

AUDIO MOSNTER V1.2.0 IMPLEMENTASI AUDIO PROGRAMING UNTUK PEMEROSAN, ANALISIA DAN MANIPULASI DATA AUDIO DIGITAL

Aneu Yulianeu¹, Lucky Andriana Hartono²

1) Prodi Manajemen Informatika

STMIK DCI

Anjusu09@gmail.com

2) LPPM STMIK DCI

luckyar09@gmail.com

ABSTRAK

Perkembangan Programming Audio Processing semakin hari semakin berkembang cukup pesat untuk mengikuti kebutuhan dan kelengkapan teknologi perangkat audio yang perkembangannya saat ini sangat signifikan, bisa dilihat sekilas perkembangan dari mulai perangkat audio mono (channel tunggal) yang dulu digunakan dalam radio analog ataupun televisi sekitar awal tahun 1970 hingga sekarang teknologi Dolby Surround System yang sudah mampu mengusung audio multi channel hingga 10.2 dan teknologi environment Audio Effect Prosesing EAX™ yang mampu mengubah tatanan suara audio digital yang seolah tata suara tersebut berada lingkungan tertentu.

Ini merupakan sebuah bukti teknologi audio terus berkembang, seiring itu pula teknik pemrograman audio digital terus berkembang mengikuti perkembangan teknologi perangkat yang ada, namun kalangan pengembang pemrograman audio ini masih sedikit sehingga bisa dijadikan sebuah peluang bagi kalangan developer IT baik itu perusahaan maupun perorangan.

Audio Monster (Manipulation Of Sound System Engine Processing) adalah merupakan salah satu audio player yang memiliki fitur – fitur rekayasa teknologi audio mutakhir yang mana mampu memberikan kualitas tatanan audio yang berkelas seperti DSP (digital Sound Processing), Parametrik Equalizer 21 channel, Fader Simulation serta dynamic bass frequency. Selain itu Audio Monster memberikan kemudahan untuk dapat merubah setting equalizer channel point beserta dengan frequency bandwidth, serta fitur – fitur lainnya.

I. PENDAHULUAN

Beberapa waktu lalu ramai dibicarakan tentang beberapa produk antivirus buatan negeri sendiri, itu merupakan sebuah tanda bahwa perkembangan software buatan lokal (dalam negeri) mulai dipertimbangkan dan diperhitungkan oleh para pengguna komputer, walaupun masih bertaraf pemakai dalam negeri namun itu

menjadi suatu perkembangan yang signifikan dalam dunia IT dalam negeri yang menandai akan mulai adanya nilai kepercayaan terhadap kemampuan bangsa Indonesia dalam bidang IT yang selama ini masih terlindas oleh produk - produk luar yang memang tidak ditutupi akan kualitas dan kuantitasnya masih jauh diatas, namun dengan adanya hal tersebut diatas bisa

dijadikan sebagai titik tolak geliat produk IT buatan dalam negeri untuk nantinya mampu bersaing dengan produk - produk luar bahkan dengan produk dari developer kenamaan sekalipun.

II. LANDASAN TEORI

2.1 Audio Programming

2.1.1 Audio

Berdasarkan pakar multimedia yang bernama *Lu*, pada tahun 1999, mengatakan bahwa pengertian suara (*audio*) adalah sesuatu yang disebabkan perubahan tekanan udara yang menjangkau gendang telinga manusia. Sedangkan berdasarkan pakar yang lain, bernama *Andleigh*, pada tahun 1995, mengatakan jika frekuensi tekanan udara berada pada jarak 20 sampai 20.000 Hertz, telinga manusia mengidentifikasi sebagai suara. Frekuensi adalah siklus yang diselesaikan oleh sinyal dalam satu detik sedangkan *Hertz* adalah Unit Dasar Frekuensi yang juga disebut siklus per detik, jumlah siklus penuh yang diselesaikan dengan sinyal bolak - balik dalam satu detik.

Berkas audio atau suara memiliki beberapa format yang berbeda-beda tergantung dari penggunaan platformnya. Masing - masing format biasanya diikuti dengan perbedaan struktur berkas yang membentuk berkas audio tersebut.

2.1.2 Audio Digital

Reproduksi dari gelombang audio atau suara analog pada frekuensi sangat tinggi (*Frequency Sampling*) dan setiap *sample*-nya (*sample frequency*) disimpan dalam bentuk angka.

2.1.3 Audio Programming

Teknik pemrograman yang menitikberatkan pada pengolahan stream dari rekaman *sample frequency audio* sehingga dapat dimanipulasi sedemikian rupa untuk mendapatkan hasil frekuensi dan atau rekaan yang diinginkan namun mampu diadaptasi oleh ukuran frekuensi yang dapat ditangkap oleh telinga manusia serta dirancang untuk tidak terlalu membebani kinerja RAM dan *Processor*.

2.2 Audio Engine Library

Pustaka kode program untuk pengolahan data audio untuk membantu dan mempermudah melakukan *scripting code* tanpa harus memberikan signal – signal kode proses kepada *processor*(*IC Processing*) pada *sound card* karena sudah ditangani oleh *Engine Libray* baik secara *runtime* maupun *stream*, dan biasanya berkas ini bersifat dinamis (*dynamic Link Library*) agar tidak memberatkan proses dengan *auto free memory* saat tidak digunakan.

Salah satu contoh *audio engine library* yang sangat terkenal dalam sistem operasi windows adalah *directSound.dll* dan *directsound3D.dll* dalam group *directX*.

2.2.1 Bass Audio Engine Libray

Bass Engine Libray adalah salah satu produk dari *developer un4seen* dengan varian versi lisensi yang salah satunya digunakan oleh penulis yaitu *lisensi Free-NonComersial-User* yang memudahkan *programmer* untuk mempelajari dan mengembangkan audio programming tanpa harus membayar mahal untuk membeli lisensi tersebut, namun dengan syarat produk IT yang menggunakan *Enggine Library* tersebut bersifat *GNU Lisence*.

Salah satu produk dari developer ternama *Pandora.Tv* yaitu *The KMPlayer* yang cukup terkenal dikalangan para pengguna komputer mempercayakan *Bass Library* sebagai *engine audio* mereka.

2.3 Digital Sound Processing (DSP)

Digital Signal Processing (Pengolahan Sinyal Digital) adalah segala bentuk manipulasi yang dilakukan pada Sinyal Audio atau video sewaktu dalam bentuk Digital.

Istilah DSP memperoleh reputasi yang tidak menguntungkan, sewaktu pada awal - awalnya, sinyal ini terkait dengan efek *reverberasi* (pemantulan) buatan (aula, stadion, dsb.) yang bisa ditambahkan sewaktu playback (pemutaran).

III. ANALISA MASALAH

Konsep identifikasi adalah sebuah usaha untuk mengetahui perilaku sistem atau proses yang kehendaki. Hal tersebut dilakukan dengan membuat sebuah pendekatan dari sebuah proses tertentu. Pendekatan ini dinilai memudahkan perancangan ataupun analisis, sehingga dengan pemodelan visualisasi dapat dilakukan monitoring perilaku sistem.

Terdapat banyak pendekatan yang dapat dilakukan dalam menyusun sebuah model. Pendekatan tersebut muncul dikarenakan kebutuhan akan berbagai macam karakteristik dan kebutuhan rencana yang ada di dunia nyata.

3.2 Analisis Sistem

3.2.1 Audio Player

Audio player adalah program komputer untuk memutar ulang audio digital, termasuk cakram optik seperti CD, SACDs, DVD -Audio, HDCD, file audio dan

audio streaming tergantung format dukungan dari audio player tersebut.

Fungsi utama dari audio player adalah harus mampu melakukan *Encode* dan *Decode (Codec)* berkas media yang didukungnya sehingga dapat memutar berkas media tersebut.

Encode dan *Encode (codec)* adalah penerapakan suatu algoritma untuk kompres dan decompres data audio digital sesuai dengan format file audio yang diberikan atau streaming format media audio. Tujuan dari algoritma ini adalah untuk mewakili sinyal tinggi kesetiaan audio dengan jumlah minimum bit sementara tetap mempertahankan kualitas. Hal ini secara efektif dapat mengurangi ruang penyimpanan dan bandwidth yang dibutuhkan untuk transmisi file audio yang tersimpan.

3.2.2 Filter Equalizer

Filter equalizer terdiri atas filter yang dapat disetel yang dapat mengubah respons Frekuensi sistem audio. equalizer dapat menggantikan *aberasi* atau kelainan respons frekuensi dalam penguat suara, kombinasi penguat suara dan juga untuk menyesuaikan keseimbangan nada rekaman.

Urutan frekuensi dalam audio sistem terdiri dari rentan frekuensi standar yang mampu didengar manusia yaitu dari kisaran 20 Hz (Hertz) hingga 20 Khz (Kilo Hertz), umumnya ada beberapa penerapan ubahan sesuai urutan besaran hertz dalam frekuensi yang mana nilai kelainan dari respon audio tersebut memiliki kepekaan tersendiri terhadap pendengaran manusia. Pada dasarnya terbagi menjadi tiga (3) besaran kelainan dari urutan frekuensi.

1. frekuensi rendah yang berada pada kisaran antara 20hz hingga 600hz yang

memiliki karakteristik pada pendengaran manusia adalah nada *bass* atau dentuman gema.

- frekuensi tengah atau sedang yang berada pada kisaran antara 600hz hingga 8Khz yang memiliki karakteristik pada pendengaran manusia adalah nada *middle* atau vokal suara.
- Frekuensi tinggi yang berada pada kisaran antara 6Khz hingga 20Khz yang memiliki karakteristik pada pendengaran manusia adalah nada *treble*.

Sebagai contoh equalizer dengan tiga channel besaran frekuensi yang diubah adalah 80hz untuk frekuensi rendah, 1Khz untuk frekuensi sedang dan 14Khz untuk frekuensi tinggi.

Dari tiap besaran frekuensi yang di filter oleh equalizer disebut sebagai channel equalizer. Semakin banyak channel equalizer yang didukung maka semakin banyak pula besaran frekuensi hertz yang dapat diatur namun akan sedikit memakan waktu lama untuk menyelaraskan dengan konfigurasi yang diinginkan jika channel equalizer terlalu banyak dan pengguna yang awam.

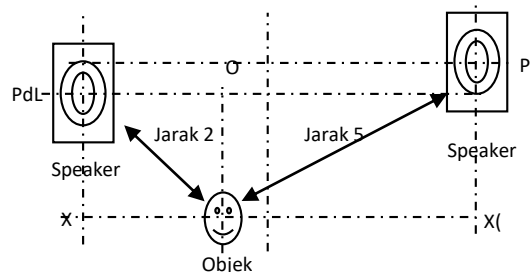
3.2.3 Distance Effect

Hal ini mungkin masih terasa asing, dan mungkin belum pernah dikembangkan oleh para developer audio.

Distance Effect adalah manipulasi keluaran suara audio dari perhitungan jarak speaker dengan pendengar untuk menghasilkan suara yang seimbang walaupun posisi pendengar dengan jarak speaker kanan dan kiri berbeda (*untuk konsep speaker Front Rear 2 Channel*).

Teknik ini mengkombinasikan antara Pan atau balance stereo dari suara dengan volume suara untuk mendapatkan kordinat

cross tengah untuk disesuaikan dengan posisi pendengar tanpa si pendengar perlu berganti posisi untuk mendapatkan suara yang seimbang antara speaker kanan dan speaker kiri.



Gambar 1 Simulasi Distance Effect

Dari gambar diatas bisa kita ambil seting konfigurasi sebagai berikut :

$$\text{Volume Kiri} = X(L) - \text{PdF}$$

$$\text{Pan Kiri} = ((X(L) + X(R)) \text{ div } 2) - O$$

$$\text{Volume Kanan} = X(R) - \text{PdR}$$

$$\text{Pan Kanan} = ((X(L) + X(R)) \text{ div } 2) + O$$

3.2.4 Sample Rate

Sample Rate adalah siklus kecepatan pemutaran rata – rata dalam perbandingan antara rasio tempo (*time playing*) dan pitch (*key tone*) yang disinkronisasikan per-detik, sample rate secara standar (*default*) memiliki rasio 44100hz per-detik, semakin tinggi rasio tersebut maka kecepatan pemutaran dan key tone akan semakin tinggi karena jumlah frekuensi akan semakin banyak dalam stage stream dalam satu detiknya dan akan mempengaruhi nilai frekuensi dasar sehingga tone yang dihasilkan akan semakin tinggi menyelaraskan dengan tingkat perputaran stream stage per-detiknya.

3.2.5 Bass Subwoofer Booster

Sebutan *Subwoofer* disini sebetulnya hanya sebatas kiasan saja, secara harfiah Subwoofer adalah bagian dari salah satu jenis speaker dari pengembangan speaker jenis woofer yang berfungsi sebagai penghantar daya dari frekuensi yang paling rendah dibawah 60hz dan menghasilkan suara bass yang halus dan menggema, sedangkan jenis woofer itu sendiri biasanya mampu menghantarkan daya dari frekuensi antara 80hz hingga 500hz yang menghasilkan suara bass yang menghentak dan keras.

Adapun jenis – jenis speaker dibagi beberapa macam, diantaranya adalah :

1. *Speaker Woofer*, untuk penghantar daya dari frekuensi kisaran 80hz hingga 500hz.
2. *Speaker SubWoofer*, untuk penghantar daya dari frekuensi kisaran 20hz hingga 60hz.
3. *Speaker Middle*, untuk menghantar daya dari frekuensi kisaran 600hz hingga 8Khz.
4. *Tweeter*, untuk menghantar daya dari frekuensi kisaran 8Khz hingga 20Khz.

Bass Subwoofer Booster disini adalah teknik mengubah nilai frekuensi kisaran 80hz hingga 500hz menjadi lebih rendah dari aslinya sehingga hentakan bass pada frekuensi tersebut diganti dengan suara bass yang halus.

3.2.6 Dynamic Frequency

Dinamic Frequency adalah teknik mensinkronisasikan keluaran frekuensi dengan maksimal frekuensi yang mampu ditahan oleh tegangan dalam speaker secara umum sehingga akan mengurangi over frequency dan akan terdengar ke telinga manusia suara yang dinamis dan

harmonis tanpa ada *overloud* bass dan atau pecah suara treble.

Penyebab utama *Over Frequency* adalah dimana saat frekuensi terendah dan frekuensi tertinggi mendapat gain desibel tertinggi secara bersamaan sehingga frekuensi yang dikirimkan kepada processor soundcard menjadi kacau yang akan menghasilkan sinyal daya yang terlalu tinggi sehingga soundcard akan memberikan arus daya terlalu tinggi juga terhadap speaker yang melebihi arus daya maksimal sehingga speaker akan mengeluarkan suara gemuruh dan atau feedback yang dapat merusak spool atau lilitan tembaga dalam speaker sebagai penghantar arus yang menjadi medan magnetnya.

3.2.7 Digital Sound Processing (DSP)

Digital Sound Processing atau disingkat DSP secara sistem adalah rekayasa dari stream digital pada saat sebelum pemutaran media dilakukan. Stream dimanupulasi sedemikian rupa sehingga mampu mengadopsi tata suara lingkungan, tempat, ruangan dan lain halnya dengan cara mengambil sampel gema pantulan dan pembekasan suara pada area atau wilayah tertentu untuk dapat menentukan secara tepat perhitungan kecepatan pantulan dan gema yang didapat untuk dapat di terapkan dalam sistem DSP yang sekarang kita kenal dengan teknologi *Environment Effect*.

Konsep dari teori *DSP Environment* ini sebetulnya cukup sederhana, yaitu teori perulangan dan penurunan yang dinamis dari suara hingga dalam imajinasi manusia akan terdengar seperti gema.

DPS Environment ini adalah kombinasi dari kecepatan perulangan suara, jumlah

perulangan suara, dan pengurangan daya suara(*volume*).

Salah satu contoh kita ambil environment gunung(*mountain*), misalnya hasil analisis dengan suara didapat dari hasil sebagai berikut,

Jumlah perulangan : 6 kali

Interval perulangan : 2 sec

Penurunan daya suara : 2dB(*desibel*) / sec(*dB* atau *Desibel* adalah *Ukuran Logaritma relatif Voltase, arus atau Daya*)

Dari hasil contoh analisis diatas maka kita bisa terapkan kedalam sistem DSP dengan nama "*Mountain Environment*".

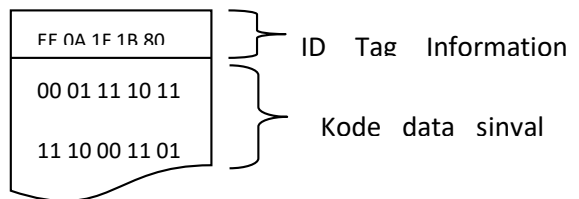
Namun ada beberapa teknik *DSP Environment* diatas banyak yang sudah dikombinasikan dengan teknik – teknik DSP lainnya seperti *Flanger* atau teknik siklus perubahan frekuensi tengah dengan penambahan dan pengurangan yang dinamis, *Chorus* atau teknik duplikasi suara dengan delay cepat sehingga menghasilkan suara semi *distorsi*(gemuruh) dan duplikasi suara dan lain sebagainya untuk mendapatkan hasil suara environment yang menyerupai area yang dikehendaki.

3.2.8 Identity Tag Information

Identity Tag Information atau yang sering kita kenal dengan *ID Tag* dalam beberapa berkas media adalah identitas dan informasi seputar berkas media tersebut, seperti : *Judul, Artis, Album, Track, Titel, Genre, Tahun* dan lain sebagainya.

ID tag ini tersimpan pada header berkas media yang akan terbaca saat kita melakukan *encode* berkas media, dan mesin *encoder* atau *engine library encoder* secara otomatis akan memisahkannya dari code data sinyal frekuensi audio karena dalam beberapa berkas media ada kode pembatas antara code data sinyal dan data kode ID tag

sehingga *encoder* akan mudah memisahkannya.



Gambar 2

Gambaran Header di Berkas Media

3.3 Analisis Kebutuhan Perangkat Keras

Spesifikasi kebutuhan minimal yang direkomendasikan adalah :

1. Processor Intel® DualCore™ 1,7Ghz atau AMD® Athlon™ X64 2.0ghz atau Processor dengan kecepatan proses yang setara.
2. Memori Proses (RAM) 1024Mb DDR2 PC-2700.
3. Ruang penyimpanan bebas dalam Harddisk lebih dari 50Mb dan memiliki Rate Per-Minutes (RPM) diatas 5400x
4. Soundcard 24Bit AC 97® atau Realtek® HD™ atau soundcard setara lainnya yang compatible
5. VGA dengan VRam 64mb
6. Monitor dengan resolusi 1024 x 800 keatas

IV. PERANCANGAN SISTEM

Setelah proses analisis maka selanjutnya adalah proses perancangan sistem berdasarkan dari hasil analisis yang bertujuan untuk mendeterminasi proses-proses serta data – data yang dibutuhkan oleh aplikasi.

Rancangan fasilitas yang dibuat :

1. Fasilitas Player Control

Mencakup kontrol pemutaran media (*play, pause, stop, back next*), pola pemutaran serta shortcut untuk seluruh fasilitas yang ada dalam aplikasi.

2. Fasilitas Media Libray Playlist

Fasilitas untuk memuat daftar media playlist dan menampilkan ID3 Tag informasi serta fasilitas untuk menyimpan(*save*) dan membuka(*load*) berkas playlist.

3. Fasilitas Equalizer 21 Channel

Fasilitas untuk mengubah gain per-channel frekuensi equalizer sebanyak 21 channel equalizer berserta fasilitas untuk mengubah besaran gain maksimal.

4. Fasilitas DSP Environment

Fasilitas untuk men-set atau mengkonfigurasi environment sesuai keinginan serta fasilitas DSP manual untuk berkreasi efek dari DSP yang telah disediakan.

5. Fasilitas Sample Rate Control

Fasilitas untuk merubah sample rate, pitch tone dan tempo pemutaran media.

6. Fasilitas Distance Effect

Fasilitas untuk mengatur Efek jarak suara dari objek terhadap speaker kanan dan kiri beserta fasilitas test distance effect.

7. Fasilitas Plugin Visualization

Fasilitas untuk menampilkan visualisasi media dari plugins baik itu secara Default windowed(*sekala kecil dalam form player*) maupun fullscreen(*seluruh layar*).

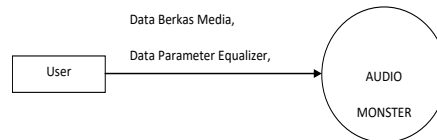
8. Fasilitas Tool Preferences

Fasilitas – fasilitas tambahan pendukung lainnya, seperti Internet Radio, pengaturan – pengaturan aplikasi dan lain – lain.

9. Fasilitas Dukungan Tipe Media

Memiliki Fasilitas dukungan untuk tipe media : *mp1, mp2, mp3, ogg, aiff, wav, mod, mtm, s3m, xm, it, mo3* dan *wma*.

4.1.1 Perancangan Diagram Alir Data Diagram Konteks



Gambar 4.1 Diagram Konteks

4.2 Perancangan Algoritma

Prosedur PlaybackMedia;

Kamus Data Global

```

Chan : Dword;
Floatable : Dword;
MediaSource : Strings;
SelectSource : TOpenDialog;
DecoderDX = Record
Channel : PAsniChar;
Type : TBassDecoderType; //0 auto
detectStat :
    TFI agState; //
[FSAuto,SStream,FSmusic]
End Record;
player : TBassDll;
{stdcall{$ELSE}cdecl{$ENDIF};external
bassdll;}
  
```

Implementasi

Begin

```

    If player.getstate='play' then
        Begin
            Player.channelstop(chan);
            Player.streamfree(chan);
        End;
        SelectSource ←
Topendialog.create(self);
If not selectSource.execute then exit
        Mediasource ←
selectsource.filename;
Player.getInfo(mediasource);
If not player.bit_sample > $16 then
    floatable← null //cek bitrate
  
```

```

Else Floatable ←
player_sample_float;
decoderDX.channel ←
PAnsiChar(mediasource);
decoderDX.type ← 0;
decoderDX.stat ← FSAuto;
Chan ←
player_StreamCreateFile(false,
PChar(mediasource),0,0,
Floatable or player_sample_float,
decoderDX);
player.play;
End;

```

Prosedur Equalizer;**Kamus Data Global**

```

Eqpeak = Record
Gain : integer;
Bandwidth : float;
Channel : TBassChannel;
Band : integer;
Center : integer;
End record;
Eqfx :Dword;
Chan : Dword;

```

Implementasi**Begin**

```

player_ChannelRemoveFX(chan, eqef);
eqfx ← player_ChannelSetFX(chan, dx8_
fx_eq, 1);
Eqpeak.Bandwidth ← 2.5; //0, 0.1 ..... 9.0
Eqpeak.Gain ← 0;
Eqpeak.Channel ← all_channel;
//set parameter frequency channel 1
Eqpeak.Band ← 0;
Eqpeak.Center ← 40;
player_SetParameters(eqfx, @eqpeak);
//set parameter frequency channel 2
Eqpeak.Band ← 1;
Eqpeak.Center ← 60;
player_SetParameters(eqfx, @eqpeak);
.

```

```

.
.
//set parameter frequency 21
Eqpeak.Band ← 20;
Eqpeak.Center ← 16000;
player_SetParameters(eqfx, @eqpeak);
player.loadeqgain(0, 100);
player.loadeqgain(1, 85);
.
.
.
Player.loadeqgain(20, 100);
Player.eq_activate(true, eqfx, @eqpeak);
End;

```

Prosedur DSP_Environment;**Kamus Data Global**

```

Dsp_set = Record
repeat : integer;
delay : float;
downrate : float;
End record;
Dsp_env : TBassDSP_fx;

```

Implementasi**Begin**

```

Player.Channel_free_dsp(Dsp_env);
Dsp_set.repeat ← 10;
Dsp_set.Delay ← 14000; //millisecond
Dsp_set.Downrate ←
(Dsp_set.Delay div Dsp_set.repeat);
Dsp_env ←
Player.dsp_set_param (bassdsp_fx,
@dsp_set);

```

```

If player.getstate = 'play' then
player.createdsp(dsp_env);

```

End;**Prosedur Distance_Effec_test;****Kamus Data Global**

```

LVol : integer;
RVol : Integer;
Pan : Integer;

```

Implementasi**Begin**

```

Lvol ← player.getvolume(L);
Rvol ← player.getvolume(R);
pan ← player.getvolume_balance;
if Lvol <= player.max_volume then
    OnsliderChange(Lvol +
        Slider.value)
Else if
    OnsliderChange(Lvol -
        Slider.value);
if Rvol <= player.max_volume
    then OnsliderChange(Rvol +
        Slider.value);
Else if
    OnsliderChange(Rvol - Slider.value);

```

End;**Prosedur Sample_Rate;****Kamus Data Global**

```

sample = Record
    Streamtone : integer;
    Streamrate : integer;
    Streamtempo : integer;

```

End Record;

```

streamresample : TbaseStreamSample;
trackbar_sample,          trackbar_pitch,
Trackbar_tempo : TTrackbar;

```

Implementasi**Begin**

```

sample ← player.Getinfo(sample_rate);
//sample rate
sample.Streamtone ←
trackbar_sample.value; sample.Streamrate
←
trackbar_sample.value;
sample.Streamtempo ←
trackbar_sample.value;player.setsample(st
reamresample, @sample);
//pitch tone
sample.Streamtone ←
trackbar_pitch.value; sample.Streamrate
←
trackbar_sample.value;

```

```

sample.Streamtempo ←
trackbar_sample.value;player.setsample(st
reamresample, @sample);
//pitch tempo
sample.Streamtone ←
trackbar_sample.value; sample.Streamrate
←
trackbar_sample.value;
sample.Streamtempo ←
trackbar_tempo.value;player.setsample(str
eamresample, @sample);
End;

```

Prosedur IDv3_tag;**Kamus Data Global**

```

Idtag = array [1..8] of string;
ITitle, lalbum, lartist, lgenre, lyear,
ltrack : TLabel;

```

IDv3 = Record

```

Title : string;
Album : string;
Artist : string;
Genre : string;
Year : string;
Track : string;

```

End Record;

```

Ver : Tid3type; //id3v1, id3v2 ...

```

Implementasi**Begin**

```

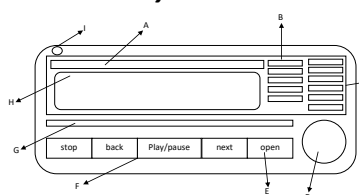
Player.getStreamInfo(ver,@ idv3);
Ltitle.caption ← idv3.title;

```

```

Ltrack.caption ← idv3.track;

```

End;**4.3 Perancangan User Interface****1. Form Player**

Gambar 4.7 Form Player

Keterangan :

- A. informasi nama berkas yang sedang dimainkan.
- B. Pilihan – pilhan menu untuk metoda pemutaran meliputi *Shuffle, Repeat All, Repeat Track, Visual Plugin, Balance*.
- C. Pilihan – pilihan menu untuk membuka form *equalizer, dsp environment, distance effect, media library* dan *sample rate*.
- D. Pengaturan volume suara
- E. Tombol untuk membuka berkas media yang nantinya akan disimpan dalam form *media library*.
- F. Tombol –Tombol *player control* standar meliputi *stop, back, next, play* dan *pause*.
- G. Informasi posisi secara visual dan sekaligus sebagai *rewind* atau *forward* posisi pemutaran media.
- H. Visualisasi dari plugins.
- I. Menu PopUp untuk Preferences dan pilihan – pilihan aplikasi lainnya.

V. IMPLEMENTASI

Setelah melakukan analisis dan perancangan terhadap aplikasi *Audio Monster*, tahapan selanjutnya adalah implementasi dan pengujian. Pada tahapan implementasi terdapat dua cakupan yaitu Implementasi Pemrograman dan spesifikasi kebutuhan sistem yang meliputi perangkat keras (hardware) dan perangkat lunak (*software*), dan hal-hal yang berhubungan dengan pengujian aplikasi.

5.1 Kebutuhan Perangkat Lunak

Untuk keperluan implementasi perangkat lunak penulis menggunakan beberapa tool dan library pendukung, diantaranya adalah :

Aplikasi *Borland Delphi 6.0 Personal GNU-Edition*

Aplikasi *Dynamic Skin Maker 12.1*

Aplikasi *Adobe Photoshop 8.0 CS*

Aplikasi *CDS Builder*

Library Audio Bass.dll

5.2 Kebutuhan Perangkat Keras

Pada saat implementasi perangkat lunak penulis menggunakan spesifikasi perangkat keras komputer dalam *platform NoteBook* sebagai berikut :

- *Processor AMD Athlon 2,1 Ghz*
- Memori jenis *DDR3* berkapasitas 4 Gb
- *Harddisk* berkecepatan *7200 Rate Per-Minutes (RPM)* dengan kapasitas penyimpanan 320 Gb.
- *Virual Graphic Adapter(VGA)* jenis *ATI Radeon 4220 HD* dengan *processor* kecepatan 533hz dan berkapasitas *VRam 512Mb*.
- *Sound Card 24bit RealTek HD*
- *LCD LED 14.4 wide*

VII. DAFTAR PUSTAKA

Fathansyah, Ir. *Basis Data*. Bandung : Informatika. 1999.

Pranata, Andy. *Pemrograman Borland Delphi 6*. Edisi 4. Yogyakarta : Andi Offset, 2002.

wikipedia (2011). *Audio Programmimg*. [online]. Tersedia :

<http://us.wikipedia.org/wiki/audioprograming>. [25 Desember 2011].

un4seen (2012). [online]

<http://un4seen.com/tutorial/basslib>

Yulianeu, Aneu. 2013. *Generate Report Critical Data In BRI Sariwangi parsing Teachique Using*. Jurnal Jutekin Vol.1 No.1 2017 LPPM STMIK DCI.