Perangkat Lunak Tuning Gitar dengan Menggunakan Karplus Strong Algorithm

Tanda Selamat[1], Kevin Angkasa[2]
STMIK IBBI
Jl. Sei Deli No. 18 Medan, telp.061-4567111, fax. 061-4527548
Email : tandaselamat@gmail.com

Abstrak

Tuning merupakan proses penyetelan senar pada alat-alat musik petik seperti gitar. Alat musik tersebut apabila sering dimainkan maka setelannya akan dapat berubah. Untuk itu biasanya dilakukan proses tuning untuk mendapatkan setelan yang pas sehingga ketika dimainkan nada-nadanya yang dihasilkan menjadi standar. Selain dengan cara mendengar lalu menyetel dapat pula dipergunakan garpu tala dalam proses tuning. Teknik tuning juga harus disesuaikan karena biasanya terdapat beberapa jenis tuning untuk alat-alat musik seperti ini. Implementasi pada tulisan ini adalah suatu program tuning untuk alat gitar musik dengan pengaturan tuning untuk 7 jenis beserta dengan 36 pilihan jenis tuning. Selain itu program ini juga menyediakan input tuning secara customize untuk alat musik petik lainnya.

Kata kunci : pengaturan, alat musik, tuning

Abstract

Tuning is the process of tuning the strings on stringed musical instruments like guitar. This instrument is often played when it will be able to change its settings. For the tuning process is usually done to get the settings right so that when played the notes produced a standard. Than by hearing and set the tuning fork can also be used in the tuning process. Tuning techniques should also be adjusted because there is usually some kind of tuning to musical instruments like this. The implementation in this paper is a guitar tuning program for musical instrument tuning arrangements for 7 types along with 36 choices of tuning. In addition the program also provides customized tuning input for other stringed instruments.

Keywords: *arrangement, musical instruments, tuning*

1. Pendahuluan

Berdasarkan kamus *webster, MUSIC is the art of sound or the meaningful organization of sounds* yang berarti musik merupakan seni suara atau organisasi suara yang memiliki arti. *Sound* (suara) dihasilkan dari getaran, baik udara maupun benda-benda padat. Ketika getaran itu bersifat tidak teratur, maka suara itu adalah *noise*; ketika getaran tersebut teratur, maka suara itu disebut *tone* atau *nada*. Musik tergantung dari nada, tidak termasuk *noise* (seperti bunyi simbal, tabrakan, piring pecah, dan lain-lain). Getaran yang pelan akan menghasilkan nada dan bunyi yang rendah (*low*), getaran yang cepat akan menghasilkan suara yang lebih tinggi (*high*). Pada prakteknya suara musik berkisar antara 40-4000 getaran per detik (*hertz*). Angka yang eksak (frekuensi) dari getaran akan menghasilkan bunyi yang sering disebut *pitch*. Pitch digunakan sebagai standar tinggi rendah dari sebuah *tone* atau nada.

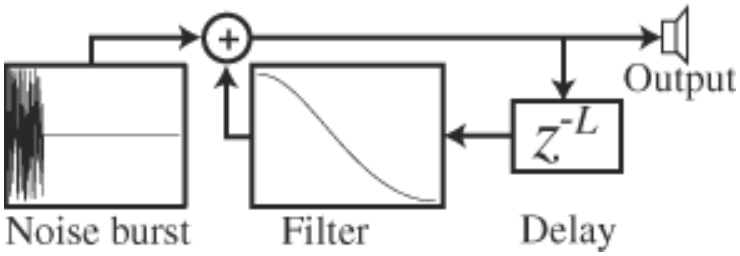
Musik melibatkan interaksi dari tiga unsur penting, yaitu:

1. *Rhythm* : perubahan teratur antara panjang dan pendek, aksen dan non aksen suara musik. Rhythm dapat dihasilkan oleh segala macam bunyi yang teratur, tapi sekarang dikhususkan pada alat-alat musik yang menghasilkan pola-pola *rhythm* seperti drum.
2. *Melody* : hasil dari bunyi bermacam-macam *pitch*.
3. *Harmony*: hasil dari beberapa *pitch* yang berbeda yang dibunyikan secara simultan.[1]

Alat musik yang menggunakan senar seperti piano dan gitar apabila sering dipakai maka pada suatu

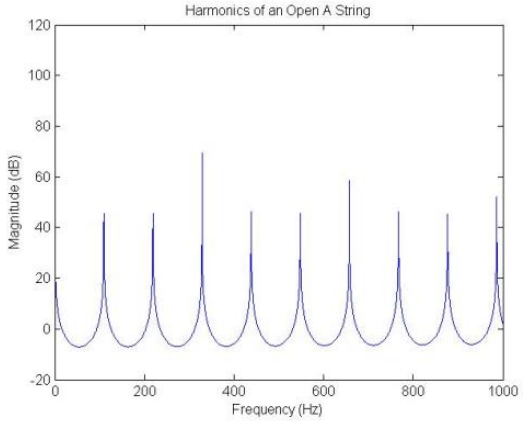
waktu harus dilakukan proses kalibrasi kembali atau dikenal dengan istilah *stem* atau *tuning*. Untuk piano harus menggunakan suatu alat yang disebut garpu tala. Khusus untuk alat musik seperti gitar dapat langsung diset melalui kuping senarnya. Penyetelan pada gitar tidak memerlukan suatu alat khusus hanya mengandalkan keahlian pemain dalam menyetel senar hingga mendapatkan posisi nada yang diinginkan pada gitar tersebut.

Salah satu solusi di atas adalah membuat suatu perangkat lunak untuk proses *tuning* dimana program dirancang untuk menghasilkan suatu nada tertentu dari keenam senar gitar melalui *output wave*. Untuk menghasilkan suara yang menyerupai suara gitar maka diperlukan suatu teknik atau algoritma khusus yang dapat mensimulasikan suara gitar. Algoritma yang dimaksud tersebut adalah *Karplus Strong Algorithm*. *Karplus Strong Algorithm* merupakan algoritam sintesis *wavetable* yang melakukan modifikasi sendiri. Cara kerja metode dari sintesis model fisikal ini adalah melakukan *loop* terhadap *waveform* pendek melalui suatu *delay* pada *filter* untuk mensimulasikan suara *string* yang dipetik seperti pada alat musik perkusi. Hasil dari algoritma ini adalah suatu bentuk *wave* yang berbunyi seperti suara gitar yang dipetik untuk nada tertentu.
Algoritma *Karplus Strong* dikembangkan oleh *Alexander Strong* dan dianalisis oleh *Kevin Karplus* sebagai suatu model untuk instrument musik pukul atau dipetik. Algoritma ini mensimulasikan impak yang bernada tajam melalui signal pita lebar seperti dalam bentuk kumpulan *noise*. Sinyal tersebut diumpan balik melalui suatu *delay line* dimana panjangnya bergantung pada frekuensi dari not yang diinginkan. Sinyal tertunda tersebut dikirimkan melalui suatu *filter lowpass* untuk menghaluskan semua frekuensi lainnya kecuali frekuensi dari nada yang inginkan, seperti ditunjukkan pada Gambar 1.[5]



Gambar 1 Diagram yang Merepresentasikan *Noise* yang Ditunda

Konsep dibalik dari algoritma Karplus Strong adalah memodelkan bentuk frekuensi not suatu alat musik petik dengan *noise* yang diistilahkan sebagai *white noise* yang mempunyai energi yang sama dalam semua frekuensi. Karena parameter lain rongga instrumen, bahan instrumen, dan berbagai, hanya frekuensi yang diberikan dan harmoniknya yang akan beresonansi, maka ini dapat disimulasikan oleh rekursif membentuk sinyal keluaran. Dengan pencocokan panjang waktu tunda untuk sesuai dengan frekuensi catatan yang diinginkan, *output* akhirnya akan suara pada frekuensi yang dipilih diberi waktu singkat. *Loop* umpan balik hanya memperkuat frekuensi fundamental dan harmoniknya. Teknik ini kadang-kadang disebut sebagai *filter* karena bentuk karakteristik *output* seperti dijelaskan pada Gambar 2.



Gambar 2 *Spektrum* dari *String* Petik

Secara singkat cara kerja dari algoritma *Karplus Strong* adalah sebagai berikut:

1. Pertama dimulai dengan membentuk *buffer* yang berisikan penuh nilai-nilai acak. Kebisingan. *Buffer* adalah beberapa memori komputer (RAM) dimana kita bisa menyimpan banyak nilai. Angka-angka dalam *buffer* ini merupakan energi awal yang ditransfer ke *string* oleh memetik.
2. Untuk menghasilkan gelombang, kita mulai membaca *buffer*, dan menggunakan nilai-nilai di dalamnya sebagai nilai-nilai sampel. Jika terus membaca *buffer* berulang-ulang, apa yang akan didapatkan akan menjadi nilai kompleks, bernada gelombang. Ini akan menjadi kompleks karena mulai keluar dengan *noise*, tetapi karena nada akan diulangi dengan angka yang sama dan angka acak lagi dan lagi. Perlu diingat bahwa setiap kali diulangi satu set nilai, maka didapatkan suara (periodik) bernada atau disebut dengan *pitch*. *Pitch* yang didapatkan adalah langsung berhubungan dengan ukuran penyangga (jumlah angka yang dikandungnya), karena setiap kali melalui *buffer* merupakan salah satu siklus lengkap (atau periode) dari sinyal.
3. Sekarang inilah trik untuk algoritma *Karplus Strong*: setiap kali membaca nilai dari *buffer*, maka rata-rata dengan nilai terakhir yang terbaca. Ini adalah nilai ini rata-rata yang kita gunakan sebagai sampel *output*. Kemudian mengambil sampel rata-rata, dan pakan kembali ke *buffer*. Dengan begitu, dari waktu ke waktu, *buffer* menjadi lebih besar dan lebih rata-rata.

Rumus sederhana dari algoritma Karplus Strong dapat dinyatakan dalam bentuk persamaan berikut ini:

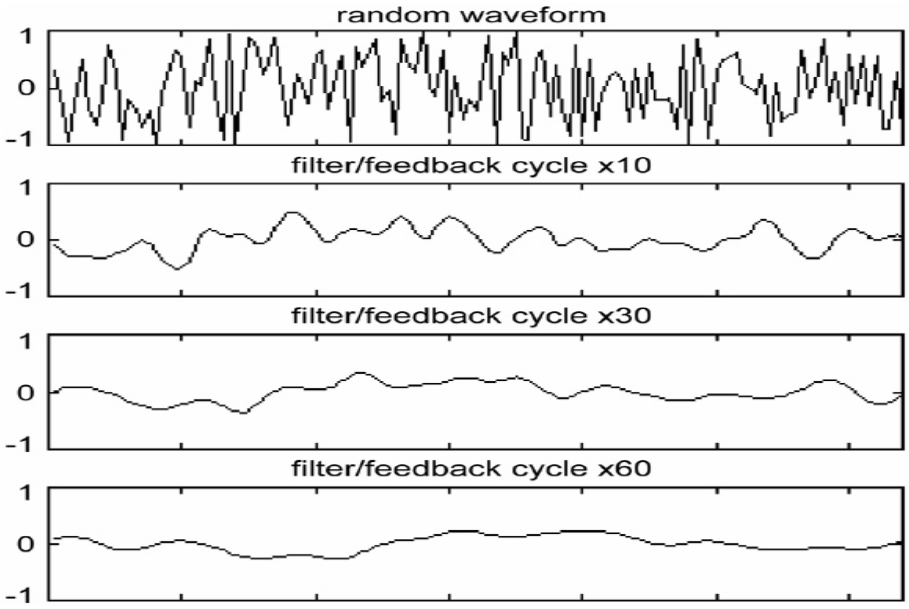
$$Y_t = ½ (Y_{t-p} + Y_{t-p-1})$$

Dimana t = waktu dan p = nilai *pitch*

Sebagai contoh misalkan untuk nilai t = 2 dan nilai p =1 dan nilai frekuensi Y_0 = 400 Hz dan Y_1 = 480 Hz, maka untuk nilai frekuensi dari Y_2 dapat dihitung $Y_2 = ½ (Y_1 + Y_0)$ sehingga $Y_2 = ½ (400 + 480) =$ 440 Hz. Gambar berikut ini menjelaskan contoh *random wave form* yang dapat menghasilkan nada *string* dipetik berdasarkan atas algoritma *Karplus Strong*. Tabel 2.1 memperlihatkan nada *Karplus Strong* untuk menghasilkan nada gitar.

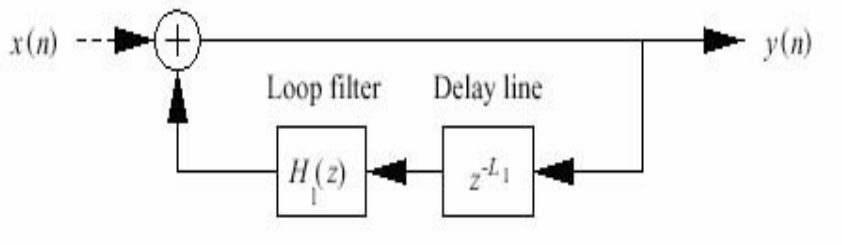
Tabel 1 Tabel Frekuensi Nada Gitar

No.	Nada	Frekuensi (Hz)	No.	Nada	Frekuensi (Hz)
1.	C	400	5.	G	450
2.	D	480	6.	A	455
3.	E	440	7.	B	452.5
4.	F	460	8.	C’	453.75



Gambar 3 Contoh *Output* Frekuensi Nada dengan *Karplus Strong*

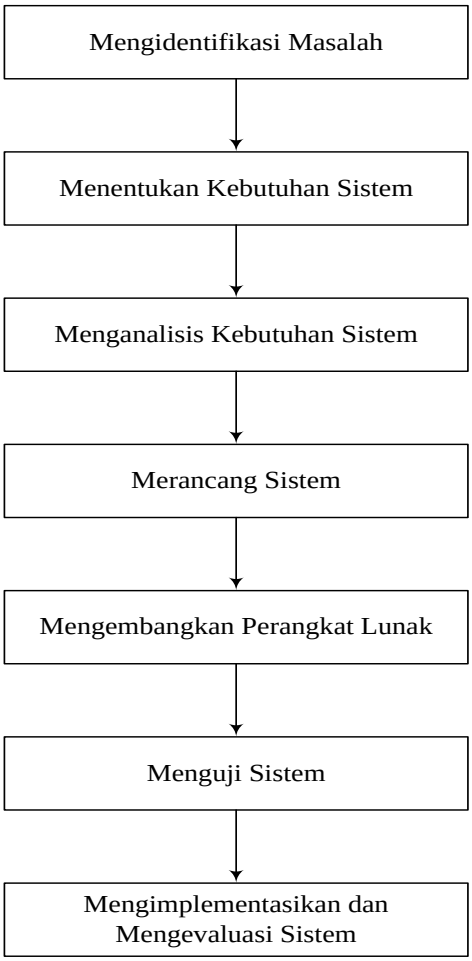
Untuk catatan setiap *switch* membalik dan *buffer* memori komputer diisi dengan nilai acak (*noise*). Untuk menghasilkan sampel, nilai-nilai yang dibaca dari *buffer* dan dirata-ratakan. Sampel yang baru dihitung dikirim baik ke aliran *output* dan diumpan balik kembali ke *buffer*. Ketika akhir *buffer* tercapai maka akan diulang dan terus membaca dari awal. *Setup* semacam ini sering disebut *circular buffer*. Setelah banyak iterasi proses ini, isi *buffer* akan telah berubah dari kebisingan menjadi gelombang sederhana seperti dijelaskan pada Gambar 2.11.



Gambar 4 Skema Implementasi algoritma *Karplus Strong*

2. Metode Penelitian

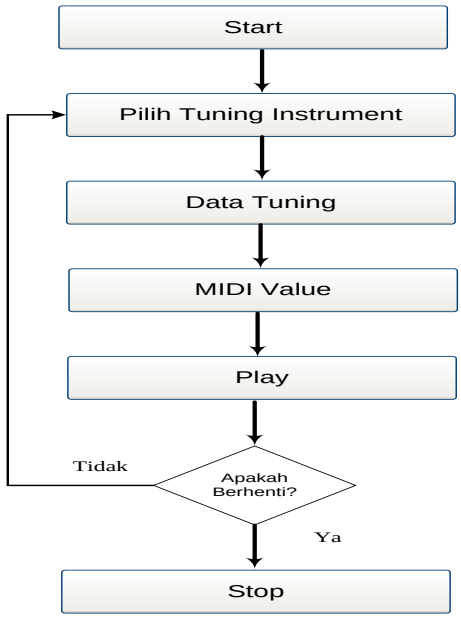
Adapun tahapan dan langkah-langkah pengembangan perangkat lunak pengenalan alat musik ini dapat digambarkan dalam bentuk diagram alir seperti diperlihatkan pada gambar 5.



Gambar 5 Metodologi Perancangan

2.1 Algoritma / Cara Kerja Perangkat Lunak Tuning

Secara umum proses dan cara kerja dari program *tuner* adalah dapat dijelaskan dengan menggunakan diagram seperti ditunjukkan pada Gambar 6.



Gambar 6 Langkah-Langkah Proses *Tuning Standard*

Langkah awal adalah meminta input *tuning instrument* dimana pada bagian ini merupakan pemilihan jenis alat musik yang akan di-*tuner* serta jenis *tuner*. Setelah itu untuk memperoleh *data tuning* berupa frekuensi yang sesuai dengan pilihan maka akan dibaca melalui *database* untuk memperoleh keterangan dan semua informasi mengenai *data tuning*. Proses selanjutnya adalah menginisialisasikan *data tuning* ini ke dalam suatu *variable* MIDI *Value*. Selanjutnya adalah menyiapkan semua MIDI *Value* ini pada semua tombol yang sesuai dengan senar yang mengandung MIDI *Value* ini. Setelah didapat nilai MIDI *Value* ini maka nilai inilah yang akan dimainkan bersamaan dengan *duration note*.

2.2 Tahapan Menentukan Jenis Instrument yang Didukung oleh MIDI

Pada suatu *sound card* biasanya terdapat suatu *chip* yang ditujukan untuk memainkan format MIDI. *Chip* tersebut mengontrol *input* dan *output* dari sintesis MIDI. Secara *default* tersedia 128 macam *instrument* yang dapat disintesis dengan *chip* ini. Setiap *instrument* mempunyai kodenya tersendiri, jadi untuk memainkan suatu *instrument* maka format *file* MIDI telah menyimpan kode tersebut dan dengan perintah *play* langsung mengirim kode tersebut ke *sound card*, selanjutnya adalah *sound card* yang mengatur kode tersebut dan memainkan pilihan *instrument* sesuai dengan kode, frekuensi, dan durasinya.

Tabel 2 berikut merupakan daftar kode instrument yang secara *default* didukung oleh *sound card*.

Tabel 2 Kode *Instrument* pada General MIDI

Nama Instrument	Kode #
ACOUSTIC GUITAR NYLON	24
ACOUSTIC GUITAR STEEL	25
ELECTRIC GUITAR JAZZ	26
ELECTRIC GUITAR CLEAN	27
ELECTRIC GUITAR MUTED	28
OVERDRIVEN GUITAR	29
DISTORTION GUITAR	30

Pada pilihan jenis *instrument* ini dari 128 pilihan yang ada hanya dipakai tujuh pilihan untuk alat musik gitar sebagai pilihan *default* yaitu pilihan dari kode *instrument* #24 hingga #30 yaitu untuk *Acoustic Guitar* (*Nylon*), *Acoustic Guitar* (*Steel*), *Electric Guitar* (*Jazz*), *Electric Guitar* (*Clean*), *Electric Guitar* (*Muted*) *Overdriven Guitar*, dan *Distortion Guitar*.

2.3 Tahapan Menentukan Pilihan Jenis Data *Tuning*

Pilihan jenis *tuning* merupakan pilihan untuk menentukan nada yang harus disetel pada masing-masing senar pada alat musik petik. Semua nilai tersebut merupakan nilai dalam bentuk frekuensi dan untuk satu jenis pilihan *tuning* maka nilai tersebut sudah merupakan ketentuan dan telah baku. Aplikasi yang dirancang mampu melakukan *tuning* dengan pilihan 36 *tuning*. Tabel 3 berikut menggambarkan jenis-jenis *tuning* yang ada dan beserta nilai frekuensinya.

Tabel 3 Pilihan Jenis *Tuning*

No.	Nama Tuning	Frekuensi					
		String #1	String #2	String #3	String #4	String #5	String #6
1.	Standard Tuning	64	59	55	50	45	40
2.	Bass Standard	47	43	38	33	28	35
3.	Dropped-D	64	59	55	50	45	38
4.	Double Dropped-D	64	59	55	50	45	38
5.	Down 1/2 Step	63	58	54	49	44	39
6.	Down 1 (Whole) Step	62	57	53	48	43	38
7.	Down 1 1/2 Steps	61	56	52	47	42	37
8.	Down 2 Steps	60	55	51	46	41	36
9.	Open D	62	57	54	50	45	38
10.	Open G	62	59	55	50	43	38
11.	Open C (Type 1)	64	60	55	48	43	36
12.	Open C (Type 2)	64	60	55	52	43	36
13.	Open E	64	59	56	52	47	40
14.	Open A	64	61	57	52	49	45
15.	Cross-Note	62	57	53	50	45	38
16.	D Modal (Type 1)	62	57	54	50	45	38
17.	D Modal (Type 2)	64	59	55	50	43	38
18.	D Modal (Type 3)	64	59	55	50	43	36
19.	D Modal (Type 4)	62	57	55	50	45	38
20.	Fourths	65	60	55	50	45	40
21.	Lute (Type 1)	64	59	54	50	45	40
22.	Lute (Type 2)	64	57	54	50	45	40
23.	Big City	69	57	54	50	45	38
24.	D Wahine	62	59	54	50	45	38
25.	D Minor	62	57	53	50	45	38
26.	D Modal (Type 5)	62	57	50	50	45	38
27.	G6	64	59	55	50	43	38
28.	G Minor	62	58	55	50	43	38
29.	C6 (Type 1)	64	59	55	48	43	36
30.	Bron-Y-Aur	64	60	55	48	45	36
31.	Parvardigar	62	60	55	48	43	36
32.	Bruce Palmer Modal	64	59	52	52	47	40
33.	New Standard	67	64	57	50	43	36
34.	Low C	62	57	55	50	43	36
35.	G Modal	62	60	55	50	43	38
36.	A Minor	64	60	57	52	48	45

2.4 Tahapan Penentuan Durasi Not

Untuk menentukan lama suatu not dimainkan maka harus ditentukan durasi dari not tersebut sewaktu dimainkan. Durasi dapat ditentukan dalam satuan detik dengan rentang 1 hingga 9.

2.5 Tahapan Perhitungan Nada yang Akan Dimainkan (MIDI Value)

Pada saat sebuah frekuensi dimainkan dengan cara mengirim secara langsung pada MIDI output maka harus dilakukan perhitungan nilainya terlebih dahulu agar didapat nilai frekuensi yang cocok. Perhitungan menggunakan dua buah persamaan dimana persamaan pertama (1) digunakan untuk memainkan nada sedangkan persamaan kedua (2) digunakan untuk menghentikan nada yang dimainkan, terakhir persamaan ketiga (3) digunakan untuk meng-*update* nilai *instrument*. Adapun bentuk persamaan untuk tujuan ini adalah sebagai berikut:

StartNote = &H90 + ((intBaseNote + NoteValue) * &H100) +
(intVolume * &H10000) + intChannel .....(1)
StopNote = &H80 + ((intBaseNote + NoteValue) * &H100) +
intChannel .....(2)
UpdateInstrument = (intInstrument * 256) + &HC0 + intChannel +
(0 * 256) * 256.....(3)

Dimana penjelasan dari nilai parameter ini adalah sebagai berikut:

a. *intBaseNote* merupakan *parameter* dari nilai awal nada dasar yang secara *default* bernilai 0.
b. *intChannel* merupakan parameter untuk menyatakan jumlah kanal yang digunakan. Nilai 1 berarti dua kanal atau *stereo* dan nilai 0 berarti satu kanal (*mono*). Secara *default* nilai ini akan diset menjadi nilai 1.
c. *intInstrument* merupakan parameter untuk menyatakan kode *instrument* yang akan dimainkan. Secara *default* akan diset menjadi nilai 0.
d. *intVolume* merupakan parameter untuk menyatakan volume dari frekuensi nada yang dimainkan. Jika diset dengan konstanta MIDI_MAX_VOLUME berarti parameter ini bernilai 127.
e. *NoteValue* merupakan parameter dari nilai frekuensi not pilihan jenis *tuning* yang akan dikirim (Tabel 3.2)
f. Konstanta bertanda &H merupakan nilai dalam bentuk heksadesimal.

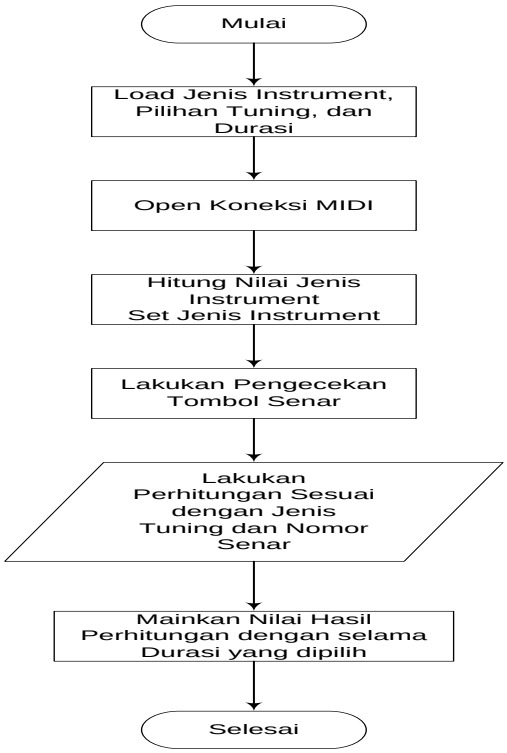
2.6 Proses Play

Proses *play* dilakukan dengan mengirimkan nilai MIDI atau MIDI *value* yang telah dihasilkan ke dalam sebuah MIDI *Mapper* dengan tujuan untuk menghasilkan nada gitar dengan frekuensi tertentu sesuai dengan nilai frekuensi yang dihitung.

2.7 Algoritma

Algoritma merupakan langkah-langkah maupun urutan bertahap dan spesifik dari suatu masalah. Algoritma ini kemudian diterjemahkan ke dalam program dengan menggunakan bahasa pemrograman tertentu. Algoritma digunakan untuk menganalisis serta menjelaskan urutan dan hubungan antara kegiatan-kegiatan yang akan ditempuh. Selain itu algoritma juga berfungsi untuk menyelesaikan suatu permasalahan sehingga tercapai tujuan yang diinginkan. Agar lebih mudah dipahami maka algoritma pada program yang dirancang akan dijelaskan dengan menggunakan *flowchart*.

Pada *flow chart* program ini dapat dijelaskan sebagai berikut. Langkah awal adalah menginput jenis *instrument* dan jenis pilihan *tuning* serta durasi sebuah not dimainkan. Selanjutnya adalah membuka koneksi *device* MIDI. Kemudian dilanjutkan dengan menghitungkan nilai jenis *instrument* kemudian meng-update nilai tersebut. Proses selanjutnya adalah jika terjadi klik pada tombol senar maka lakukan pengecekan indeks tombol yang diklik dan hitung nilainya sesuai dengan jenis *tuning* yang dipilih. Setelah itu mainkan nilai yang dihitung melalui *device* MIDI sesuai dengan durasi yang telah ditentukan. Agar lebih jelasnya proses ini dapat dilihat pada gambar 7 berikut ini.



Gambar 7 *Flow Chart* Memainkan Tuning Melalui MIDI

2.7.1 Algoritma Menambah dan Menentukan Jenis *Instrument*

Berikut ini merupakan algoritma untuk menginput jenis instrument dan mengecek jenis instrument yang dipilih.

```
cmbInst.AddItem "Acoustic Guitar (Nylon)"
cmbInst.AddItem "Acoustic Guitar (Steel)"
cmbInst.AddItem "Electric Guitar (Jazz)"
cmbInst.AddItem "Electric Guitar (Clean)"
cmbInst.AddItem "Electric Guitar (Muted)"
cmbInst.AddItem "Overdriven Guitar"
cmbInst.AddItem "Distortion Guitar"

Select Case strLCase
    Case "acoustic guitar (nylon)"
       ValInstrument = 24
   Case "acoustic guitar (steel)"
       ValInstrument = 25
   Case "electric guitar (jazz)"
       ValInstrument = 26
   Case "electric guitar (clean)"
       ValInstrument = 27
   Case "electric guitar (muted)"
       ValInstrument = 28
   Case "overdriven guitar"
       ValInstrument = 29
   Case "distortion guitar"
       ValInstrument = 30
End Select
```

2.7.2 Algoritma Memainkan Note

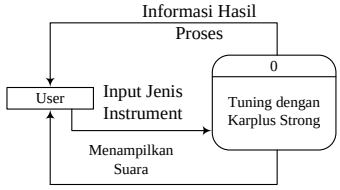
StartNote NoteValue
PauseNow Duration
StopNote NoteValue

2.7.3 Algoritma Menghitung Nilai Frekuensi Note

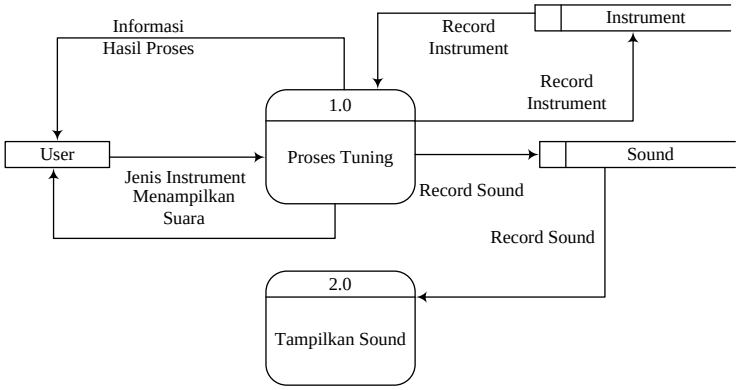
Get intBaseNote, Note Value, intVolume, intChannel
lonMsg = &H90 + ((intBaseNote + NoteValue) * &H100) +
(intVolume * &H10000) + intChannel

2.8 Pemodelan Sistem

Berikut ini merupakan penjelasan dalam bentuk *data flow diagram* dari proses kerja dari aplikasi. Pada sistem ini, *user* akan memilih *file* yang akan disisip, diekstrak berikut dengan pesannya. Setelah proses selesai dilakukan maka sistem akan menampilkan hasil proses yang telah dilakukan seperti ditunjukkan pada Gambar 8 dan Gambar 9.



Gambar 8 Diagram Konteks



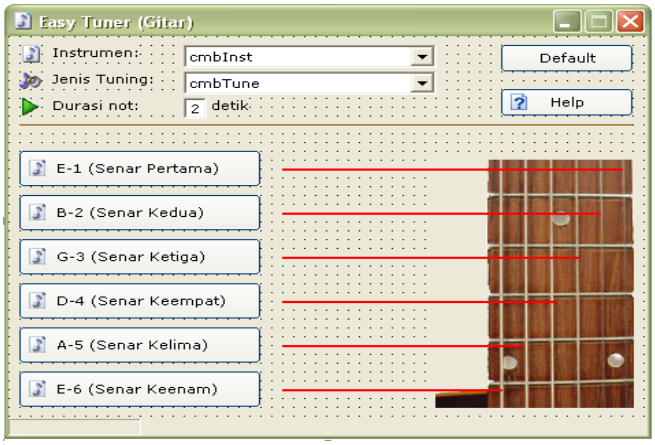
Gambar 9 *Data Flow Diagram Level* 0

3. Hasil dan Pembahasan

Dalam aplikasi dirancang dua buah *form* yaitu *form* utama, *form other*, *form customize*, *form menu*, dan *form splash screen*. Semua *form* ini dirancang dengan menggunakan objek dasar dari *Visual Basic* seperti *command button*, *option button*, *label*, *image*, *combo box*, dan *text box*. Pada Gambar 10 di bawah ini merupakan bentuk rancangan dari *form splash screen* yang akan muncul pertama sekali sewaktu program di-*load*.

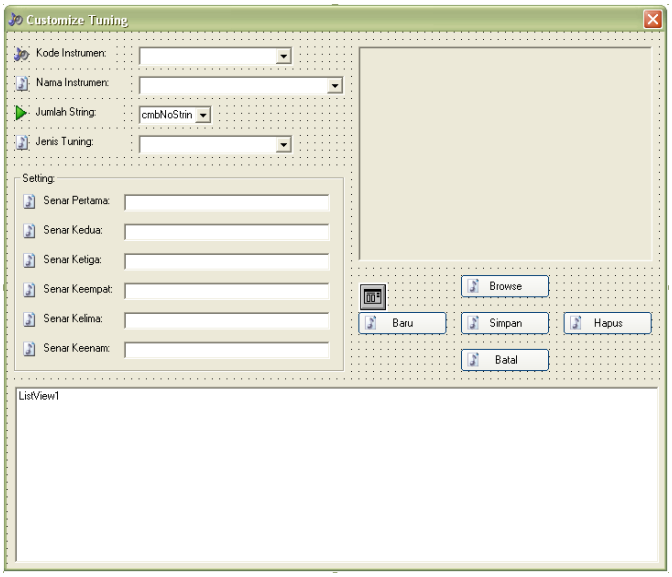
Gambar 10 Rancangan *Dialog Splash Screen*

Bentuk rancangan *form* berikutnya adalah *form* utama seperti terlihat pada Gambar 11 berikut ini.



Gambar 11 Rancangan *Form* Utama

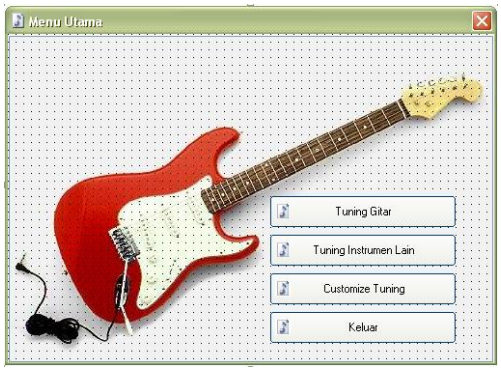
Berikutnya adalah bentuk rancangan dari *form customize.* Form ini digunakan untuk menginput jenis tuning untuk instrument lainnya sehingga tuning tidak hanya terbatas pada alat musik gitar saja. Komponen utama dari *form* ini hanya menggunakan objek *picture*, *label*, *listview*, *common dialog*, *command button*, dan *text box*. *Picture* digunakan untuk menampilkan gambar alat musik seperti ditunjukkan pada Gambar . 12.



Gambar 12 Rancangan *Form Customize*

Untuk membedakan *tuning* yang secara khusus untuk gitar dengan *tuning* yang dibuat sendiri oleh *user* maka hasil *tuning* yang diinput sendiri akan dapat dimainkan pada *form other*. *Form* ini sangat mirip dengan *form main* dengan komponen rancangan menggunakan *label*, *combo box*, *command button*, dan *picture box*.

Bentuk rancangan *form* terakhir pada program ini adalah rancangan menu utama yang dipergunakan untuk mengakses dan menampilkan *form-form* yang lain. Terdapat empat pilihan pada *form* ini yang menggunakan *command button* dalam rancangannya. *Button* pertama digunakan untuk menampilkan *form tuning* gitar (*form main*), *button* kedua digunakan untuk menampilkan *tuning* alat musik lain hasil *customize*, *button* ketiga dipergunakan untuk menginput tuning secara *customize* oleh *user*, dan terakhir *button* keempat dipergunakan untuk keluar dari program ini seperti ditunjukkan pada Gambar 13.



Gambar 13 Rancangan *Form Menu*

3.1 Perancangan Module Memainkan MIDI

Untuk memainkan *file* format MIDI juga dapat dengan mudah dilakukan dengan memanggil sebuah fungsi API dalam *library* "winmm.dll". Bedanya untuk *file* format MIDI fungsi yang dipanggil adalah *mciSendString*, *mciGetDeviceID*, *mciSendCommand*. Perintah pertama untuk mengirim perintah seperti membuka (*open*), memainkan (*Play*), dan menghentikan (*stop*) *file* MIDI. Perintah kedua hanya mengecek peralatan *multimedia*, sedangkan perintah ketiga mempunyai fungsi yang hampir sama dengan *mciSendString*.

Agar fungsi ini dapat berjalan maka perlu dibuat deklarasinya pada bagian *module* seperti baris kode di bawah ini:

```
Declare Function mciSendString Lib "winmm.dll" Alias "mciSendStringA" (ByVal lpstrCommand As String, ByVal lpstrReturnString As String, ByVal uReturnLength As Long, ByVal hwndCallback As Long) As Long
Declare Function mciGetDeviceID Lib "winmm.dll" Alias "mciGetDeviceIDA" (ByVal lpstrName As String) As Long
Declare Function mciSendCommand Lib "winmm.dll" Alias "mciSendCommandA" (ByVal wDeviceID As Long, ByVal uMessage As Long, ByVal dwParam1 As Long, ByVal dwParam2 As Long) As Long
Private Declare Function midiOutShortMsg Lib "winmm.dll" (ByVal hMidiOut As Long, ByVal dwMsg As Long) As Long
```

Perintah di bawah ini adalah contoh baris kode untuk memainkan dan menghentikan *file* MIDI:

```
nReturn = mciSendString("Open " & sFile & " ALIAS " & sAlias & "
TYPE " & "Sequencer", "", 0, 0)
nReturn = mciSendString("Play " & sAlias, "", 0, 0)
nReturn = mciSendString("Stop " & sAlias, "", 0, 0)
nReturn = mciSendString("Close " & sAlias, "", 0, 0)
```

Perintah di bawah ini adalah contoh baris kode untuk mengecek apakah ada *file* MIDI yang sedang dimainkan atau tidak.

```
nReturn = mciSendString("Status " & sAlias & " mode", sStatus, 255, 0)
```

Perintah di bawah ini adalah contoh baris kode untuk memainkan nada dengan frekuensi dan interval tertentu.

```
midiOutShortMsg lonMIDIHand, lonMsg
```

Dimana lonMIDIHand adalah nama *device* dari MIDI sedangkan lonMsb merupakan frekuensi nada yang akan dimainkan.

3.2 Perancangan Grafik

Grafik merupakan elemen yang paling penting dan mendasar dari pembuatan suatu aplikasi *multimedia*. Grafik yang bagus akan menambah minat seseorang untuk menggunakan aplikasi tersebut. Untuk keperluan grafik dalam aplikasi ini dipakai beberapa program pengolah grafik seperti *Paint, Adobe PhotoShop 7.0*, *Macromedia FreeHand 10* serta *Corel Draw 10*. Elemen dasar grafik dibagi yang dihasilkan akan ditempatkan pada *form* utama dan untuk merancang halaman *web* untuk *Help* seperti ditunjukkan pada Gambar 14.

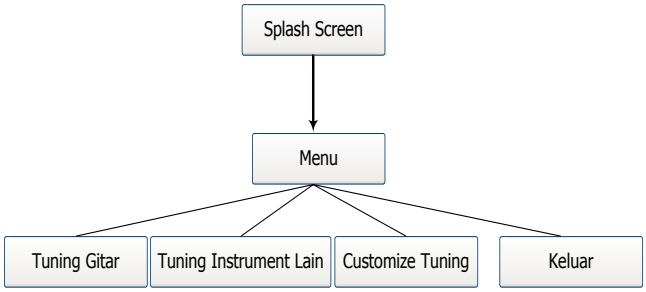
Grafik Leher Gitar

Grafik Icon

Gambar 14 Rancangan Grafik

3.3 Struktur Aplikasi

Struktur aplikasi menggambarkan hubungan pemanggilan *form* dalam suatu aplikasi. Pada program *tuning* alat musik petik ini struktur aplikasi dapat digambarkan pada Gambar 15 berikut ini.



Gambar 15 Struktur Aplikasi

4. Kesimpulan

Dengan adanya perangkat lunak *tuning* untuk alat musik petik maka dapat membantu *user* dalam melakukan proses *tuning* sendiri dan untuk melakukan *stemming* pada alat musik gitar. *Tuning* dilakukan dengan memanfaatkan MIDI *Value* yang dimainkan secara langsung dengan mengirimkan nilai tersebut melalui perintah midiOutShortMsg berdasarkan nilai yang dihasilkan oleh algoritma *Karplus Strong Algorithm*. Pemilihan proses *tuning* dapat dilakukan untuk 36 jenis *tuning* yang berbeda dimana pilihan ini sudah mencakup seluruh *tuning* alat musik seperti gitar.

Referensi :

Buku Teks:

[1] Dennis Siahaan, *Teknik Menulis Note Balok dan Memainkan MIDI Menggunakan Encore*, ANDI, 2004.
[2] Para Editor Majalah Guitar Player, *How To Play Guitar: Teknik Dasar & Lanjutan untuk Gitar Elektrik & Akustik*, ISBN 979-22-1459-3, Penerbit PT. Gramedia.
[3] Smith, Julios, O, 2005, *Virtual Acoutic Musical Intruments: Review of Models and Selected Research*, Center for Computer Research in Music and Acoustics (CCRMA), Department of Music, Stanford University, Stanford, California.
[4] Yayan Sopyan, *Panduan Tracking Mengomposisi Musik Dengan Komputer*, PT. Kawan Pustaka, 2003.
[5] (http://cnx.org/content/m19006/latest/)