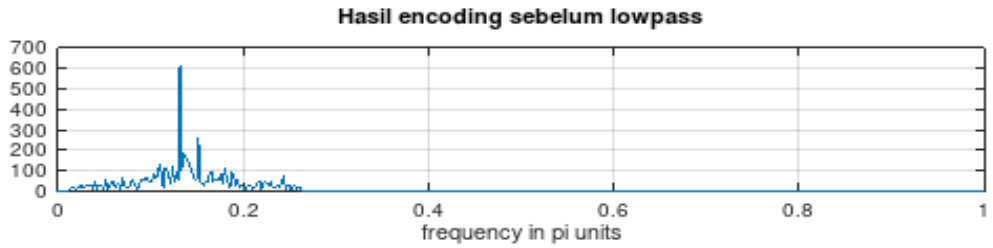
```
34   %Bunyikan sinyal hasil filtering lowpass
35   nbits=16;
36   soundsc(x4,fs,nbits);
37
```

Gambar 16 Membunyikan sinyal audio yang telah di filter lowpass

Sesuai pada gambar 2, dilakukan proses encoding dengan mengalikan sinyal cosinus frekuensi 3.3Khz. Berikut adalah scriptnya.

```
38   %Lakukan Encoding
39   L=length(x);
40   m=1:1:L;
41   x3=x4(:,1)'.*cos(2*3300*pi*m/fs); %encoding
42   X3=x3*(exp(-j*pi/500)) .^ (m'*k);
43   magX3 = abs(X3); angX3 = angle(X3);
44   subplot(4,2,4); plot(k/500,magX3/pi);grid
45   xlabel('frequency in pi units'); title('Hasil encoding sebelum lowpass');
46
```

Gambar 17 perintah untuk melakukan encoding

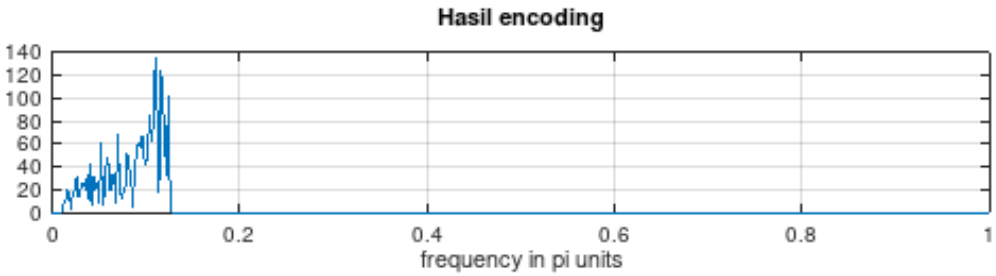


Gambar 18 Hasil encoding, namun masih belum dilowpass di frekuensi 3Khz

Setelah dilakukan proses lowpass, maka akan didapatkan hasil akhir encoding. Perintah dibawah ini melakukan proses filtering pada hasil encoding gambar 18 kemudian dilakukan plotting dan pembunyian suara.

```
47  %dilakukan filtering lowpass pada x3
48  m=1:length(x3);
49  a=[1];
50  x5=filter(h,a,x3);
51  X5=x5*(exp(-j*pi/500)) .^ (m'*k);
52  magX5 = abs(X5); angX5 = angle(X5);
53  subplot(4,2,5); plot(k/500,magX5/pi);grid
54  xlabel('frequency in pi units'); title('Hasil encoding');
55  soundsc(x5,fs,nbits);
```

Gambar 19 Perintah untuk filtering low pass sinyal hasil encoding pada gambar 18



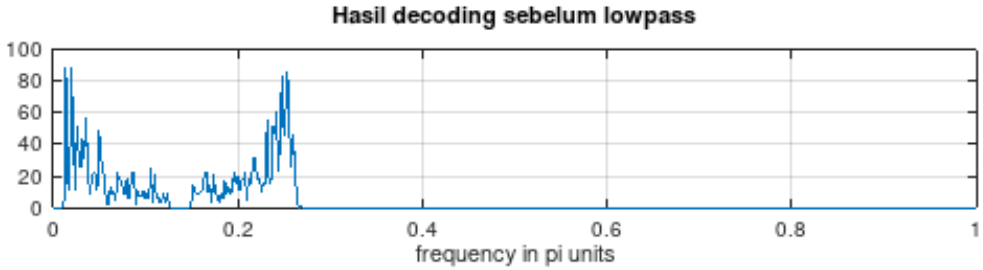
Gambar 20 Spektrum sinyal gambar 18 setelah dilakukan filtering lowpass

Setelah dibunyikan, hasil simulasi menghasilkan suara yang tidak jelas yang menandakan proses encoding telah sesuai harapan. Langkah selanjutnya adalah proses decoding pada *receiver*. Caranya mirip dengan proses encoding, yaitu dengan dikalikan sinyal cosinus frekuensi 3.3Khz.

```
56
57  %Lakukan Decoding
58  L=length(x);
59  m=1:1:L;
60  x6=x5.*cos(2*3300*pi*m/fs); %decoding
61  X6=x6*(exp(-j*pi/500)) .^ (m'*k);
62  magX6 = abs(X6);
63  subplot(4,2,6); plot(k/500,magX6/pi);grid
64  xlabel('frequency in pi units'); title('Hasil decoding sebelum lowpass');
```

Gambar 21 Perintah *decoding*

Sehingga didapatkan spektrum sebagai berikut. Hasil dibawah ini masih belum final. Sesuai skema pada gambar 3, akan dilakukan proses filtering lowpass lagi sebelum hasil decoding akhir.

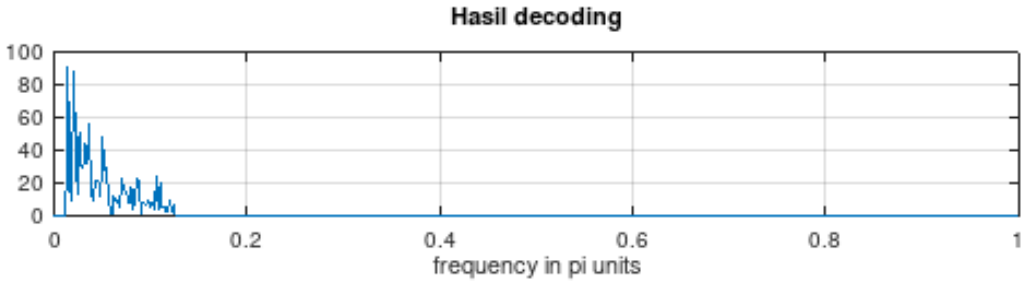


Gambar 22 Hasil *decoding* sebelum dilakukan filtering lowpass

Kemudian langkah terakhir decoding adalah dengan dilakukan filtering lowpass. Berikut ini adalah perintah untuk melakukan filtering lowpass, plotting spektrumnya, dan membunyikan sinyalnya.

```
66  %dilakukan filtering lowpass pada x6
67  m=1:length(x6);
68  a=[1];
69  x7=filter(h,a,x6);
70  X7=x7*(exp(-j*pi/500)) .^ (m'*k);
71  magX7 = abs(X7);
72  subplot(4,2,7); plot(k/500,magX7/pi);grid
73  xlabel('frequency in pi units'); title('Hasil decoding');
74  soundsc(x7,fs,nbits);
```

Gambar 23 Perintah *filtering* lowpass agar didapat sinyal *decoding final*



Gambar 24 Sinyal hasil *decoding*

Setelah dilakukan proses decoding, ternyata spektrumnya mirip dengan spektrum sinyal pada gambar 15. Dan setelah didengarkan, bunyi sinyalnya sudah bisa didengar dengan baik. Ini menunjukkan bahwa tahap *decoding* telah berhasil dilakukan.

LOGBOOK MODUL 2-*AUDIO ENCODING AND DECODING*

TABEL 1 HASIL PENGUKURAN AMPLITUDA

FREKUENSI	**MAGNITUDE SPEKTRUM SINYAL INFORMASI**	**MAGNITUDE SPEKTRUM HASIL ENCODING**	**MAGNITUDE SPEKTRUM HASIL DECODING**
100			
200			
300			
400			
500			
600			
700			
800			
900			
1000			
1100			
1200			
1300			
1400			
1500			
1600			
1700			
1800			
1900			
2000			
2100			
2200			

2300			
2400			
2500			
2600			
2700			
2800			
2900			
3000			

Gambar spektrum magnituda sinyal audio yang di lowpass	Gambar spektrum magnituda sinyal audio hasil *encoding*	Gambar spektrum magnituda sinyal audio hasil *decoding*

Apakah informasi suara hasil encoding masih bisa didengar dengan jelas?

Apakah informasi suara hasil decoding bisa didengar dengan jelas?