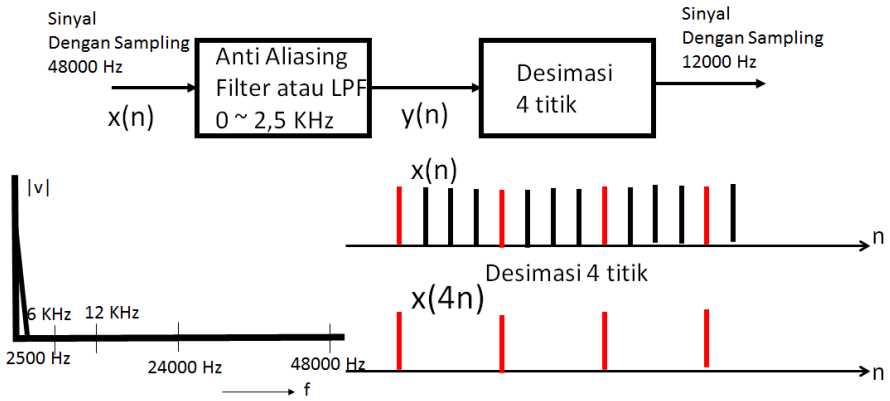
IV. PRAKTIKUM 3 – PENGONDISIAN SINYAL

IV.1 PENDAHULUAN

a). Filter Antialiasing

Untuk menghindari kondisi yang tidak menentu karena variasi USB-Soundcard yang ada dipasaran, praktikum ini dilakukan dengan penananganan pengaturan USB-Soundcard dengan ragam pengaturan default, termasuk sampling frekuensi default adalah 48 kHz. Sedangkan program WSPR membutuhkan pengambilan sampling frekuensi 12 kHz. Oleh karena itu, perlu downsampling dari 48 kHz menjadi 12 kHz (perhatikan Gambar IV-1). Harus diingat bahwa sebelum proses downsampling dari 48 kHz ke 12 kHz, filter lowpass 6 kHz harus disisipkan untuk menghindari terjadinya aliasing.



Gambar IV-1: Proses downsampling dengan mengunakan teknik desimasi di domain waktu

Filter digital antialiasing dengan cutoff 2500 Hz dengan sampling frekuensi 48 kHz yang dipergunakan adalah jenis filter IIR yang diturunkan dari pendekatan fillter lowpass RC. Persamaan beda dari IIR beserta harga koefisiennya, serta highlight sintak pemrograman dapat dilihat pada Gambar IV-2.

$$y(n) = a_1 y(n-1) + b_1 x(n) + c_1 x(n-1)$$

$$\tau = \frac{1}{2\pi(48000)}$$

$$a_1 = \frac{1}{1 + T/2\tau}$$

$$b_1 = c_1 = \frac{T/2\tau}{1 + T/2\tau}$$

```
float pi = 3.141592654;
float fc = 2500;
float frq=48000;
float a1,b1,c1,tau1;

 tau1=1/(2*pi*fc);
 a1=(1.0-1.0/(2.0*tau1*frq))/(1.0+1.0/(2.0*tau1*frq));
 b1=c1=(1.0/(2.0*tau1*frq))/(1.0+1.0/(2.0*tau1*frq));

 buf2_int16[0]=buf2[0];
 for (i = 1; i < buffer_frames_48; i++){
   buf2_float[i]=a1*buf2_float[i-1]+b1*(float)buf2[i]+c1*(float)buf2[i-1];
   buf2_int16[i]=(int16_t)buf2_float[i];
 }
```

Gambar IV-2: Filter Anti Aliasing dengan frekuensi cutoff 2500 Hz, frekuensi sampling 48000 Hz.

Hasil dari bagian ini adalah sinyal AFSK panjang 114 detik dengan frekuensi sampling 12 KHz. Bentuk gelombang akan diamati dengan memloting segmen sinyal (misalnya 256 sampel) dengan bantuan antarmuka pipa GNUPLOT. Para siswa dapat mengamati sinyal plotting melalui TP-1 (lihat Gambar III-1), dan secara intuitif mahasiswa akan menentukan bahwa hasil perekaman sinyal adalah dalam keadaan normal atau terganggu.

b). Zooming sinyal di domain frekuensi

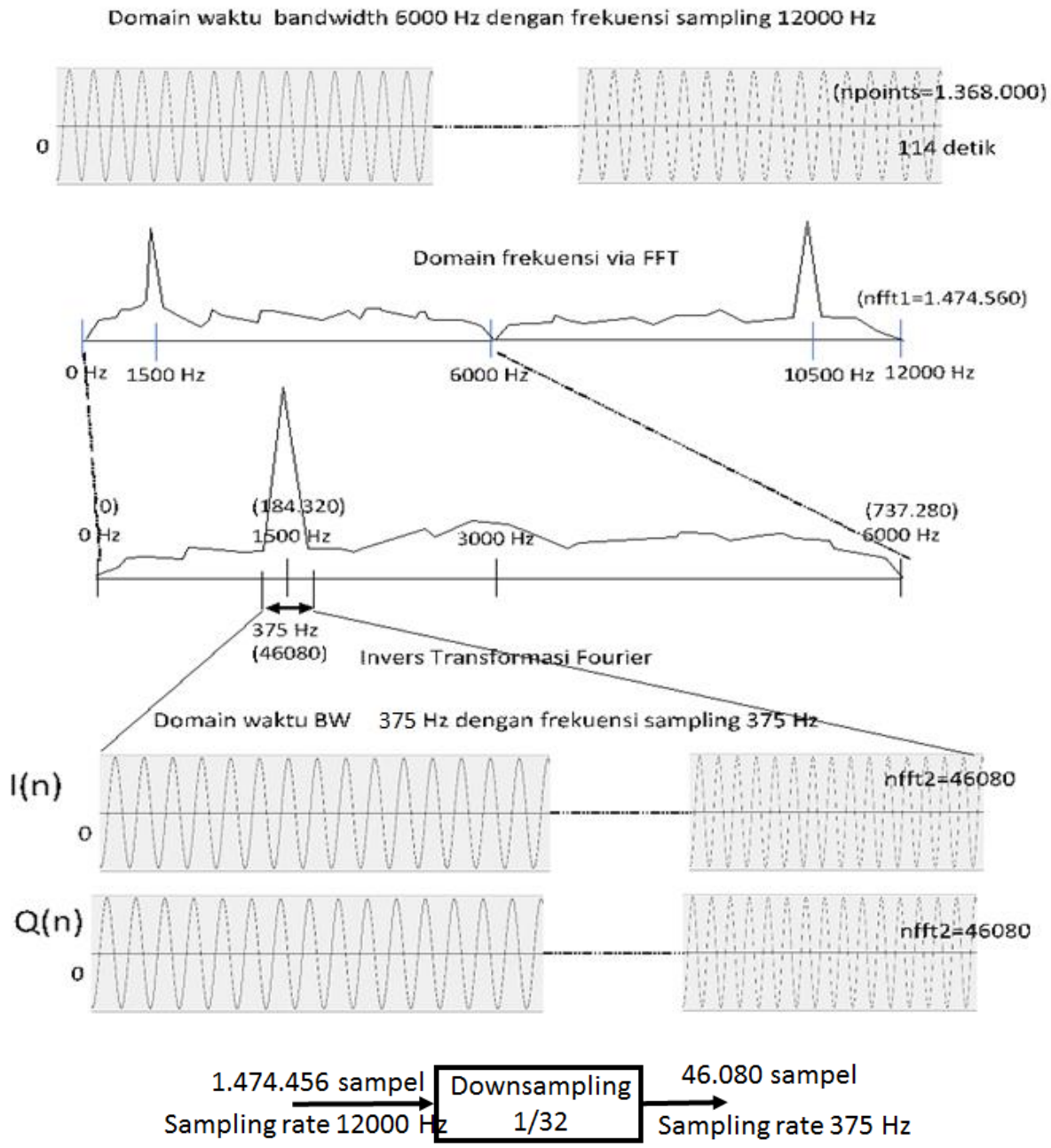
Sinyal WSPR yang diterima oleh sistem receiver kita ini memiliki level daya rendah dan menempati bandwidth yang sangat sempit (hanya 6 Hz). Cara seksama harus dilakukan untuk mendeteksi dan mendekode sinyal WSPR yang tertimbun oleh nois dan sangat banyak menerima rugi-rugi karena propagasi ionosfer jarak jauh. Pergeseran Doppler, drift frekuensi, kesalahan kalibrasi frekuensi, serta pergeseran pewaktu, merupakan kendala inheren dalam sistem telekomunikasi digital nirkabel, sehingga memerlukan frekuensi offset maupun maupun timing offset selama proses dekoding.

Pengondisian sinyal adalah pemrosesan sinyal untuk mempersiapkannya sinyal guna menghadapi tahap pemrosesan selanjutnya. Praktikum III bertujuan untuk mempersiapkan segmen sinyal AFSK (Audio Frequency Shift Keying) yang berpusat pada frekuensi 1500 Hz dengan frekuensi sampling 12 kHz, bandwidth 3000 Hz (sesuai dengan bandwidth saluran penerima HF), dan panjang 114 detik. Selanjutnya untuk meningkatkan akurasi saat pemrosesan di domain

frekuensi pada saat demodulasi FSK, maka diperlukan proses penurunan frekuensi sampling atau down-sampling sebesar 1/32 menjadi 375 Hz.

Program WSPR menggunakan domain frekuensi untuk proses downsampling sinyal AFSK. Pertama, 114 detik sinyal WSPR, dengan frekuensi sampling 12 kHz (yaitu, 12.000×114=1.368.000 sampel) ditransformasikan ke domain frekuensi dengan memanfaatkan fungsi dari library FFTW menjadi 2 × 1.474.560 poin komponen riil dan imajiner dalam domain frekuensi, dengan resolusi frekuensi 0,008138 Hz per poin. Jadi, untuk mendapatkan sinyal dengan bandwidth 375 Hz, kita membutuhkan 46080 poin (yakni, 375/0,008138). Hasilnya adalah 2 × 46080 poin (komponen riil dan imajiner) hasil windowing dengan pusat di titik 184.320 (yaitu, frekuensi 1500 Hz). Kemudian, sampel-sampel di domain frekeunsi hasil windowing 2 × 46080 ditransformasi kembali ke domain waktu dengan menggunakan fungsi library FFTW menjadi 2 × 46080 sampel, yang merupakan komponen sefase dan kuadratur dalam domain waktu. Dengan demikian sekarang

didapat sinyal IQ-AFSK dengan frekuensi sampling lebih rendah, yakni, 375 Hz. Ilustrasi proses pengolahan sinyal ini dapat dilihat pada Gambar IV-3 dibawah ini.



Gambar IV-3: Proses down-sampling melalui domain frekuensi

Sama dengan praktikum sebelumnya, mahasiswa harus melakukan Pengujian *white-box* untuk melacak proses ini dari awal program dengan melacak dan mengenali sintaksis demi sintaksis. Para siswa akan mengamati bentuk gelombang dengan memploting segmen sinyal (katakan 256 sampel) dengan menggunakan antarmuka pipa GNUPLOT. Para siswa dapat mengamati sinyal

plot melalui TP-2, dan secara intuitif mereka akan menentukan bahwa sinyal terlihat teratur atau dicurigai mengalami gangguan.

c). Membuat Sinyal WSPR Dummy

Rangkaian percobaan ini akan memanfaatkan sinyal asli yang telah ditangkap oleh radio HF dan disimpan dalam memori pemyimpan WSPR untuk selanjutnya akan dianalisis dan diekstrak untuk memperoleh kembali digit-digit message yang telah dikirim dari tempat yang sangat jauh itu. Namun demikian tetap diperlukan sinyal pembanding untuk meyakinkan bahwa sistem sudah berjalan sesuai dengan desain dan siap untuk melayani analisis sinyal yang sebenarnya.

Membangkitkan sinyal WSPR dummy dilakukan dengan cara menjalankan program "wsprdum" dengan memasukkan parameter message sesuai dengan prosedur dan protokolnya, sehingga dihasilkan file "simbol.txt" yang berisi deretan 162 simbol yang masing-masing simbol adalah satu dari empat kemungkinan simbol. Proses ini telah dijelaskan detil pada Bab I yang berisi penjelasan perihal sisi pemancar dari WSPR.

Selanjutnya adalah pembangkitan sinyal FSK-4 yang dilakukan dengan cara menjalankan program "fskmod". Fungsi program ini adalah membangkitkan sinyal sinusoidal dengan perubahan frekuensi sesuai simbol yang dibaca dari file "simbol.txt", untuk selanjutnya hasil pembangkitan sinyal FSK-4 ini disimpan dalam format WAV sebagai file suara dengan nama "simbol.WAV". Sinyal audio inilah yang kemudian diumpankan ke modulator transmiter menjadi gelombang radio yang akan dipancarkan via antena untuk merambat jauh melalui kanal propagasi HF (High Frequency). Blok diagram dari proses ini dapat dilihat pada Gambar I-12(b). Contoh terminal command untuk langkah 1 dan 2 dapat dilihat pada Gambar IV-5.

Isu menarik yang perlu digaris bawahi pada saat pembangkitan FSK adalah perihal diskontinuitas sinyal yang harus dihindari dengan cara penambah fasa yang besarnya sama dengan fase sampel terakhir dari sinyal simbol sebelumnya. WSPR tidak mengandalkan fasa sebagai tunggangan informasi, sehingga sinyal FSK yang dibangkitkan dapat diubah-ubah fasanya guna menghindari kondisi diskontinuiatas. Teori rinci perihal diskontinuitas tidak dibahas pada rangkaian kegiatan praktikum ini. Para mahasiswa dipersilahkan untuk merujuk kepada teori atau praktikum pengolahan sinyal dalam bab mengenai efek diskontinuitas.

Persamaan pembangkitan sinyal secara umum dapat ditulis sbb,

$$jika\ s = 0, x_s(n) = A \sin\left((\frac{2\pi}{12000} f_s)n\right)$$

$$jika\ s > 0, x_s(n) = A \sin\left(\left[\left((\frac{2\pi}{12000} f_s)n\right) + \varphi_s\right]\right)$$

Dimana,

A adalah amplitudo sinyal

f adalah frekuensi sinyal simbol

n adalah urutan sampel dimana n= 0,1,2,3,...,N-1

N = banyaknya sampel dalam 1 simbol = 8192.

s adalah urutan simbol dimana s= 0,1,2,3,...,S-1

S = banyaknya simbol dalam 1 paket data WSPR = 162

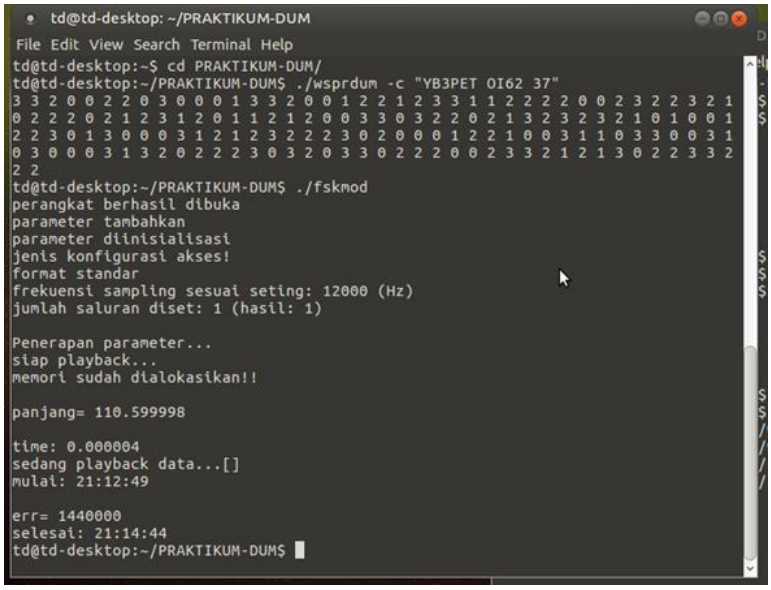
φ_s adalah sudut fasa yang besarnya adalah,

$$\varphi_s = (\frac{2\pi}{12000} f_{s-1})N$$

Gambar IV-4 menunjukkan gejala diskontinuitas dapat diatasi dengan menambahkan fase yang diambil dari fase sampel terakhir sinyal simbol sebelumnya.

Gambar IV-4: Efek diskontinyu dapat diatasi dengan menambahkan fase yang diambil dari fase dari sampel terakhir simbol didepannya.

IV.2 LANGKAH PERCOBAAN



Gambar IV-5: Contoh line command untuk menjalankan langkah 1 dan 2 guna membangkitkan sinyal WSPR dummy

1. Siapkan modul Raspberry Pi untuk dioperasikan.

2. Masuk ke direktori “PRAKTIKUM-DUM”
   %> cd PRAKTIKUM-DUM \r
   Jalankan program “dsamp” dengan memasang argumen “xxxxx.raw”. Program “dsamp” adalah program untuk downsampling file suara dari sampling frekuensi 48 kHz menjadi 12 kHz. Proses down-sampling dilaksanakan dengan metode desimasi di domain waktu. Hasil program ini adalah dua file suara, yakni “aliasing.raw” yang merupakan file suara hasil downsampling tapi tanpa melibatkan filter lowpass sebagai filter antialiasing. Hasil kedua adalah file suara “antialiasing.raw” yang merupakan hasil downsampling dengan menyisipkan filter lowpass sebagai filter anti aliasing. Selain dari pada itu program ini juga akan membandingkan besarnya SNR hasil down-sampling antara yang memakai anti aliasing filter (antialiasing.raw) dan tanpa aliasing filter (aliasing.raw).

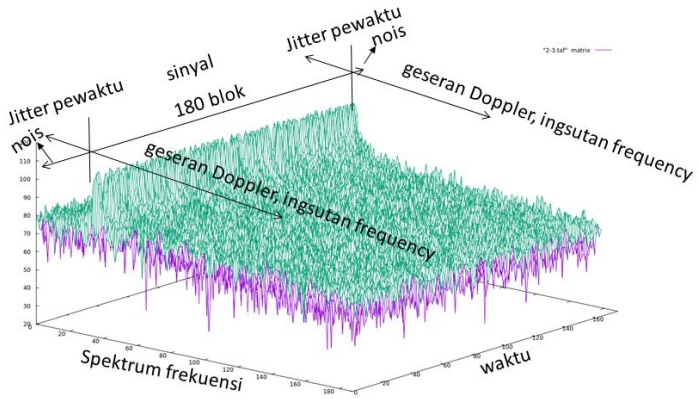
   %> dsamp xxxxx.raw \r
3. Jalankan program “wsprddum” dengan argumen data seperti dibawah ini
   %> wsprddum -cd “YB3PET OI62 37” \r
   Hasil dari program ini adalah file teks simbol.txt berisi 162 simbol (masing-masing 2 digit atau 4 kemungkinan). Kemudian file simbol.txt dipakai sebagai dasar pembangkitan sinyal WSPR dengan menjalankan program “fskmod”
   %> fskmod \r
   Selanjutnya akan didapat file suara dengan dengan nama file simbol.WAV. Saat ini kita sudah memiliki 3 file suara yang masing-masing didapat dari 3 cara berbeda, yakni: (1) file

hasil proses penerimaan dari sinyal WSPR yang dipancarkan dari suatu tempat di belahan bumi yang lain; (2)file hasil penerimaan dari sumber pemancar dari laboratorium sendiri dalam bentuk RAW, yang kemudian dilakukan downsampling dengan maupun tanpa filter anti-aliasing, dengan nama file aliasing.WAV dan antialiasing.WAV; (3)file hasil proses pembangkitan dari perangkat lunak untuk keperluan simulasi, dengan nama file simbol.WAV.

4. Jalankan program "wsprd" dengan argumen masing-masing file suara (ada 4 file sinyal).
%> wsprd xxxxxx.WAV
Masing file sinyal akan menghasilkan 3 jenis ploting, yakni:

- ploting spektrum frekuensi dari seluruh sinyal (sepanjang 110,6 detik) dengan BW 6 kHz,
- ploting suatu segmen sinyal,
- ploting spektrum frekuensi dari suatu simbol dengan BW 375 Hz.

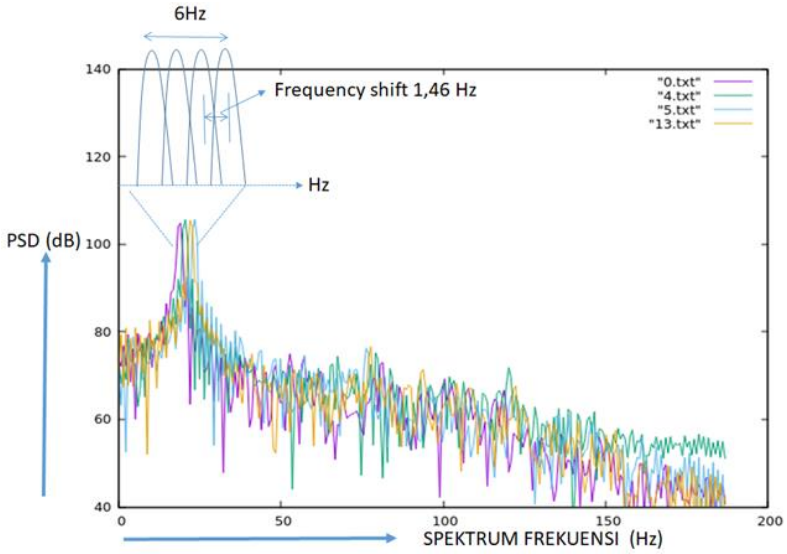
5. Catatan penting:



Gambar IV-6: Spektrum frekuensi seluruh segmen sinyal dilihat secara 3 dimensi.

Gambar IV-6 menunjukkan gambar 3-dimensi spektrum sinyal simbol WSPR. Jumlah simbol sebenarnya adalah 162 simbol blok, namun sebagai antisipasi terjadinya jitter akibat eror pewaktu, maka secara keseluruhan sistem merekam data lebih panjang yakni 180 simbol blok. Setiap blok simbol digambarkan dengan 1 lapis gunung spektrum yang independen terhadap gunung spektrum tetangganya. Deteksi puncak spektrum

dilaksanakan dalam proses yang kompleks, dengan memperhatikan efek dopler maupun efek ingsutan dari osilator sisi pemancar maupun penerima.



Gambar IV-7: Spektrum frekuensi dari 4 simbol yang berbeda, yang digambar secara 2 dimensi.

Gambar IV-7 adalah Spektrum frekuensi dari 4 simbol yang berbeda, yang digambar secara 2 dimensi. Gambar ini dibuat berdasarkan prosesing mandiri yang bukan menjadi bagian dari proses WSPR. Spektrum frekuensi dari Gambar IV-6 dan IV-7 ini dibuat setelah proses down-sampling sinyal dengan sampling 12 KHz menjadi sampling 375 Hz. Nampak bahwa beda frekuensi 1,46 Hz adalah sangat sempit dan berdekatan, sehingga perlu proses yang seksama agar dapat mendeteksi frekuensi puncak dari masing-masing gunung spektrum yang mewakili satu simbol tersebut.

```
  nfft2=46080; // Jumlah sampel setelah didown sampling dengan faktor 32
  nh2=nfft2/2;
  nfft1=nfft2*32;      //Akan didownsampling dengan faktor 32
  df=12000.0/nfft1;
  i0=1500.0/df+0.5;
  npoints=114*12000;//jumlah sampel hasil perekaman dgn sampling 12 KHz

  float *realin;       // deklarasi bilangan input real FFTW3
  fftwf_complex *fftin, *fftout;   // deklarasi bilangan kompleks FFTW3

// alokasi array input real
    realin=(float*) fftwf_malloc(sizeof(float)*nfft1);
// alokasi array output kompleks dari FFTW3
 fftout=(fftwf_complex*) fftwf_malloc(sizeof(fftwf_complex)*(nfft1/2+1));

persiapan FFTW3 dgn jumlah input nfft1, input real pada array realin, hasilnya
kompleks fftout
PLAN1 = fftwf_plan_dft_r2c_1d(nfft1, realin, fftout, PATIENCE);
```

Gambar IV-8: Fungsi FFTW yang dipakai pada proses down-sampling

```
// memasukkan input FFTW3 dgn faktor normalisasi 2^13 agar
harga fftout tidak terlalu besar

for (i=0; i<npoints; i++) {
      realin[i]=buf2_int16[i]/32768.0;
    }
for (i=npoints; i<(size_t)nfft1; i++) { //padding FFT
        realin[i]=0.0;
    }
free(buf2);
    free(buf2_int16);
    free(buf2_float);
    fftwf_execute(PLAN1);
    fftwf_free(realin);
```

Gambar IV-9: Fungsi FFTW yang dipakai pada proses down-sampling

```
fftin=(fftwf_complex*) fftwf_malloc(sizeof(fftwf_complex)*nfft2);

/* i0===1500Hz; -nh2----->i0----->nh2,   nfft2=2nh2  */
for (i=0; i<(size_t)nfft2; i++) {
     j=i0+i;
     if( i>(size_t)nh2 ) j=j-nfft2;
     fftin[i][0]=fftout[j][0];
     fftin[i][1]=fftout[j][1];
   }
fftwf_free(fftout);
fftout=(fftwf_complex*) fftwf_malloc(sizeof(fftwf_complex)*nfft2);
PLAN2 = fftwf_plan_dft_1d(nfft2, fftin, fftout, FFTW_BACKWARD, PATIENCE);
fftwf_execute(PLAN2);

for (i=0; i<(size_t)nfft2; i++) {
       idat[i]=fftout[i][0]/1000.0;       /* normalisasi dgn 1000 */
       qdat[i]=fftout[i][1]/1000.0;       /* normalisasi dgn 1000 */
   }
```

Gambar IV-10: Fungsi FFTW yang dipakai pada proses down-sampling

IV.3 TUGAS DAN LAPORAN PRAKTIKUM

1. Jelaskan maksud dari parameter/konstanta berikut ini: nfft1, nfft2, nh2, df, i0 dan npoints.
2. Tunjukkan fungsi bahasa C yang menyertakan parameter/konstanta yang tersebut pada no.1.
3. Bandingkan SNR dari data yang diperoleh tanpa filter antialiasing dan data dengan filter anti aliasing. Fenomena apakah yang terjadi sehingga mengganggu proses pengukuran SNR.
4. Laporan praktikum dikumpulkan sebelum Praktikum-4 dilaksanakan.