# RANCANG BANGUN PERANGKAT LUNAK KOMPOSER MUSIK MENGGUNAKAN MATLAB

## SKRIPSI

**OLEH**

**RONALD WILSON**

**04 04 03 0741**

**DEPARTEMEN TEKNIK ELEKTRO**

**FAKULTAS TEKNIK UNIVERSITAS INDONESIA**

**GENAP 2007/2008**

# RANCANG BANGUN PERANGKAT LUNAK KOMPOSER MUSIK MENGGUNAKAN MATLAB

## SKRIPSI

**OLEH**

**RONALD WILSON**

**04 04 03 0741**

**SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK**

**DEPARTEMEN TEKNIK ELEKTRO**

**FAKULTAS TEKNIK UNIVERSITAS INDONESIA**

**GENAP 2007/2008**

# PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul:

**RANCANG BANGUN PERANGKAT LUNAK KOMPOSER MUSIK MENGGUNAKAN MATLAB**

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau Instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, Mei 2008

Ronald Wilson

NPM. 04 04 03 0741

# PENGESAHAN

Skripsi dengan judul:

**RANCANG BANGUN PERANGKAT LUNAK KOMPOSER MUSIK MENGGUNAKAN MATLAB**

dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan dalam sidang ujian skripsi.

Depok, Mei 2008

Dosen Pembimbing

Dr. Ir. Arman Djohan, M.Eng

NIP. 131 476 472

# UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada :

**Dr.Ir.Arman Djohan Diponegoro, M.Eng**

selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberi pengarahan, diskusi dan bimbingan serta persetujuan sehingga skripsi ini dapat selesai dengan baik.

Ronald Wilson
NPM 04 04 03 074 1
Departemen Teknik Elektro

Dosen Pembimbing
Dr.Ir.Arman Djohan Diponegoro, M.Eng

**RANCANG BANGUN PERANGKAT LUNAK KOMPOSER MUSIK MENGGUNAKAN MATLAB**

**ABSTRAK**

Saat ini, teknologi pemrosesan sinyal dijital sudah banyak digunakan dan berkembang di berbagai bidang. Salah satu aplikasi dari pemrosesan sinyal dijital adalah dalam bidang audio, salah satu contohnya adalah musik. Walaupun bidang musik penerapannya banyak di dunia hiburan, tetapi dibalik itu semua terdapat teknologi yang sebenarnya tidak sederhana.

Untuk membuat musik sendiri, biasanya diperlukan alat-alat musik seperti: piano, bass, dan lain-lain. Namun, tanpa bantuan alat musik, musik sendiri tetap dapat dibuat dengan bantuan perangkat lunak komposer musik. Perangkat lunak komposer musik ini dibuat dengan bantuan MATLAB. Melalui perangkat lunak ini, dapat diciptakan musik sendiri sesuai dengan keinginan sendiri, tanpa bantuan alat musik.

Pada perangkat lunak komposer musik ini, disediakan suara piano, bass, string, juga suara gelombang sinusoidal. Format *database* suara yang digunakan adalah WAV, kecuali gelombang sinusoidal. Melalui pemrosesan sinyal dijital, suara-suara tersebut dapat digabungkan serta dapat diatur tempo dan *volume*nya, sehingga dapat dihasilkan sebuah musik tepat seperti yang diinginkan.

**Kata kunci : Pemrosesan Sinyal Dijital, Perangkat Lunak, Musik, WAV**

Ronald Wilson
NPM 04 04 03 074 1
Departement Electical Engineering

Guidance lecturer
Dr.Ir.Arman Djohan Diponegoro, M.Eng

**MUSIC COMPOSER SOFTWARE USING MATLAB**

**ABSTRACT**

Nowadays, digital signal processing technology has been applied and develop in many ways. The technology can be applied in audio, which is music, for instance. Music has its advantages in entertaintment industries, but the technology itself is more complicated than it was seen by most user.

In producing music, there has to be instruments, such as: piano, bass, etc. Nevertheless, there are alternate way to produce music, by using Music Composer Software. This software is created by using MATLAB. This software enable us to produce our own music as we want, without playing any music insruments.

This software is inserted with various music instruments samples, such as piano, bass, string, and sinusoidal wave. All the audio samples was formatted in WAV, except sinusoidal wave. By applying digital signal processing, we can combine the audio samples and set the volume and tempo to create our own music as we want them to be.

**Key words : Digital Signal processing, Software, Music, WAV**

# DAFTAR ISI


| | | Halaman |
|---|---|---|
| PERNYATAAN KEASLIAN SKRIPSI | | i |
| PERSETUJUAN | | ii |
| UCAPAN TERIMA KASIH | | iii |
| ABSTRAK | | iv |
| ABSTRACT | | v |
| DAFTAR ISI | | vi |
| DAFTAR GAMBAR | | viii |
| DAFTAR TABEL | | ix |
| DAFTAR LAMPIRAN | | x |
| BAB 1. PENDAHULUAN | | 1 |
| 1.1 | LATAR BELAKANG | 1 |
| 1.2 | TUJUAN PENULISAN | 2 |
| 1.3 | BATASAN MASALAH | 2 |
| 1.4 | SISTEMATIKA PENULISAN | 2 |
| BAB 2. TEORI PEMROSESAN SINYAL DIJITAL DAN MUSIK | | 3 |
| 2.1 | DASAR PEMROSESAN SINYAL DIJITAL (PSD) | 3 |
| 2.1.1 | DFT dan FFT | 3 |
| 2.1.2 | Frekuensi sampling | 4 |
| 2.1.3 | WAV | 4 |
| 2.2 | REPERSENTASI SUARA | 4 |
| 2.2.1 | *Analog to Digital Convertion* (ADC) | 5 |
| 2.2.2 | *Digital to Analog Convertion (DAC)* | 6 |
| 2.3 | TEORI MUSIK | 7 |
| 2.3.1 | Suara (Sound) | 7 |
| 2.3.2 | Nada Dan Nilainya | 8 |
| 2.3.3 | Tanda Kromatis | 9 |
| 2.3.4 | Tangga Nada | 10 |
| | *2.3.4.1 Tangga nada kromatik* | 10 |

*2.3.4.2 Tangga nada mayor* 12
2.3.5 Interval 13
2.3.6 Pasangan Chord 15
2.3.7 *Pitch* 16
2.3.8 Model Nada 17
2.3.9 BPM (*Beat per minute*) 18
2.3.10 Oktaf 19
BAB 3. PERANGKAT LUNAK KOMPOSER MUSIK 20
3.1 GAMBARAN UMUM PERANGKAT LUNAK 20
3.2 FASILITAS DAN PENGGUNAAN PERANGKAT LUNAK 21
3.3 DIAGRAM ALIR PERANGKAT LUNAK KOMPOSER MUSIK 23
3.3.1 Membaca Sumber Suara 24
3.3.2 Membaca Panjang Nada Dan Tempo 24
3.3.3 Membaca *Text Box* 25
3.3.4 Membuat FFT 27
3.3.5 Mengatur *Volume* 28
3.3.6 Mengatur Mute Dan Solo 29
3.3.7 Menggabungkan Suara Dan Membuat IFFT 30
3.3.8 Membunyikan Lagu Dan Memplot Grafik 30
BAB 4. HASIL DAN ANALISIS PERANGKAT LUNAK KOMPOSER MUSIK 32
4.1 ANALISIS GRAFIK 32
4.2 ANALISIS MENGGABUNGKAN SUARA DENGAN PEMBATASAN JUMLAH KOLOM 35
BAB 5. KESIMPULAN 37
DAFTAR ACUAN 38
DAFTAR PUSTAKA 39
LAMPIRAN 40

# DAFTAR GAMBAR


| | | Halaman |
|---|---|---|
| **Gambar 2.1** | Diagram blok ADC | 5 |
| **Gambar 2.2** | Langkah-langkah ADC | 6 |
| **Gambar 2.3** | Proses perambatan suara | 8 |
| **Gambar 2.4** | Sebuah not | 8 |
| **Gambar 2.5** | Not dan nilainya | 9 |
| **Gambar 2.6** | Visualisasi tangga nada kromatik | 11 |
| **Gambar 2.7** | Visualisasi tangga nada mayor dalam paranada | 12 |
| **Gambar 2.8** | Visualisasi tangga nada mayor dengan piano | 12 |
| **Gambar 2.9** | Grafik dari amplitudo energi dalam domain waktu untuk nada *single* | 18 |
| **Gambar 2.10** | Visualisasi nada pada domain frekuensi | 18 |
| **Gambar 3.1** | Tampilan perangkat lunak komposer musik | 20 |
| **Gambar 3.2** | Diagram alir perangkat lunak komposer musik | 23 |
| **Gambar 4.1** | Grafik lagu *twinkle twinkle little star* | 32 |
| **Gambar 4.2** | Grafik lagu *twinkle twinkle little star* setelah diubah | 33 |
| **Gambar 4.3** | Hasil dari *Cool Edit Pro 2.0* | 34 |
| **Gambar 4.4** | Hasil *Cool Edit Pro 2.0* setelah diubah ke satuan desibel | 35 |

# DAFTAR TABEL


| | | Halaman |
|---|---|---|
| **Tabel 2.1** | Tangga Nada Kromatik dalam Musik | 10 |
| **Tabel 2.2** | Tangga nada kromatik dalam musik | 11 |
| **Tabel 2.3** | Interval pada musik | 13 |
| **Tabel 2.4** | Tingkat interval | 14 |
| **Tabel 2.5** | Penggolongan triad | 16 |

# DAFTAR LAMPIRAN


| | | Halaman |
|---|---|---|
| **Lampiran 1** | Frekuensi dan Panjang Gelombang Nada | 40 |

# BAB 1
# PENDAHULUAN

## 1.1 LATAR BELAKANG

Aplikasi dari pemrosesan sinyal dijital saat ini sangat luas dan sudah merambah ke berbagai bidang dalam kehidupan manusia. Salah satu aplikasinya adalah dalam bidang audio. Aplikasi pada bidang audio adalah sesuatu yang menarik. Walaupun penerapannya banyak di dunia hiburan, tetapi di balik itu semua terdapat teknologi yang sebenarnya tidak sederhana. Salah satu contoh aplikasi dari audio adalah dalam bidang musik. Musik terdiri dari nada-nada, sedangkan nada-nada terdiri dari beberapa komponen seperti: frekuensi, amplitudo, durasi, dan sebagainya.

Untuk membuat musik sendiri, biasanya dibutuhkan alat-alat musik, seperti piano, bass, string, dan sebagainya. Bila tidak memiliki alat musik, maka dibutuhkan suatu perangkat lunak yang dapat menyusun musik seperti yang dihendaki. Melalui bantuan MATLAB, disusunlah sebuah perangkat lunak komposer musik. Tampilan perangkat lunak komposer musik ini diperindah dengan menggunakan GUI (*Graphical User Interface*). Dengan menggunakan GUI, perangkat lunak ini dapat dijalankan secara *user friendly*. Melalui perangkat lunak ini, musik sendiri dapat dibuat dengan tempo dan *volume* yang berbeda-beda, sesuai keinginan sendiri.

Format suara yang digunakan adalah WAV (*Waveform Audio Format*), sebab format ini adalah format dasar yang mudah untuk sampling audio. Semua sampel suara berbentuk WAV, kecuali suara gelombang sinus, sebab suara gelombang sinus tidak diambil dari *database*, melainkan dibangkitkan dengan perintah tertentu. Pada skripsi ini digunakan contoh lagu *twinkle-twinkle little star*, karena lagu tersebut nadanya tidak rumit, durasinya tidak lama, dan mudah untuk diaplikasikan.

Sebenarnya sudah ada *software* yang mirip seperti perangkat lunak komposer musik ini, namun berbeda formatnya, yaitu MIDI.

## 1.2 TUJUAN PENULISAN

Tujuan penulisan laporan skripsi ini adalah membuat suatu perangkat lunak komposer musik dengan teknik pemrosesan sinyal dijital, sehingga musik yang diinginkan dapat dibuat tanpa bantuan alat musik .

## 1.3 BATASAN MASALAH

Permasalahan difokuskan dan dibatasi pada WAV, pemrosesan sinyal dijital, dan musik yang terdiri dari: piano, bass, string, dan gelombang sinus.

## 1.4 SISTEMATIKA PENULISAN

Skripsi ini dibagi menjadi 5 bab, yaitu :

BAB 1 PENDAHULUAN

Berisi penjelasan mengenai latar belakang, tujuan, batasan masalah, serta sistematika penulisan skripsi.

BAB 2 TEORI PEMROSESAN SINYAL DIJITAL DAN MUSIK

Berisi pemaparan berbagai teori tentang pemrosesan sinyal dijital, serta dasar-dasar musik yang penting.

BAB 3 PERANGKAT LUNAK KOMPOSER MUSIK

Berisi penjelasan mengenai perangkat lunak komposer musik.

BAB 4 HASIL DAN ANALISIS PERANGKAT LUNAK KOMPOSER MUSIK

Berisi analisis dan hasil dari perangkat lunak komposer musik

BAB 5 KESIMPULAN

Berisi penutup yang berupa kesimpulan dari hasil dan analisis pada bab sebelumnya.

# BAB 2
# TEORI PEMROSESAN SINYAL DIJITAL DAN MUSIK

## 2.1 DASAR PEMROSESAN SINYAL DIJITAL (PSD)

PSD merupakan suatu teknologi yang saat ini sudah mendasari hampir seluruh bidang dalam kehidupan manusia. PSD dibedakan dengan bidang *sains* yang lain karena keunikan data yang diolahnya, yaitu sinyal. Sinyal yang diolah pada umumnya berasal dari alam dalam bentuk analog [1]. Lalu sinyal ini diubah dalam bentuk dijital melalui proses *sampling*, kuantisasi, dan *coding*. Semua proses ini dilakukan oleh alat yang bernama ADC (*Analog to Digital Converter*). Setelah sinyal menjadi bentuk dijital, barulah diproses secara dijital oleh prosesor DSP. Hasil keluaran dari proses ini selanjutnya diubah menjadi analog kembali oleh DAC (*Digital to Analog Converter*). Hal ini penting dilakukan karena pada umumnya sinyal yang bisa dilihat atau dengar adalah sinyal analog.

Berikut ini adalah beberapa teori yang penting dalam PSD :

### 2.1.1 DFT dan FFT

DFT termasuk salah satu jenis transformasi. Dengan menerapkan formula DFT, suatu sinyal dalam domain waktu dapat diubah ke dalam domain frekuensi. Pada domain frekuensi, sinyal dipresesntasikan dalam frekuensi (sumbu x) dan magnitude (sumbu y). Jadi dapat dilihat dengan jelas frekuensi kerja dan *power* suatu sinyal ataupun sistem. Formula DFT dapat dilihat pada persamaan 2.1 [2].

$$X(k) = F_D[x(nT)] = \sum_{n=0}^{N-1} x(nT) e^{-jk\Omega nT} \qquad (2.1)$$

FFT sebenarnya memiliki fungsi yang sama dengan DFT, hanya FFT menggunakan algoritma perhitungan yang lebih efisien dibandingkan dengan DFT. Hal ini diperlukan dalam aplikasi untuk meningkatkan efisiensi kerja suatu prosesor DSP.

### 2.1.2 Frekuensi Sampling

Frekuensi sampling (*sample rate*) adalah jumlah *sample* per detik dalam suatu suara [3]. Sebagai contoh: jika frekuensi sampling adalah 44.100 Hertz, rekaman dengan durasi 60 detik akan berisi 2.646.000 *samples*. Nilai yang biasa digunakan untuk frekuensi sampling adalah 44.100 Hertz (untuk kualitas CD) dan 22.050 Hertz (cukup untuk pembicaraan biasa).

Menurut teorema Nyquist, frekuensi sampling minimum adalah dua kali bandwidth dari sinyal yang di sampling untuk mencegah terjadinya aliasing.

### 2.1.3 WAV

WAV (atau WAVE) merupakan singkatan dari istilah dalam bahasa Inggris *waveform audio format,* dan merupakan standar format berkas audio yang dikembangkan oleh Microsoft dan IBM. Format ini adalah format dasar yang mudah untuk sampling audio. WAV merupakan varian dari format *bitstream* RIFF dan mirip dengan format IFF dan AIFF yang digunakan komputer Amiga dan Macintosh. Baik WAV maupun AIFF kompatibel dengan sistem operasi Windows dan Macintosh. Meskipun sebuah file WAV dapat menahan audio yang terkompres, format WAV yang paling umum berisi audio yang belum terkompres dalam bentuk modulasi *pulse-code* (PCM). Audio PCM adalah format audio standar untuk *Compact Disc* (CD) pada 44.100 sampel perdetik, 16 bit persampel. Semenjak PCM menggunakan metode penyimpanan yang tidak terkompres, lossless, yang menyimpan semua sampel dari sebuah track audio, pengguna ahli atau ahli audio dapat menggunakan format WAV untuk audio kualitas maksimum. Audio WAV dapat juga dirubah dan dimanipulasi dengan *software* (perangkat lunak). Software yang dapat menciptakan WAV dari Analog Sound misalnya adalah Windows Sound Recorder. WAV jarang sekali digunakan di internet karena ukurannya yang relative besar. Maksimal ukuran file WAV adalah 2GB.

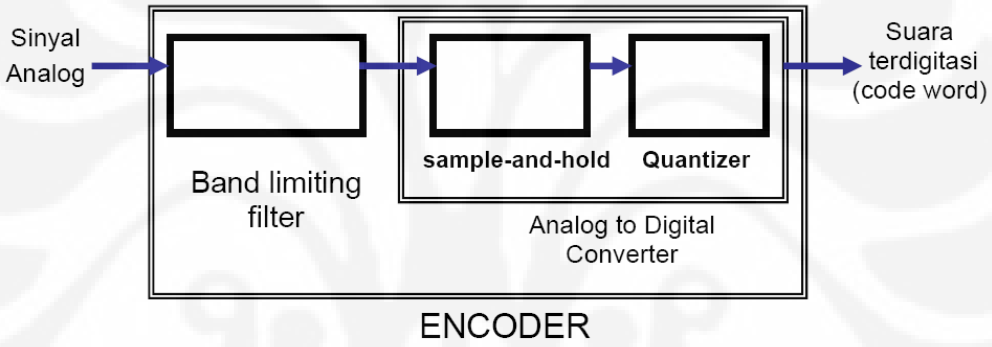
## 2.2 REPRESENTASI SUARA

Gelombang suara analog tidak dapat langsung direpresentasikan pada komputer. Komputer mengukur amplitudo pada satuan waktu tertentu untuk

menghasilkan sejumlah angka. Tiap satuan pengukuran ini dinamakan *"SAMPLE"*. Teknik untuk mengubah sinyal analog menjadi sinyal dijital menggunakan ADC, dan untuk mengubah sinyal dijital menjadi sinyal analog menggunakan DAC.

### 2.2.1 *Analog to Digital Convertion* (ADC)

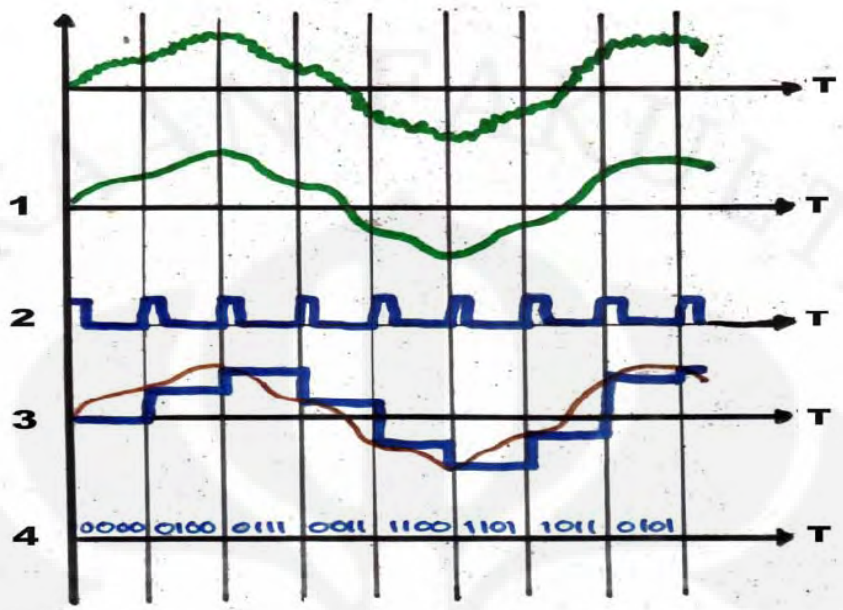
ADC adalah proses mengubah gelombang bunyi ke dalam waktu interval tertentu (disebut juga sampling), sehingga menghasilkan representasi digital dari suara. Gambar 2.1 berikut adalah diagram blok ADC:




Gambar 2.1 Diagram blok ADC

Berikut adalah langkah-langkah dalam ADC yang terlihat pada Gambar 2.2:

1. Membuang frekuensi tinggi dari source signal.
2. Mengambil sample pada interval waktu tertentu (sampling).
3. Menyimpan amplitudo sample dan mengubahnya ke dalam bentuk diskrit (kuantisasi).
4. Merubah bentuk menjadi nilai biner.




Gambar 2.2 Langkah-langkah ADC

Teori Nyquist Sampling Rate adalah : untuk memperoleh representasi akurat dari suatu sinyal analog secara lossless, amplitudonya harus diambil sample-nya setidaknya pada kecepatan (rate) sama atau lebih besar dari 2 kali lipat komponen frekuensi maksimum yang akan didengar. Misalnya: Untuk sinyal analog dengan bandwith 15Hz – 10kHz → sampling rate = 2 x 10KHz = 20 kHz.

### 2.2.2 *Digital to Analog Convertion* (DAC)

DAC adalah proses mengubah digital audio menjadi sinyal analog. DAC biasanya hanya menerima sinyal digital Pulse Code Modulation (PCM). PCM adalah representasi digital dari sinyal analog, dimana gelombang disampel secara beraturan berdasarkan interval waktu tertentu, yang kemudian akan diubah ke biner. Proses pengubahan ke biner disebut Quantisasi. PCM ditemukan oleh insinyur dari Inggris, bernama Alec Revees pada tahun 1937. Contoh DAC adalah: *soundcard, CDPlayer, IPod, mp3player*.

## 2.3 TEORI MUSIK

Musik adalah bunyi yang diterima oleh individu dan berbeda-beda berdasarkan sejarah, lokasi, budaya dan selera seseorang [3]. Definisi sejati tentang musik juga bermacam-macam:

- Bunyi yang dianggap enak oleh pendengarnya .
- Segala bunyi yang dihasilkan secara sengaja oleh seseorang atau kumpulan dan disajikan sebagai musik .

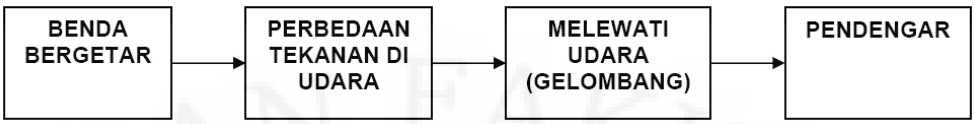
Beberapa orang menganggap musik tidak berwujud sama sekali. Musik menurut Aristoteles mempunyai kemampuan mendamaikan hati yang gundah, mempunyai terapi rekreatif dan menumbuhkan jiwa patriotisme.

Musik juga memiliki abjad, disebut sebagai tangga nada (*scale*). Setiap nada identik dengan huruf yang nantinya bersama-sama membentuk *chord*. Kemudian *chord* bersama-sama membentuk frasa (kalimat musik). Sekumpulan frasa yang baik membentuk lagu yang dapat dinyanyikan. Jadi *chord* adalah kosa kata. Kosakata saja tidak cukup, kosakata harus mampu membentuk kata yang bermakna ketika diucapkan, dan akhirnya membentuk kalimat yang baik yang dapat dimengerti oleh orang lain [4].

Musik adalah gelombang sinusoidal yang frekuensi dan besar tengangannya tidak konstan melainkan naik turun sesuai dengan alunan musiknya. Tegangan ini bisa negatif dan bisa juga positif. Standard pengukuran spesifikasi rating daya keluaran sistem audio adalah dengan inputan sinyal sinusoidal. Dengan menggunakan frekuensi pada rentang 20 Hz - 20 KHz.

### 2.3.1 Suara (*Sound*)

Suara adalah suatu fenomena fisik yang dihasilkan oleh getaran benda, atau getaran suatu benda yang berupa sinyal analog dengan amplitudo yang berubah secara kontinyu terhadap waktu [5]. Proses perambatan suara dapat dilihat pada Gambar 2.3:




Gambar 2.3 Proses perambatan suara

Suara berhubungan erat dengan rasa “mendengar”. Suara atau bunyi biasanya merambat melalui udara. Suara atau bunyi tidak bisa merambat melalui ruang hampa. Gelombang suara terdiri dari beberapa gelomang yang frekuensinya berbeda-beda. Konsep dasarnya, suara dihasilkan oleh getaran suatu benda. Selama bergetar, perbedaan tekanan terjadi di udara sekitarnya. Pola osilasi yang terjadi dinamakan sebagai “GELOMBANG”. Gelombang mempunyai pola sama yang berulang pada interval tertentu, yang disebut sebagai “PERIODE”. Contoh suara periodik adalah: instrumen musik, nyanyian burung, dll. Contoh suara nonperiodik adalah: batuk, percikan ombak, dll.

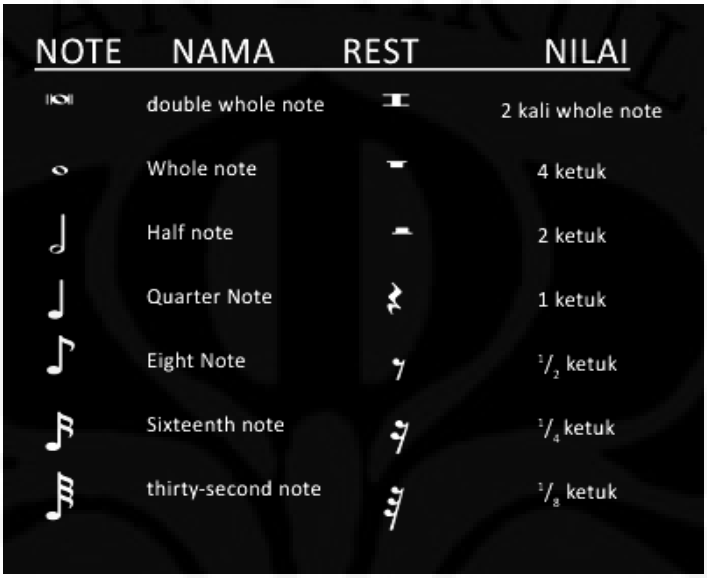
### 2.3.2 Nada Dan Nilainya

Nada adalah bunyi yang beraturan, yaitu memiliki frekuensi tunggal tertentu [3]. Dalam teori musik, setiap nada memiliki tinggi nada atau tala tertentu menurut frekuensinya ataupun menurut jarak relatif tinggi nada tersebut terhadap tinggi nada patokan. Nada dasar suatu karya musik menentukan frekuensi tiap nada dalam karya tersebut. Nada dapat diatur dalam tangga nada yang berbeda-beda. Istilah "nada" sering dipertukarkan penggunaannya dengan "not", walaupun kedua istilah tersebut memiliki perbedaan arti.

Dalam bahasa musik, dikenal partitur dan salah satu bagiannya adalah not, yang menggambarkan nada yang diperdengarkan. Not sendiri memiliki bagian yang disebut *flag*, *stem*, dan *note head*. Untuk lebih jelasnya, gambarnya dapat dilihat pada Gambar 2.4 berikut:.


Gambar 2.4 Sebuah not

Nilai not beserta detailnya dapat dilihat pada Gambar 2.5 di bawah ini.




Gambar 2.5 Not dan nilainya

### 2.3.3 Tanda Kromatis

Tanda kromatis atau tanda *Accidentals* adalah tanda-tanda untuk menaikkan, menurunkan, dan mengembalikan nada-nada yang dinaikkan atau diturunkan ke nada semula. Tanda kromatis ini ditulis di depan not yang akan diturunkan atau dinaikkan. Tanda kromatis dapat dilihat dari Tabel 2.1 berikut:

Tabel 2.1 Tangga Nada Kromatik dalam Musik

| | |
|---|---|
|  | *sharp* (kres)<br>untuk menaikkan 1/2 nada (*semitone*) |
|  | *double sharp* (kres ganda)<br>untuk menaikkan 1 nada (*tone*) |
|  | *flat* (mol)<br>untuk menurunkan 1/2 nada |
|  | *double flat* (mol ganda)<br>untuk menurunkan 1 nada |
|  | *natural* (tanda pugar)<br>untuk mengembalikan nada |

Nada-nada yang dinaikkan setengah jarak, mendapat sebutan is (C menjadi Cis). Nada-nada yang diturunkan setengah jarak, mendapat sebutan Es (B menjadi Bes). Nada-nada yang mendapat tanda pugar, kembali ke nada semula (Cis menjadi C, Bes menjadi B).

### 2.3.4 Tangga Nada

Dalam seni musik dikenal ada istilah tangga nada. Tangga nada berisikan kumpulan nada-nada yang harmonis. Keharmonisannya terjadi karena ada ‘aturan’ dibalik itu semua.

#### *2.3.4.1 Tangga Nada Kromatik*

Kumpulan dari semua nada dalam musik disebut sebagai tangga nada kromatik [6]. Kromatik merupakan sebuah nama berasal dari bahasa Yunani : *chrôma*, yang artinya warna. Dalam hal ini tangga nada kromatik berarti “nada dari tiap warna”. Sama seperti warna cahaya menyatakan frekuensi yang berbeda-beda maka demikian juga dengan nada. Karena nada selalu berulang untuk tiap oktaf yang ada, maka istilah ‘tangga nada kromatik’ sering dipakai untuk ke-12 nada dari tiap oktaf.

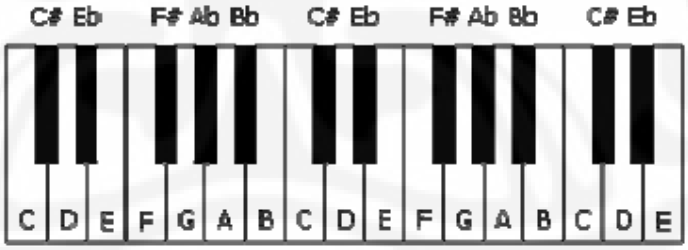
Perbedaan antara 2 buah *pitch* (nada) yang berdekatan disebut sebagai *semitone*. Meskipun ada 12 nada dalam 1 oktaf, tapi hanya 7 huruf pertama dari abjad yang dipakai untuk memberi nama pada nada, yaitu dari A sampai G. Kelima nada yang lain dalam 1 atau tanda *mol* tangga nada kromatik diberi nama dengan menempatkan tanda *kres* (#) ata tanda *mol* (b) setelah notasi nada.

Tabel 2.2 berikut ini menunjukkan frekuensi dari ke-12 nada antara nada A pada 440 Hz dan nada A satu oktaf di atasnya.

Tabel 2.2 Tangga Nada Kromatik dalam Musik

| Tangga Nada Kromatik | |
|---|---|
| A | 440.00 Hz |
| A# / Bb | 466.16 Hz |
| B | 493.88 Hz |
| C | 523.25 Hz |
| C# / Db | 554.37 Hz |
| D | 587.33 Hz |
| D# / Eb | 622.25 Hz |
| E | 659.25 Hz |
| F | 698.46 Hz |
| F# / Gb | 739.99 Hz |
| G | 783.99 Hz |
| G# / Ab | 830.61 Hz |
| A | 880.00 Hz |

Untuk piano, visualisasinya dapat dilihat pada Gambar 2.6 berikut ini




Gambar 2.6 Visualisasi tangga nada kromatik

Karena dalam tangga nada kromatik ada 12 nada, maka dapat dibuat berbagai tangga nada dengan membuat suatu susunan kombinasi dari nada-nada tersebut.

*2.3.4.2 Tangga Nada Mayor*

Tangga nada mayor adalah tangga nada yang sangat umum dipakai untuk musik Barat (*western*). Ketika dimainkan secara berurutan tangga nada mayor ini dikenal dengan istilah : do-re-mi-fa-so-la-si-do. Pengucapan doremisasi berasal dari teks doa Sancto Iohannes : ***Ut*** *queant laxis,* ***Re****sonare fibris,* ***Mi****ra gestorum,* ***Fa****muli tuorum,* ***Sol****ve polluti,* ***La****bii reatum* [7]. Jarak 2 nada berdekatan disebut *semitone*, dan dua buah *semitone* disebut *tone*. Tangga nada ini disusun berdasarkan kekhususan aturan, yaitu kombinasi *interval semitone* antara nada-nada yang ada. Aturannya adalah :

2 (*tone*)
2 (*tone*)
1 (*semitone*)
2 (*tone*)
2 (*tone*)
2 (*tone*)
1 (*semitone*)

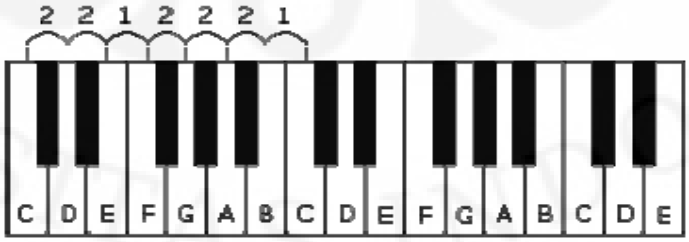
atau, ada juga yang menuliskan seperti di bawah ini ( contoh tangga nada C )

| C | - | D | - | E | - | F | - | G | - | A | - | B | - | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | | 1 | | ½ | | 1 | | 1 | | 1 | | ½ | |

Dalam paranada, ditampilkan seperti pada Gambar 2.7 berikut:

Gambar 2.7 Visualisasi tangga nada mayor dalam paranada

Dalam piano, ditampilkan seperti pada Gambar 2.8 berikut:




Gambar 2.8 Visualisasi tangga nada mayor dengan piano

### 2.3.5 *Interval*

*Interval* adalah jarak atau selang antara dua buah nada [8,9]. Ada banyak ukuran *interval* yang dapat dibuat untuk membuat tangga nada. Dalam subbab sebelumnya *interval* yang dipakai (dalam satuan *semitone*) untuk membentuk tangga nada mayor adalah : 2-2-1-2-2-2-1 [8]. Tangga nada tidak hanya tangga nada mayor. Ada banyak jenis tangga nada yang lain. Dan itu semua dibuat berdasarkan aturan terhadap *interval*. Ukuran *interval* yang bervariasi akan memberikan tidak hanya suara terdengar yang berbeda tapi juga memberi kesan 'rasa' yang juga berbeda. Untuk ukuran *interval* tertentu campuran nada dapat 'dirasakan' begitu 'pas' atau cocok (*consonant*). Tetapi campuran nada yang lain bisa saja kedengarannya rada kurang 'enak' atau kurang cocok (*dissonant*). Kombinasi *consonant* dan *dissonant* sangat diperlukan dalam musik. Musik yang hanya berisi *consonant* akan terdengar lembut dan lunak. Pemberian *dissonant* membuat ada 'tekstur' dalam musik.

Berikut ini diberikan Tabel 2.3 mengenai ukuran *interval* (dalam *semitone*) dan namanya serta penjelasan singkat. Untuk kemudahan dalam istilah, maka bagian ini ditulis dalam bahasa Inggris.

Tabel 2.3 *Interval* pada musik

| Interval | Ukuran | Keterangan |
|---|---|---|
| **unison** | 0 | Two identical notes played together are always strongly consonant, and difficult to tell apart. |
| **minor second** | 1 | Strongly dissonant, with a warbling sound in the background, as if the two notes are fighting with each other. |
| **second** | 2 | Less dissonant, but the notes still do not sit completely at ease with each other. |
| **minor third** | 3 | Strongly consonant, with a melancholy flavour to the sound. Forms the basis of minor chords and scales. |
| **major third** | 4 | Strongly consonant, making a stable and pleasing sound. Forms the basis of major chords and scales. |
| **perfect fourth** | 5 | Mildly dissonant, with a stretched feeling as if it would rather return to a major third. |
| **tritone** | 6 | Dissonant, and often found in chords of four notes or more, where it adds a particular harmonic "spice". |
| **perfect fifth** | 7 | Strongly consonant, and found in both minor and major chords. It adds solidness, but not much character to the harmony. |
| **minor sixth** | 8 | Mildly dissonant. |
| **major sixth** | 9 | Consonant. |
| **minor seventh** | 10 | Mildly dissonant. |
| **major seventh** | 11 | Dissonant. |
| **octave** | 12 | Strongly consonant, like unison, because notes an octave apart sound the same, just higher or lower. |

Nama dan ukuran *interval* pada Tabel 2.3 sangat penting, khususnya ketika membuat sebuah *chord*. *Chord* dibuat dengan mengkombinasikan *interval* yang ada dan memainkan tiga atau lebih nada secara serentak. Ada banyak jenis *chord*, dan masing-masing dibuat berdasarkan formula yang unik dari pengkombinasian *interval* nada. Tiap *interval* diukur dari nada awal (dikenal sebagai nada dasar) dari tipe *chord* yang akan disusun. Misalkan ingin membuat *chord* D, maka nada D dihitung sebagai nada awal (dasar) dalam perhitungan *interval.*

*Interval* dari tipe *chord* yang disusun dari nada dasar dikenal sebagai tingkat (*degree*). Penamaan tingkat ini mirip dengan nama *interval*, hanya tingkat ini lebih sering dipakai karena penulisannya yang singkat. Berikut ini pada Tabel 2.4 diberikan daftar penamaan tingkat dan ekivalennya dengan *interval* dan ukuran *interval* (dalam satuan *semitone*). Untuk tetap jelas, ditulis dalam bahasa Inggris.

Tabel 2.4 Tingkat *interval*

| Tingkat | Nama | Interval | Ukuran |
|---|---|---|---|
| **1** | root (tonic) | unison | 0 |
| **2** | second | major second | 2 |
| **b3** | flat third | minor third | 3 |
| **3** | third | major third | 4 |
| **4** | fourth | perfect fourth | 5 |
| **b5** | flat fifth | tritone | 6 |
| **5** | fifth | perfect fifth | 7 |
| **#5** | sharp fifth | augmented fifth | 8 |
| **6** | sixth | major sixth | 9 |
| **b7** | flat seventh | minor seventh | 10 |
| **7** | seventh | major seventh | 11 |
| **b9** | flat ninth | minor ninth | 13 |
| **9** | ninth | major ninth | 14 |
| **#9** | sharp ninth | augmented ninth | 15 |
| **11** | eleventh | eleventh | 17 |
| **#11** | sharp eleventh | augmented eleventh | 18 |
| **b13** | flat thirteenth | minor thirteenth | 20 |
| **13** | thirteenth | thirteenth | 21 |

Sangatlah penting mengingat tingkat pada Tabel 2.4, karena ini sangat berkaitan dalam penyusunan sebuah *chord*. Misalkan saja, untuk membuat *chord* mayor diperlukan nada dari tingkat 1, 3, dan 5. Maka jika ingin menyusun *chord* C mayor, nada yang dimainkan adalah nada C, E, dan G (tingkat 1, 3, dan 5 dari tangga nada C mayor).

### 2.3.6 Pasangan *Chord*

Dalam mengiringi sebuah lagu maka jenis *chord* yang dimainkan tentu juga tergantung dari nada dasar lagu tersebut. Selain itu, untuk menentukan pasangan *chord* yang dipakai untuk mengiringi lagu juga memiliki aturan tersendiri.

Untuk memudahkan pencarian formulanya, ada baiknya dimulai dengan menganalisis dari sistem tangga nada C mayor.

Tangga nada C mayor : C – D – E – F – G – A – B – C

Masing-masing nada dalam tangga nada C mayor ini dicari *triad*-nya, maka [10] :

Triad : C – E - G
Tingkat : 1 – 3 – 5 ➔ C mayor

Triad : D – F – A
Tingkat : 1 – b3 – 5 ➔ D minor

Triad : E – G – B
Tingkat : 1 – b3 – 5 ➔ E minor

Triad : F – A – C
Tingkat : 1 – 3 – 5 ➔ F mayor

Triad : G – B – D
Tingkat : 1 – 3 – 5 ➔ G mayor

Triad : A – C – E

Tingkat : 1 – b3 – 5 ➔ A minor

Triad : B – D – F
Tingkat : 1 – b3 – b5 ➔ B diminished

Triad : C – E – G
Tingkat : 1 – 3 – 5 ➔ C mayor

Dari penurunan di atas terdapat 3 buah triad mayor yaitu pada tingkat 1, 4, dan 5 relatif terhadap tangga nada C mayor. Ini berarti bila mengiringi dari C mayor, pasangan *chord* mayor lainnya adalah F mayor dan G mayor (biasanya cukup ditulis dengan F dan G saja). Kemudian didapati juga ada 3 buah *chord* minor, yaitu pada tingkat 2,3, dan 6 relatif terhadap tangga nada C mayor. Ini juga berarti pasangan *chord* lainnya yang dapat mengiringi *chord* C mayor adalah : D minor (Dm), E minor (Em), dan A minor (Am). Kemudian terdapat satu *chord* diminished, yaitu B diminis (Bdim).

Bagian ini sering digolongkan seperti pada Tabel 2.5 berikut :

Tabel 2.5 Penggolongan Triad

| Tingkat | Nama |
|---|---|
| I | Tonic |
| II | Supertonic |
| III | Mediant |
| IV | Subdominant |
| V | Dominant |
| VI | Submediant |
| VII | Leading Tone |

### 2.3.7 *Pitch*

*Pitch* dapat dikatakan sebagai tinggi rendahnya nada. Sebuah not A yang dimainkan diatas C4 (*middle C*) memiliki frekuensi *pitch* sebesar 440 Hz. Perubahan yang sedikit tidak mempengaruhi secara signifikan dari *tone* yang dimainkan. Perubahan yang masih diijinkan sebesar lima persen. Namun

perubahan tersebut akan terasa signifikan bila dimainkan secara bersama-sama dengan nada yang lainnya.

*Pitch* bergantung pada amplitudo dari suara. Hal ini sangat penting untuk suara-suara berfrekuensi rendah. Contohnya suara bass akan terdengar lebih rendah. Seperti yang sudah lama dialami oleh banyak orang, dimana persepsi dari *pitch* dapat 'kabur' contohnya ilusi pada audio dan *tritone paradox*.

*Pitch* seringkali diberi label dengan menggunakkan cara *scientific* yang merupakan kombinasi dari huruf dan angka dimana mereperesentasikan frekuensi fundamental, contohnya A4 atau A440. Namun hal ini menimbulkan 2 masalah yaitu dalam standarisasi Barat penulisan G4## memiliki penulisan *pitch* yang sama dengan A4. Kedua, persepsi manusia terhadap *pitch* adalah logaritmik, sehingga hal yang diterima ketika mendengar A220 dan A440 adalah sama ketika mendengar A440 dan A880.

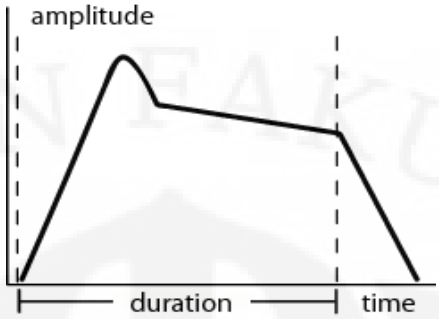
Untuk menghindari hal tersebut, para ilmuwan berusaha merepresentasikan *pitch* dari nada-nada yang ada dengan menggunakan *numerical scale*. Cara ini berdasarkan pada logaritma dari frekuensi fundamental masing-masing nada. Contohnya, dalam mengadaptasi bunyi *pitch* ke dalam sistem MIDI dapat dilakukan konversi sebagai berikut :

$$p = 69 + 12 \times \log_2 \left( \frac{f}{440 \text{ Hz}} \right) \tag{2.2}$$

Hal ini membentuk penomoran yang linear untuk 12 nada dalam 1 oktaf. Contohnya untuk A440 memliki nomor *pitch* 69. Sistem ini bersifat fleksibel dimana dapat mengikutsertakan "*microtones*" yang tidak ditemukan dalam keyboard yang standar.

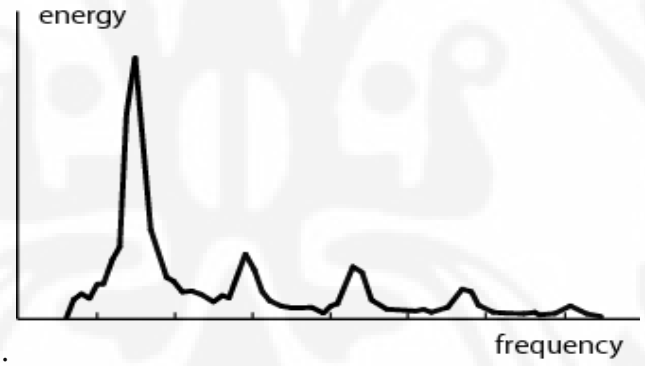
### 2.3.8 Model Nada

Untuk piano dan alat musik petik dan gesek, representasi nada memiliki profil energi yang sama. Nada tersebut mulai dengan sebuah *attack* dengan *overshoot* yang cukup tinggi dan berakhir dengan keadaan *sustain* dan akhirnya menurun [3]. Panjang dari *attack* dan *sustain* akan menentukan durasi dari nada. Hal tersebut dapat dilihat seperti pada Gambar 2.9 berikut:




Gambar 2.9 Grafik dari amplitudo energi dalam domain waktu untuk nada *single*

Karakteristik spectrum untuk tiap nada bervariasi bergantung pada instrumen musik yang dimainkan. Gambar 2.10 menunjukkan puncak tertinggi merupakan frekuensi fundamental dimana terletak pada frekuensi yang rendah. Puncak-puncak yang lainnya merupakan harmonik dari frekuensi fundamental yang muncul pada saat n * f, dimana n adalah nilai integer lebih besar dari satu




Gambar 2.10 Visualisasi nada pada domain frekuensi

### 2.3.9 BPM (*Beat Per Minute*)

BPM adalah berapa banyak beat dalam satu menit. Jadi jika kita mendengar temponya 120, atau 90, itu berarti ada 120 beat dalam satu menit, atau ada 90 beat dalam satu menit. [3]

Pengertian beat adalah ketukan untuk menghitung durasi musik dimana acuannya adalah jumlah pukulan. Pengertian tempo adalah kecepatan lagu dimana merupakan hasil dari jumlah beat berbanding terbalik dengan satuan waktu

menit(M) dimana jumlah tempo= beat(B)/menit(M)...(B/M = BPM = beats per minute)

### 2.3.10 Oktaf

Dalam musik, satu oktaf (kadang disingkat menjadi 8ve) adalah interval antara suatu not dengan not lain dengan frekuensi dua kalinya [3]. Perbandingan frekuensi antara dua not yang terpisah oleh interval satu oktaf adalah 2:1.

# BAB 3

# PERANGKAT LUNAK KOMPOSER MUSIK

## 3.1 GAMBARAN UMUM PERANGKAT LUNAK

Perangkat lunak komposer musik adalah perangkat lunak untuk membuat suatu musik sendiri sesuai dengan keinginan kita sendiri. Tampilan dari perangkat lunak komposer musik dapat dilihat dari Gambar 3.1 berikut:


Gambar 3.1 Tampilan perangkat lunak komposer musik

Suara di *text box* yang tersedia adalah

1 Piano

Suara alat musik piano.

2 Chord

Gabungan dari 3 nada suara piano.

3 Bass

Suara alat musik gitar bass.

4 String

Suara alat musik string (gabungan banyak biola).

5 Sinwave

Suara gelombang sinus, untuk melodi lagu.

Batasan suara yang tersedia adalah:

1. Suara piano tersedia dari setengah ketuk (not seperdelapan) sampai 4 ketuk (not penuh), sebanyak 3 oktaf.
2. Suara bass tersedia dari setengah ketuk (not seperdelapan) sampai 4 ketuk (not penuh), sebanyak 2 oktaf.
3. Suara string tersedia dari setengah ketuk (not seperdelapan) sampai 4 ketuk (not penuh), sebanyak 3 oktaf.
4. Suara sinwave tersedia dari setengah ketuk (not seperdelapan) sampai 4 ketuk (not penuh),sebanyak 2 oktaf.
5. No (istirahat ) tersedia dari setengah ketuk (not seperdelapan) sampai 4 ketuk (not penuh).

## 3.2 FASILITAS DAN PENGGUNAAN PERANGKAT LUNAK

Fasilitas yang tersedia dalam perangkat lunak ini adalah: pengaturan tempo, pengaturan *volume* baik untuk tiap instrumen maupun *volume* utama (*master volume*), *text box* untuk pengisian nada lagu, pembersihan teks baik untuk tiap instrumen maupun teks keseluruhan, grafik sinyal lagu, mute dan solo untuk tiap instrumen, *play*, *stop*, *help*, dan *close.*

Cara pengisian *text box* untuk menyusun lagu yang digunakan cukup mudah. Penulisan terdiri dari 3 bagian, yaitu: nada, durasi, dan oktaf.

1. Nada

Penulisan huruf kecil berarti membunyikan nada yang sesuai seperti yang ditulis. Penulisan huruf besar berarti membunyikan nada *kres*, yaitu setengah nada lebih tinggi dari yang ditulis.

2. Durasi

   Angka 1 adalah not seperdelapan (setengah ketuk).

   Angka 2 adalah not seperempat (satu ketuk).

   Angka 3 adalah not setengah (dua ketuk).

   Angka 4 adalah not penuh (empat ketuk).

3. Oktaf

   Angka 1 adalah oktaf pertama.

   Angka 2 adalah oktaf kedua.

   Angka 3 adalah oktaf ketiga .

Untuk tanda istirahat, penulisannnya hanya dua bagian, yaitu no dan durasi.

Contoh penulisan di *text box*:

1. Penulisan c11 berarti memainkan nada c dengan panjang 1 (dalam istilah musik disebut not seperdelapan atau sebanyak setengah ketuk) dan di oktaf 1.
2. Penulisan C32 berarti memainkan nada cis dengan panjang 3 (dalam istilah musik disebut not setengah atau sebanyak 2 ketuk) dan di oktaf 2.
3. Penulisan no1 berarti tidak memainkan nada sebanyak setengah ketuk (dalam istilah musik, disebut tanda istirahat).

Tempo yang tersedia dalam perangkat lunak ini adalah 100 BPM, 120 BPM, 140 BPM, 160 BPM, dan 180 BPM. Dengan kelima tempo ini, dapat terwakili banyak tempo-tempo yang ada, yaitu dengan melihat kelipatan tempo tersebut. Misalnya diinginkan tempo 200 BPM, yaitu sama dengan 2 kali tempo 100 BPM. Atau tempo 90 BPM, yaitu sama dengan setengah tempo 180 BPM.

### 3.3 DIAGRAM ALIR PERANGKAT LUNAK KOMPOSER MUSIK

Diagram alir perangkat lunak komposer musik dapat dilihat seperti pada gambar 3.2 dibawah ini:


Gambar 3.2 Diagram alir perangkat lunak komposer musik

### 3.3.1 Membaca Sumber Suara

Sumber suara yang digunakan berasal dari hasil rekaman sendiri. Digunakan sumber suara asli agar menghasilkan suara musik yang lebih nyata. Tiap nada direkam satu per satu dan disimpan dalam format WAV. Dalam merekam nada, digunakan *sofware* bantuan, yaitu NUENDO. Semua data direkam dengan tempo dasar 100 BPM. Maka dihasilkan sebuah *data base* suara yang siap untuk diolah menjadi sebuah lagu seperti yang diinginkan. Sumber suara tersebut diolah dengan MATLAB. Karena sumber suara berformat WAV, maka untuk membaca sumber suara tersebut di gunakan perintah wavread. Contohnya:

```
C1=wavread('piano c1 100.wav');
```

Maka C1 adalah sebuah matriks. Matriks ini terdiri dari 1 kolom dan banyak baris. Tetapi, karena pada gelombang sinus matriksnya terdiri dari 1 baris dan banyak kolom, maka matriks WAV perlu di *transpose.*

```
c1=C1'
```

Untuk suara gelombang sinus, tidak diambil dari data base, tetapi dibangkitkan. Jumlah baris matriks WAV terpendek adalah 98490. Frekuensi samping yang digunakan adalah 44100 Hz. Jadi Ts = 1/44100 s. Maka waktu untuk gelombang sinus adalah : t = 98490 * Ts = 2.23

```
Fs = 44100;
Ts = 1/Fs;
x4 = [0:Ts:t];
swc41=sin(2*pi*523.25*x4);
```

Tujuan dipilih matriks WAV dengan baris terpendek adalah agar semua matriks dapat terbaca dalam perintah (`1 : panjang kolom`). Penjelasan mengenai hal ini akan dibahas lebih lanjut dalam pembahasan 3.3.2.

### 3.3.2 Membaca Panjang Nada Dan Tempo

Untuk mengatur panjang nada dan tempo, dapat dilakukan dengan memanipulasi jumlah kolom yang akan dimainkan. Untuk 100 BPM, pengaturan panjang nada dapat dilakukan dengan program:

```
panjang1 = fix(baris/8); panjang2 = fix(baris/4); panjang3 = fix(baris/2); panjang4 = fix(baris);
```

Panjang 1 adalah not seperdelapan, panjang 2 adalah not seperempat, panjang 3 adalah not setengah, panjang 4 adalah not penuh. Perintah `fix` digunakan untuk

pembulatan (terdekat menuju nol). Pada bagian 3.3.1 telah disebutkan bahwa matriks yang pilih adalah yang mempunyai baris terpendek. Hal ini bertujuan agar pada program

```
c11=c1(1:panjang1);  c21=c1(1:panjang2);  c31=c1(1:panjang3);
c41=c1(1:panjang4);
```

seluruh matriks menjadi dapat terbaca. Apabila yang dipilih adalah kolom terpanjang, maka matriks yang mempunyai kolom terpendek tidak memiliki panjang kolom yang cukup untuk dibaca.

Untuk gelombang sinus, pengaturan panjang nada dillakukan dengan program:

```
x1 = [0:Ts:t/8]; x2 = [0:Ts:t/4]; x3 = [0:Ts:t/2]; x4 = [0:Ts:t];
```

x1 adalah not seperdelapan, x2 adalah not seperempat, x3 adalah not setengah, x4 adalah not penuh. Setelah itu dibuat program:

```
swc11=sin(2*pi*523.25*x1);swc21=sin(2*pi*523.25*x2);swc31=sin(2*pi
*523.25*x3);swc41=sin(2*pi*523.25*x4);
```

Prinsip dasarnya adalah sama seperti yang dilakukan pada matriks WAV.

Untuk mengatur tempo, prinsip dasarnya juga sama dengan mengatur panjang nada, yaitu dengan memanipulasi jumlah kolom yang akan dimainkan. Sebagai contoh, berikut adalah program manipulasi untuk BPM =120:

```
panjang1 = fix(baris/(8*1.2)); panjang2 = fix(baris/(4*1.2));
panjang3 = fix(baris/(2*1.2)); panjang4 = fix(baris/1.2); t =
t/1.2;
```

Matriks awal dikali dengan 100/120, atau sama saja dengan dibagi 1.2. Maka akan dihasilkan matriks dengan tempo 120 BPM. Cara ini juga berlaku untuk tempo tempo yang lainnya (140 BPM, 160 BPM, 180 BPM).

### 3.3.3 Membaca *Text Box*

Terdapat 7 *Text Box* yang terdiri dari piano, chord1, chord2, chord3, bass, string, dan sinwave. Setelah *user* menuliskan nada-nada pada *text box*, maka tulisan tersebut akan diterjemahkan menjadi sebuah susunan matriks. Berikut program untuk menerjemahkan tulisan dari *text box* :

```
ep=findobj('tag','edit_piano');
kode=get(ep,'String'); p=size(kode); panjang=p(2); piano=[0];
```

```
for i=1:1:panjang
    % cari karakter yang ditulis pada text box
    karakter=kode(i);
    if (karakter=='c' || karakter=='C'|| karakter=='d'||
karakter=='D' || karakter=='e'|| karakter=='f'|| karakter=='F'||
karakter=='g'|| karakter=='G'|| karakter=='a'|| karakter=='A'||
karakter=='b'|| karakter=='n')
       % cari kode yang diikuti angka
       karakter=kode(i:i+2);
       if (karakter=='c11'); piano=[piano,c11]; end
```

(dan seterusnya.....)

Pada program tersebut, tulisan yang diterjemahkan adalah pada *text box* piano. Pertama-tama, perintah `findobj` berfungsi unuk mencari objek dengan tag yang bernama edit_piano. Setelah tag edit_piano didapat, tulisan (string) pada edit_piano didapat dengan perintah `get` . Perintah `p= size(kode)` dan `panjang = p(2)` adalah untuk mengetahui panjang dari tulisan. Agar matriks piano mempunyai nilai awal, maka matriks piano pertama-tama dibuat bernilai nol. Perintah `for i= 1:1:panjang` dan `karakter=kode(i)` adalah pencarian karakter sepanjang seluruh tulisan . Jika ditemukan karakter c, C, d, D, e, f, F, g, G, a, A, b, n, maka akan dilihat kedua karakter setelahnya. Hal tersebut dilakukan dengan perintah : `karakter=kode(i:i+2)`, sebab panjang tulisan (string) yang perlu dibaca hanya 3 string. Perintah terakhir berarti jika karakter yang ditemukan adalah c11, maka isi matriks piano akan ditambahkan dengan matriks c11. Perintah ini terus dilakukan sampai pencarian semua karakter selesai.

Prinsip yang sama dilakukan untuk menerjemahkan tulisan pada *text box* chord1, chord2, chord3, bass, string, sinwave. Berikut adalah contoh lagu yang digunakan, yaitu lagu *twinkle-twinkle little star*:

```
Piano=c23,c23,g23,g23,a23,a23,g33,a23,a23,g23,g23,f23,f23,e33,g23,g21,f23,g2
      1,e23,g21,d23,g21,g23,g21,f23,g21,e23,g21,d23,g21,e23,e23,b23,b23,f23,
      f23,b33,a23,a23,g23,g23,f23,f23,e43
chord1=g31,b31,c32,b21,A21,a31,g31,f31,g31,c32,b31,g31,f31,c32,b31,g31,f31,
      g31,b31,c32,b21,A21,a31,g31,f31,g41
chord2=b31,d32,e32,d22,C22,c32,b31,G31,b31,e32,d32,c32,G31,e32,d32,c32,G3
      1,b31,d32,e32,d22,C22,c32,b31,G31,b41
```

```
chord3=e32,g32,a32,g22,F22,f32,e32,d32,e32,g32,f32,e32,d32,g32,f32,e32,d32,e
      32,g32,a32,g22,F22,f32,e32,d32,e42
bass=c32,e22,no1,e12,f32,e22,D22,d32,g22,no1,g11,g31,c32,g21,no1,g11,g31,g2
      1,no1,g11,g31,g21,no1,g11,g31,g21,no1,g11,g31,c32,e22,no1,e12,f32,e22,
      D22,d32,g22,no1,g11,g31,c42
string=c33,b32,a32,g32,a32,g32,f32,e32,g41,g41,g42,g42,e33,d33,c33,b32,a32,g3
      2,f32,e42
sinwave=c21,c21,g21,g21,a21,a21,g31,f21,f21,e21,e21,d21,d21,c31,g21,g21,f21,f
      21,e21,e21,d31,g21,g21,f21,f21,e21,e21,d31,c21,c21,g21,g21,a21,a21,g31
      ,f21,f21,e21,e21,d21,d21,c31
```

Setelah semua tulisan diterjemahkan, maka perlu dilakukan pencarian jumlah kolom instrumen terpanjang, yaitu dengan program:

```
pp  = size(piano); panjang_piano = pp(2);
pb  = size(bass); panjang_bass = pb(2);
pc1 = size(chord1); panjang_chord1 = pc1(2);
pc2 = size(chord2); panjang_chord2 = pc2(2);
pc3 = size(chord3); panjang_chord3 = pc3(2);
ps  = size(string); panjang_string = ps(2);
psw  = size(sinwave); panjang_sinwave = psw(2);
akhir= max([panjang_piano panjang_bass panjang_chord1
       panjang_chord2 panjang_chord3 panjang_string
       panjang_sinwave]);
```

Dengan mengetahui jumlah kolom terpanjang, maka semua jumlah kolom dapat disamakan sesuai dengan jumlah kolom yang terpanjang. Hal ini bertujuan agar tidak ada matriks yang terpotong ketika menjadi sebuah lagu.

### 3.3.4 Membuat FFT

Pembuatan FFT untuk masing-masing instrumen dilakukan dengan program:

```
a=fft(piano,akhir);
b=fft(bass,akhir);
c=fft(chord1+chord2+chord3,akhir);
d=fft(string,akhir);
e=fft(sinwave,akhir);
```

Pada bagian ini, chord1, chord2, chord3 disatukan. Seluruh instrumen di FFT sesuai dengan panjang kolom terpanjang (akhir). Maka, masing-masing

instrumen yang tadinya dalam domain waktu, maka sekarang berubah menjadi domain frekuensi.

### 3.3.5 Mengatur *Volume*

Pengaturan *volume* dapat dilakukan dengan mengatur *slider* atau *text box volume*. *Volume* dapat diatur berkisar 0 sampai 100. Pengaturan *volume* untuk masing-masing instrumen dilakukan dengan program:

```
volume_piano=findobj('tag','vol_piano');
piano_volume = str2num(get(volume_piano,'String'));
a=(piano_volume/100)*a;
volume_bass=findobj('tag','vol_bass');
bass_volume = str2num(get(volume_bass,'String'));
b=(bass_volume/100)*b;
volume_chord=findobj('tag','vol_chord');
chord_volume = str2num(get(volume_chord,'String'));
c=(chord_volume/100)*c;
volume_string=findobj('tag','vol_string');
string_volume = str2num(get(volume_string,'String'));
d=(string_volume/100)*d;
volume_sinwave=findobj('tag','vol_sinwave');
sinwave_volume = str2num(get(volume_sinwave,'String'));
e=(sinwave_volume/100)*e;
```

Pada program tersebut, pertama dilakukan pencarian objek yang dituju dengan perintah `findobj`. Kemudian tulisan pada *text box volume* diterjemahkan dengan perintah `str2num`, yaitu bentuk string dibaca menjadi bentuk numerik. Matriks-matriks yang telah dibuat sebenarnya berisi nilai-nilai amplitudo dari nada, maka keras lemahnya nada dapat diatur dengan perintah `a=(piano_volume/100)*a;` Perintah ini dilakukan untuk tiap instrumen.

Untuk pengaturan GUI *slider* dan *text box* nya, perlu ada program tambahan pada m-file GUI nya (musik.m). Untuk pengaturan slider, pada *execute on slider movement* perlu ditambahkan program:

```
nilai_slider = round(get(handles.slider_piano,'Value'));
handles.nilai_slider=nilai_slider;
set(handles.vol_piano,'string',handles.nilai_slider);
```

Sedangkan untuk pengaturan *text box volume*, pada fungsi *callback* nya ditambahkan program:

```
if nilai >= get(handles.slider_piano,'Min')&...
        nilai <=get(handles.slider_piano,'Max')
            set (handles.slider_piano,'Value',round(nilai));
```

```
else
    errordlg('range nilai slider 0 -100','Inputan Salah');
end
```

Untuk pengaturan master *volume*, akan dibahas lebih lanjut pada bagian 3.3.7, yaitu setelah seluruh matriks instrumen disatukan. Namun, prinsip dasarnya adalah sama dengan pengaturan *volume* masing-masing instrumen.

### 3.3.6 Mengatur *Mute* Dan *Solo*

*Mute* berarti tidak memainkan instrumen yang dipilih. Program untuk *mute* adalah:

```
mute_piano=findobj('tag','mute_piano');
if get(mute_piano,'value')==1; a=0; end
mute_bass=findobj('tag','mute_bass');
if get(mute_bass,'value')==1; b=0; end
mute_chord=findobj('tag','mute_chord');
if get(mute_chord,'value')==1; c=0; end
mute_string=findobj('tag','mute_string');
if get(mute_string,'value')==1; d=0; end
mute_sinwave=findobj('tag','mute_sinwave');
if get(mute_sinwave,'value')==1; e=0; end
```

Pada program tersebut, setelah dilakukan pencarian objek dengan perintah `findobj`, akan dilihat nilai dari masing-masing *mute* instrumen. Jika *mute* piano bernilai satu, maka matriks a akan bernilai 0. Hal yang sama juga dilakukan pada bass, chord, string, dan sinwave.

*Solo* berarti hanya memainkan instrumen yang dipilih. Program untuk *solo* adalah:

```
solo_piano=findobj('tag','solo_piano');
solo_bass=findobj('tag','solo_bass');
solo_chord=findobj('tag','solo_chord');
solo_string=findobj('tag','solo_string');
solo_sinwave=findobj('tag','solo_sinwave');
if (get(solo_piano,'value')==1 & get (solo_bass,'value')==0
& get(solo_chord,'value')==0 & get(solo_string,'value')==0 &
get(solo_sinwave,'value')==0)
    b=0; c=0; d=0; e=0;
end
```

Program *solo* lebih rumit jika dibandingkan dengan program *mute.* Pada program *mute* dimungkinkan untuk fokus pada satu suara saja, namun pada

program *solo* perlu diperhatikan keseluruhan suara. Contohnya, jika *solo* piano bernilai 1 , *solo* bass bernilai 0, *solo* chord bernilai 0, *solo* string bernilai 0, dan *solo* sinwave bernilai 0; maka nilai b, c, d, e bernilai 0. Hal ini harus dilakukan untuk seluruh kombinasi a, b, c, d, dan e. Jumlah kombinasinya adalah 30 buah.

### 3.3.7 Menggabungkan Suara Dan Membuat IFFT

Setelah semua suara telah di atur seperti yang dikehendaki, maka langkah berikutnya adalah menggabungkan suara-suara tersebut, yaitu dengan program:

```
f=a+b+c+d+e;
song=ifft(f);
```

Suara a, b, c, d, e dijumlahkan menjadi gabungan, yaitu suara f. Karena suara f masih dalam domain frekuensi, maka perlu dikembalikan ke domain waktu, yaitu dengan perintah `ifft(f)`. IFFT adalah singkatan dari *inverse fast fourier transform*. Suara akhir dinamakan song. Setelah song dibuat, maka master *volume* dapat diatur dengan program:

```
volume_master=findobj('tag','vol_master');
master_volume = str2num(get(volume_master,'String'));
song=(master_volume/100)*song;
```

Pengaturan master *volume* ini mempunyai prinsip dasar yang sama dengan pengaturan *volume* masing-masing instrumen.

### 3.3.8 Membunyikan Lagu Dan Memplot Grafik

Program MATLAB untuk membunyikan lagu dan memplot grafik adalah:

```
sound(song,Fs);
axes=findobj('tag','grafik');
time=[0:(akhir*Ts)/(akhir-1):akhir*Ts];
plot(time,song);
xlabel('Time(second)');
ylabel('Amplitude(Normalized Value)');
wavwrite(song,Fs,'song');
```

Perintah `time=[0:(akhir*Ts)/(akhir-1):akhir*Ts];` adalah perintah agar tampilan sumbu x dalam satuan detik. Sebelumnya, sumbu x adalah dalam satuan jumlah kolom. Sedangkan perintah x label dan y label adalah untuk menamakan

sumbu x dan y. Perintah terakhir adalah `wavwrite`, perintah ini berguna untuk membuat file WAV dari song yang telah dibuat.

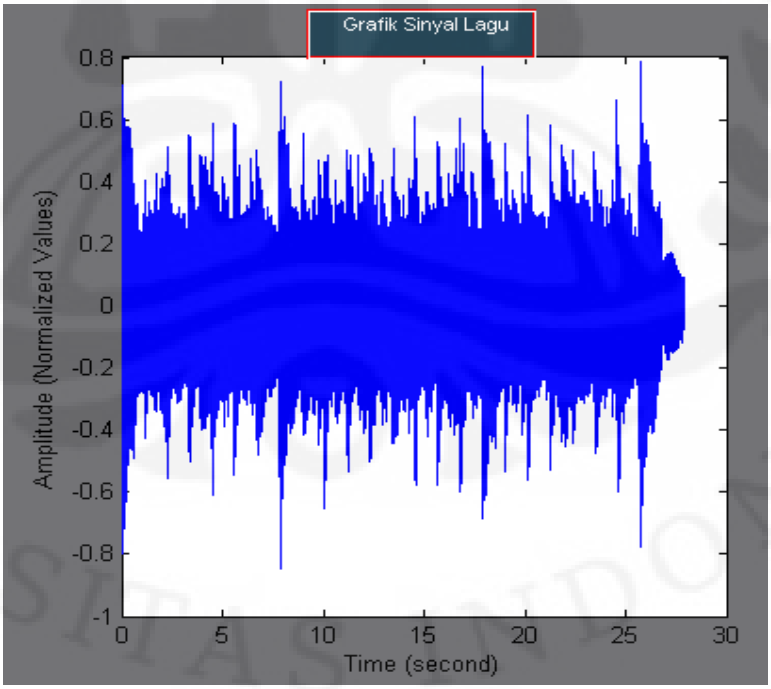
# BAB 4
# HASIL DAN ANALISIS
# PERANGKAT LUNAK KOMPOSER MUSIK

Pada skripsi ini digunakan contoh lagu *twinkle-twinkle little star*, karena lagu tersebut nadanya tidak rumit, durasinya tidak lama, dan mudah untuk diaplikasikan.

Parameter – parameter yang mempengaruhi dalam analisis adalah: tempo (bpm), volume (amplitudo), serta nada (frekuensi).

## 4.1 ANALISIS GRAFIK

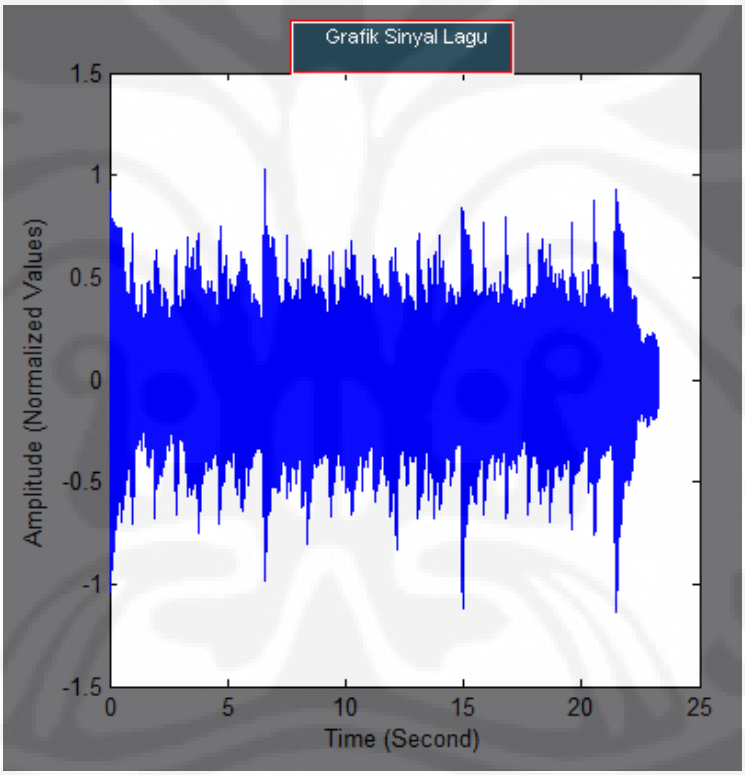
Pada Gambar 4.1 dibawah ini, diperlihatkan grafik dari lagu *twinkle-twinkle little star*. Tempo yang dipakai adalah: 100 BPM. Pengaturan *volume* yang dipakai adalah: *volume* piano = 100, *volume* chord = 100, *volume* bass = 100, *volume* string = 35, *volume* sinwave = 25, master *volume* = 50.




Gambar 4.1 Grafik lagu *twinkle twinkle little star*

Gambar 4.1 memperlihatkan grafik hasil perbandingan antara amplitudo terhadap waktu. Amplitudo dalam satuan skala normal (*normalized value*), sedangkan waktu dalam waktu detik. Pada Gambar 4.1 terlihat bahwa dalam tempo 100 BPM, lamanya lagu sekitar 27 detik dan amplitudo maksimum sekitar 0,8 skala normal.

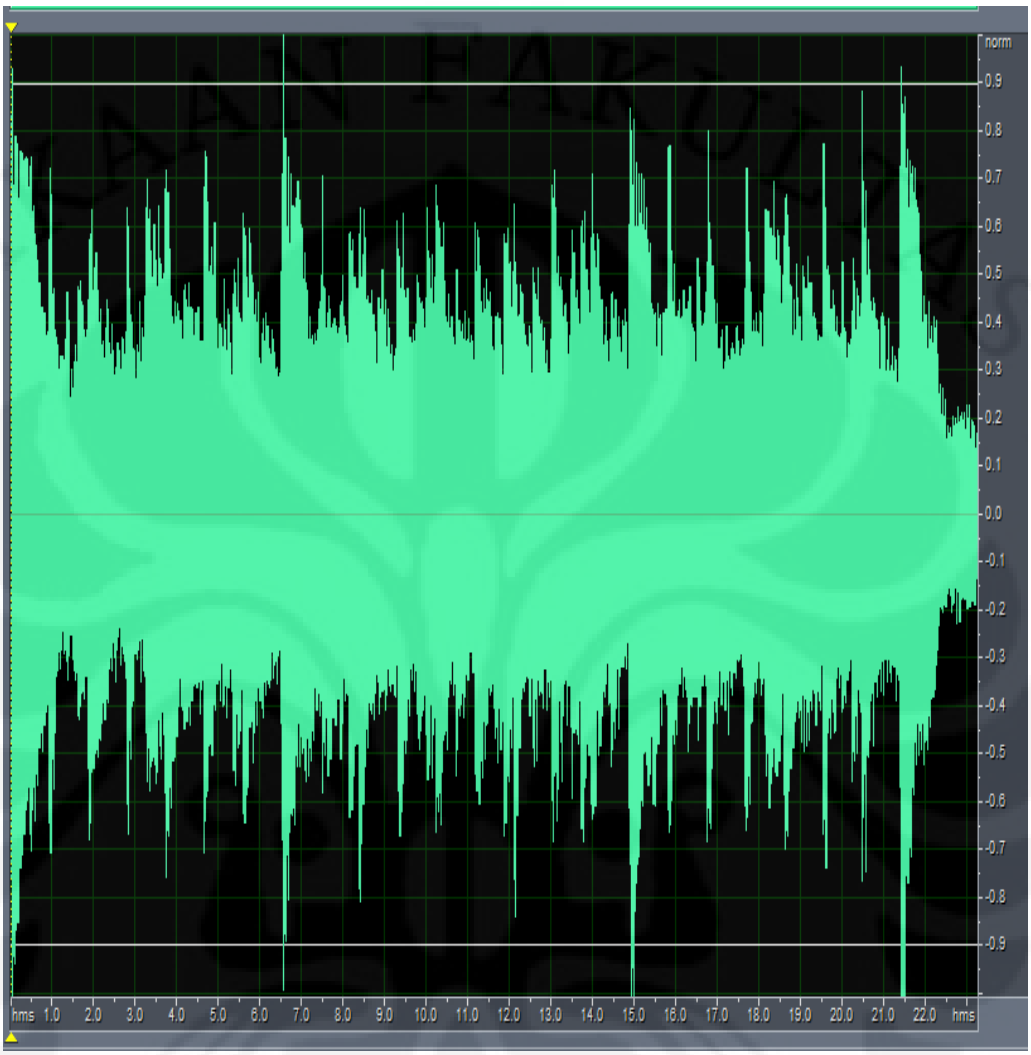
Jika dilakukan perubahan terhadap tempo dan master *volume*, misalnya tempo diubah menjadi 120 BPM, dan master *volume* dinaikkan menjadi 65, maka grafik akan menjadi seperti pada Gambar 4.2:




Gambar 4.2 Grafik lagu *twinkle twinkle little star* setelah diubah

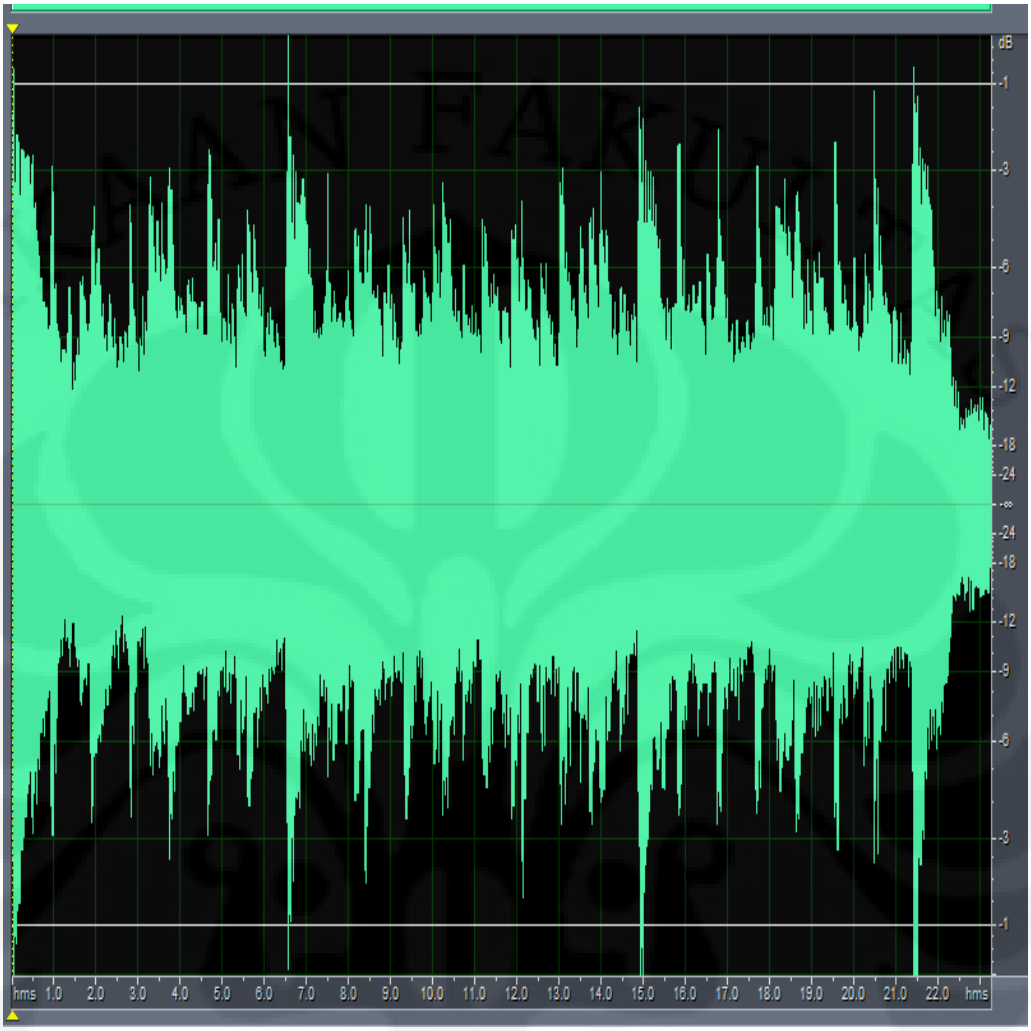
Pada Gambar 4.2, terlihat bahwa dengan tempo 120 BPM, maka lagu menjadi semakin cepat, yaitu sekitar 22 detik. Dengan perubahan master *volume* menjadi 65, maka amplitudo maksimum juga naik menjadi sekitar 1 skala normal.

Selain itu, hasil yang serupa didapat ketika digunakan software *Cool Edit Pro 2.0*, seperti ditunjukkan pada Gambar 4.3.




Gambar 4.3. Hasil dari *Cool Edit Pro 2.0*

Selain dalam satuan skala normal, satuan amplitudo dapat diubah ke dalam satuan desibel dengan rumus: $dB = 20 \log_{10} |\text{skala normal}|$ . Hasilnya dapat dilihat seperti pada Gambar 4.4.




Gambar 4.4 Hasil Cool Edit Pro 2.0 setelah diubah ke satuan desibel.

## 4.2 ANALISIS MENGGABUNGKAN SUARA DENGAN PEMBATASAN JUMLAH KOLOM DAN PERBANDINGANNYA DENGAN FFT.

Selain dengan menggunakan FFT, penggabungan suara dapat dilakukan dengan menggunakan pembatasan jumlah kolom. Berikut adalah contoh programnya:

```
minimum= min([panjang_piano panjang_bass panjang_chord1
panjang_chord2 panjang_chord3 panjang_string
panjang_sinwave]);

a= piano(1:minimum);
b= bass(1:minimum);
c= chord1(1:minimum)+chord2(1:minimum)+chord3(1:minimum);
d= string(1:minimum);
e= sinwave(1:minimum);


song=a+b+c+d+e;
```

Penggabungan suara dengan cara pembatasan jumlah kolom ini memiliki kekurangan jika dibandingkan dengan penggabungan cara FFT. Cara pembatasan kolom mengambil jumlah kolom terpendek, sedangkan cara FFT mengambil jumlah kolom terpanjang. Dengan menggunakan jumlah kolom terpendek, maka pasti akan ada bagian instrumen lain yang terpotong, sehingga lagu tidak menjadi lengkap.

# BAB 5
# KESIMPULAN

Dari hasil analisis, dapat disimpulkan sebagai berikut:

1. Melalui perangkat lunak komposer musik, dapat dibuat musik kita sendiri. Perangkat lunak ini dapat berjalan dengan baik.
2. Untuk mengatur panjang nada dan tempo, dapat dilakukan dengan memanipulasi jumlah kolom dari matriks.
3. Suara dari database (WAV) dapat digabungkan dengan suara yang dibangkitkan dari gelombang sinus dengan mengatur matriks dan tempo (BPM).
4. Pengaturan *volume* dapat dilakukan dengan mengalikan amplitudo dengan suatu bilangan skalar.
5. Grafik suara adalah perbandingan amplitudo terhadap waktu. Satuan amplitudo adalah skala normal atau desibel. Satuan waktu adalah detik.
6. Semakin tinggi nilai BPM, semakin cepat tempo lagu.
7. Penggabungan suara melalui cara FFT lebih baik dari cara membatasi jumlah kolom.

# DAFTAR ACUAN


[1] Bello, J.P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., Sandler, M.B. (2005) "A Tutorial on Onset Detection in Music Signals", IEEE Transactions on Speech and Audio Processing 13(5), pp 1035-1047

[2] Bello, J.P, Duxbury, C., Davies, M., Sandler, M. (2004). "On the use of phase and energy for musical onset detection in the complex domain". *IEEE Signal Processing Letters*

[3] http://www.wikipedia.com 3 juni 2008

[4] http://members.aol.com/chordmaps/index.htm

[5] "im 2023 multimedia", Universitas Kristen Duta Wacana

[6] http://en.wikipedia.org/wiki/Music_information_retrieval

[7] M. Soeharto, Belajar Notasi Balok, Gramedia Jakarta, 1978

[8] http://www.chordwizard.com/theory.html

[9] Iwan Irawan, Pelajaran Gitar Klasik/Spanish, jilid 2, 1982

[10] http://www.zentao.com\guitar\index.htm

# DAFTAR PUSTAKA


Benson, D. "Music: A Mathematical Offering", Department of Mathematics, Meston Building, University, 2007

Blonstein, S. dan Katorgi, M., "eXpressDSP for DUMMIES", Wiley Publishing, Inc., 2004

Gunawan, D., "Lecture Note Digital Signal Processing", Fakultas Teknik Universitas Indonesia, 2003

Ifeachor, E.C. dan Jervis, B.K., "Digital Signal Processing", Prentice Hall Second Edition, 2001

Iwan Irawan, Pelajaran Gitar Klasik/Spanish, jilid 2, 1982

M. Soeharto, Belajar Notasi Balok, Gramedia Jakarta, 1978

# LAMPIRAN 1

## FREKUENSI DAN PANJANG GELOMBANG NADA

Kecepatan Suara = 345 m/s = 1130 ft/s = 770 miles/hr
("*Middle* C" is $C_4$ )
Konversi cm ke inches, bagi dengan 2.54

| Note | Frequency (Hz) | Wavelength (cm) |
|---|---|---|
| $C_2$ | 65.41 | 527. |
| $C^{\#}_2/D^{b}_2$ | 69.30 | 498. |
| $D_2$ | 73.42 | 470. |
| $D^{\#}_2/E^{b}_2$ | 77.78 | 444. |
| $E_2$ | 82.41 | 419. |
| $F_2$ | 87.31 | 395. |
| $F^{\#}_2/G^{b}_2$ | 92.50 | 373. |
| $G_2$ | 98.00 | 352. |
| $G^{\#}_2/A^{b}_2$ | 103.83 | 332. |
| $A_2$ | 110.00 | 314. |
| $A^{\#}_2/B^{b}_2$ | 116.54 | 296. |
| $B_2$ | 123.47 | 279. |
| $C_3$ | 130.81 | 264. |
| $C^{\#}_3/D^{b}_3$ | 138.59 | 249. |
| $D_3$ | 146.83 | 235. |
| $D^{\#}_3/E^{b}_3$ | 155.56 | 222. |
| $E_3$ | 164.81 | 209. |
| $F_3$ | 174.61 | 198. |
| $F^{\#}_3/G^{b}_3$ | 185.00 | 186. |
| $G_3$ | 196.00 | 176. |
| $G^{\#}_3/A^{b}_3$ | 207.65 | 166. |
| $A_3$ | 220.00 | 157. |
| $A^{\#}_3/B^{b}_3$ | 233.08 | 148. |
| $B_3$ | 246.94 | 140. |
| $C_4$ | 261.63 | 132. |

| Note | Frequency (Hz) | Wavelength (cm) |
|---|---|---|
| $C^{\#}_4/D^{b}_4$ | 277.18 | 124. |
| $D_4$ | 293.66 | 117. |
| $D^{\#}_4/E^{b}_4$ | 311.13 | 111. |
| $E_4$ | 329.63 | 105. |
| $F_4$ | 349.23 | 98.8 |
| $F^{\#}_4/G^{b}_4$ | 369.99 | 93.2 |
| $G_4$ | 392.00 | 88.0 |
| $G^{\#}_4/A^{b}_4$ | 415.30 | 83.1 |
| $A_4$ | 440.00 | 78.4 |
| $A^{\#}_4/B^{b}_4$ | 466.16 | 74.0 |
| $B_4$ | 493.88 | 69.9 |
| $C_5$ | 523.25 | 65.9 |
| $C^{\#}_5/D^{b}_5$ | 554.37 | 62.2 |
| $D_5$ | 587.33 | 58.7 |
| $D^{\#}_5/E^{b}_5$ | 622.25 | 55.4 |
| $E_5$ | 659.26 | 52.3 |
| $F_5$ | 698.46 | 49.4 |
| $F^{\#}_5/G^{b}_5$ | 739.99 | 46.6 |
| $G_5$ | 783.99 | 44.0 |
| $G^{\#}_5/A^{b}_5$ | 830.61 | 41.5 |
| $A_5$ | 880.00 | 39.2 |
| $A^{\#}_5/B^{b}_5$ | 932.33 | 37.0 |
| $B_5$ | 987.77 | 34.9 |