PEMBUATAN DATABASE TRANSKRIP AKORD INSTRUMEN TUNGGAL MENGGUNAKAN METODE *ENHANCED PITCH CLASS PROFILE* (EPCP)

Milla Fitriani Kushidayati[1], Miftahul Huda[2], Setiawardhana[2]
[1]Mahasiswa [2]Dosen
Jurusan Telekomunikasi – Politeknik Elektronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember (ITS) Surabaya
Kampus PENS – ITS, Keputih, Sukolilo, Surabaya
Telp : +62+031+5947280, Fax +62+031+5946011
e-mail : millaszone@gmail.com
dayat@student.eepis-its.edu

Abstrak - Bermain musik dan mengenali akord yang dimainkan adalah dua hal yang tidak terpisahkan. Namun, mengenali akord bukanlah perkara yang mudah. Kebanyakan pemula sering mengalami kesulitan dalam menentukan jenis akord yang terdiri dari beberapa nada. Oleh karena itu, orang lebih memilih bergantung kepada guru privat untuk belajar musik.

Didasarkan pada sifat manusia yang menyukai cara praktis dan mudah dalam menyelesaikan masalah. Maka dalam proyek akhir ini, akan dirancang sebuah software pendeteksi akord secara otomatis sehingga mudah untuk dipahami oleh orang yang belum mengenal musik. Dengan menggunakan metode EPCP (*Enhanced Pitch Class Profile*) yang terdiri dari 12 *class* nada berdasarkan range frekuensi nada yang dihasilkan dari deteksi puncak pada proses FFT(*Fast Fourier Transform*) mampu mengenali jenis akord yang dimainkan tanpa memperdulikan tingkatan oktaf.

Pembuatan database *software* ini, mampu menyimpan data yang telah diproses melalui proses *sampling* kemudian proses *thresholding* yang berfungsi untuk mengambil *peak* yang dominan dan akan dilanjutkan pada proses selanjutnya yaitu *frame blocking*, *windowing*, FFT dan EPCP. Masukan data yang akan diproses berupa file audio berekstensi *.wav. Pendeteksian jenis akord ini, hanya untuk jenis akord mayor dan minor saja.

Kata Kunci : **Metode EPCP*(Enhanced Pitch Class Profile), Sampling,* FFT *(Fast Fourier Transform),* Database**

1. PENDAHULUAN

1.1 Latar Belakang

Seiring berkembangnya jaman, musik telah menjadi salah satu alternatif hiburan bagi kebanyakan masyarakat baik tua maupun muda. Namun, kebanyakan orang menjadi sebatas penikmat saja tanpa mengetahui, memahami dan menguasai komposisi nada yang tediri dari beberapa barisan akord.

Salah satu masalah yang paling sering ditemui dalam mempelajari musik terutama piano adalah susahnya mengenali akord yang dimainkan. Selain itu, kesulitan yang juga ditemui adalah ketidaktahuan peletakan akord pada oktaf keberapa saat dimainkan padahal penentuan oktaf dalam memainkan piano sangatlah penting. Posisi oktaf menentukan tinggi rendahnya nada, semakin tinggi oktaf maka nada yang dihasilkan juga akan semakin tinggi. Dari permasalahan tersebut, sebagian besar orang untuk mempelajari akord membutuhkan guru musik baik guru privat maupun pendidikan formal sekolah musik.

Oleh karena itu, dalam proyek akhir ini akan dirancang sebuah *software* yang dapat menampilkan nama akord secara langsung dengan metode EPCP (*Enhanced Pitch Class Profile*) yang memungkinkan akord dikenali dengan error yang kecil. Untuk dapat menampilkan jenis akord tersebut dibutuhkan pengumpulan data dari input audio *wav yang akan disimpan di *database*.

Penggunaan *software* ini sangat diperlukan untuk mempelajari jenis akord khususnya akord mayor dan minor. Pengguna dapat mempelajari jenis akord secara individu tanpa harus membutuhkan guru privat lagi.

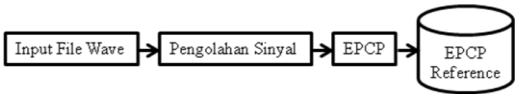
1.2 Permasalahan

Dalam pembuatan pada proyek akhir ini, terdapat beberapa permasalahan, antara lain sebagai berikut :

a. Pengumpulan data berupa file .wav sebagai masukan yang akan diproses mengunakan pengolahan digital berupa sinyal.
b. Bagaimana menentukan algoritma yang tepat pada sistem yang akan dibuat.
c. Bagaimana merancang database dengan menggunakan komponen pengolahan sinyal yang mendukung aplikasi developer yaitu Delphi.
d. Bagaimana mengolah metode penyimpanan database berupa array.

2. PERENCANAAN DAN TEORI PENUNJANG

Dalam proyek akhir ini, dasar teori yang digunakan untuk mendukug perencanaan sistem yaitu, materi tentang akord, EPCP, dan pengolahan sinyal.



Gambar 1 Skema Perencanaan Sistem

2.1 Akord

Akord merupakan gabungan dari beberapa not yang dimainkan secara bersamaan membentuk harmoni sebagai pengiring melodi yang dimainkan di sisi lain. Akord dapat dimainkan dengan iringan yang bervariatif sesuai dengan kesesuaian dengan jenis dan tempo dari lagu yang dimainkan. Jenis akord bermacam-macam, antara lain akord mayor, akord minor, akord diminished, akord augmented, akord minor seventh, akord mayor seventh, akord suspended dan lain-lain. Masing-masing jenis akord tersebut memiliki karakteristik dan aturan pembentukkannya sendiri.

2.2.1 Akord Mayor

Akord Mayor biasanya dituliskan hanya berupa huruf kapital seperti C, D, E, F, G, A, B. Untuk mencari akord mayor maka anda dapat menggunakan nada ke 1 - 3 - 5 dari (C) *Mayor Scales*, contoh akord mayor :

- Cb (Cb-Eb-Gb) = B
- C (C-E-G)
- C# (C#-E#-G#) = Db (Db-F-Ab)
- D (D-F#-A)
- D# (D#-G-A#) = Eb (Eb-G-Bb)
- E (E-G#-B) = Fb (Fb-Ab-Cb)
- E# (E#-A-B#) = F (F-A-C)
- F (F-A-C)
- F# (F#-A#-C#) = Gb (Gb-Bb-Db)
- G (G-B-D)
- G# (G#-B#-D#) = As (Ab-C-Eb)
- A (A-C#-E)
- A# (A#-D-E#) = Bb (Bb-D-F)
- B (B-D#-F#) = Cb
- B# (B#-E-G) = C

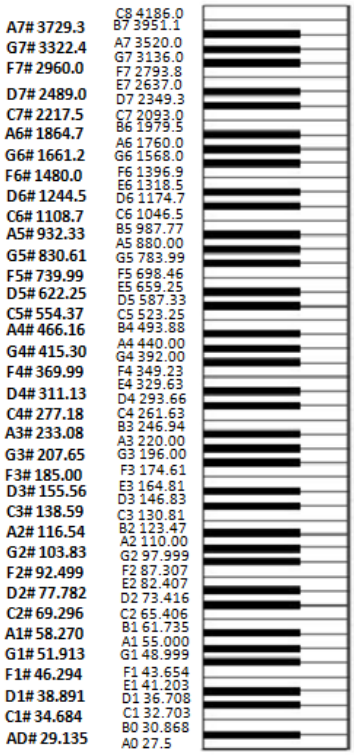
Akord yang memiliki nama berbeda namun bila dimainkan bersuara sama disebut Akord Enharmonis

2.2.2 Akord Minor

Akord Minor biasanya dituliskan dengan penambahan karakter ‘m’ setelah huruf Kapital seperti Cm, Dm, Em, Fm, Gm, Am, Bm. Biasanya penuliskan akord minor dilakukan dengan huruf kecil seperti c, d, e, f, g, a, b. Apabila pengguna sudah mengetahui suatu akord mayor misalnya; C mayor maka anda bisa mengetahui pula akord minornya (C minor) yaitu dengan cara menurunkan nada yang ada ditengah sebanyak setengah interval. Sehingga didapat akord C minor adalah C-Es(E diturunkan setengah menjadi Es)-G. Dapat juga dapat dicari dengan menggunakan nada ke 1 – 3b (nada ke 3 diturunkan setengah interfal) – 5 dari (C) *Mayor Scales*, sebagai contoh :
Akord Cm = Nada C - D#/Eb – G.

Apabila ditemukan nada maupun akord seperti D#/Eb, hal tersebut berarti bunyi dari D# terdengar sama dengan bunyi Eb. Akord inilah yang disebut sebagai akord atau nada Enharmonis. Karena akord atau nada seperti ini memiliki nama berbeda namun bila dimainkan menghasilkan frekuensi dan suara yang sama.

2.2 Penentuan Frekuensi Pada Piano



Gambar 2 Frekuensi Tiap Bin Piano

Setiap nada memiliki frekuensi masing-masing. Secara internasional, nada A4(A pada oktaf ke-4) memiliki frekuensi 440 Hz. Dan untuk mengetahui berapa frekuensi nada-nada lainnya dapat menggunakan persamaan :

$$f_n = f_0 * a^n \quad \text{..... [1]}$$

Dimana:

f_n adalah frekuensi nada dengan jarak-n yang dicari

f_o adalah frekuensi yang diketahui (A4 =440 Hz)

a^n adalah $(2)^{1/12}$

2.3 Metode EPCP (*Enhanced Pitch Class Profile*)

EPCP merupakan singkatan dari *Enhanced Pitch Class Profile*. Masing-masing nada dalam musik memiliki pitch. Metode EPCP merupakan perkembangan dari metode PCP (*Pitch Class Profile*). Pada metode ini, pitch dari nada-nada yang ada dikelompokkan berdasarkan 12 *class* dari *pitch*, yaitu C, C#, D, D#, E, F, F#, G, G#, A, A#, B. Yang membedakan metode EPCP dengan yang lain adalah dalam metode ini dilakukan proses rata-rata pada hasil PCP, sehingga EPCP menghasilkan *error* yang lebih kecil. Pada aplikasinya, dengan menggunakan metode EPCP untuk *Chord Recognition* maka pada oktaf mana pun akord dimainkan, tidak akan mempengaruhi keluaran dari perangkat lunak.

2.4 Pengolahan Sinyal Digital

2.4.1 Thresholding

Pada proses ini sinyal input diproses dengan mengambil peak-peak dominan. Sebelumnya,sinyal akan mengalami *powering* terlebih dahulu dengan melakukan pengkuadratan pada sinyal, dengan persamaan :

$$P = \sum_{i=0}^{i=N} \sqrt{X_i^2} \qquad \text{..... [2]}$$

Dimana:

P = Power sinyal (dB)
Xi = Frame ke-i (dB)
N = Jumlah sampel per frame
i = Sampel ke-i

Powering bertujuan untuk mengetahui letak awal dan akhir dari suatu sinyal yang berada di suatu frame. Dari nilai power didapatkan suatu nilai rata-rata, dengan menambahkannya dengan standar deviasi, maka didapatkan nilai awal dan akhir dari suatu *frame.*

Standar deviasi merupakan penyimpangan data standar dari data yang ada. Jika nilai data mendekati nilai rata-rata, maka standar deviasinya lebih kecil (mendekati nol) dan sebaliknya. Berikut persamaan dari standart deviasi dan rata-rata :

$$\text{Standar Deviasi} = \sum_{i=0}^{n} \frac{x_i^2}{n} \qquad \text{..... [3]}$$

$$Rata - rata = \sum_{i=0}^{n} \frac{x_i}{n} \qquad \text{..... [4]}$$

Sinyal dengan nilai power diatas standart deviasi diambil sebagai awal dan akhir sinyal serta dianggap sebagai suatu sound.

sound > mean + standart deviasi

2.4.2 Frame Block

Frame Blocking adalah pembagian sinyal audio menjadibeberapa frame. Satu frame terdiri dari beberapa sampel tergantung tiap berapa detik suara akan disampel dan berapa besar frekuensi samplingnya. Pada proses ini dilakukan pemotongan sinyal dalam slot-slot tertentu agar memenuhi 2 syarat yaitu linear dan time invariant.

2.4.3 Windowing

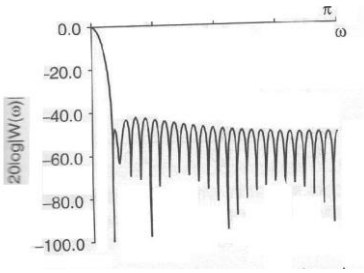
Windowing digunakan untuk menghilangkan discontinuitas yang diakibatkan oleh proses Frame Blocking atau Framing. Pada proses ini jenis window yang dipakai adalah jenis Hamming Window. Disini digunakan Hamming window karena hamming window mempunyai side lobe yang paling kecil dan Main lobe yang paling besar sehingga hasil windowing akan lebih halus dalam mengghilangkan efek diskontinuitas. Beberapa jenis window yaitu *Hamming, Blackman, Hanning, Bartlet*, dan *Rectanguler*.

Persamaan *Hamming window* :

$$w(n) = 0.54 - 0.46 \cdot \cos\left(\frac{2 \cdot \pi \cdot n}{N-1}\right) \text{..... [5]}$$

Dimana :

w(n) adalah *windowing*
N merupakan jumlah data dari sinyal
n merupakan waktu diskrit ke



Gambar 3 Respon Frekuensi Hamming *Window*

2.4.4 FFT (*Fast Fourier Transform*)

Transformasi fourier adalah suatu metode yang sangat efisien untuk menyelesaikan transformasi fourier diskrit yang banyak dipakai untuk keperluan analisa sinyal seperti pemfilteran, analisa korelasi, dan analisa *spectrum*. *Diskrit Fourier Transformasi* (DFT) adalah deretan yang terdefinisi pada kawasan frekuensi – diskrit yang merepresentasikan *Transformasi Fourier* terhadap suatu deretan terhingga (*finite duration sequence*). DFT berperan penting untuk implementasi algoritma suatu varitas pengolahan sinyal, karena efisien untuk komputasi berbagai aplikasi.

Fast Fourier Transformation atau transformasi Fourier cepat, merupakan proses lanjutan dari DFT *(Diskrit Fourier Transformation*).

Transformasi Fourier ini dilakukan untuk mentransformaikan sinyal dari domain waktu ke domain frekuensi. FFT adalah bentuk khusus dari persamaan integral fourier.

Di dalam proses *Fast Fourier Transform* akan menghasilkan dua buah nilai yaitu nilai real dan nilai imajiner.

3. PEMBUATAN DAN PENGUJIAN

Dalam proses pembuatan, proses-proses yang dilakukan adalah perekaman sebagai proses pengambilan data, pengolahan sinyal, EPCP, dan proses input database EPCP Reference

3.1 Perekaman

Dalam melakukan perekaman dibutuhkan software yang mampu untuk merekam audio file berekstensi *.wav dan menetapkan pengaturan *sampling rate* serta pengaturan channel yang akan digunakan. Oleh karena itu, untuk mempermudah proses perekaman digunakan *software* Steinberg Nuendo. Pengaturan perekaman Pada Proyek Akhir ini menggunakan sampling rate sebesar 44100 Hz dan mono *channel*. Berikut penjelasan gambar mengenai ruang lingkup menggunakan *software* Steinberg Nuendo.

3.2 Pengolahan sinyal

3.2.1 Proses Sampling

Pada proses *sampling* ini dilakukan pembacaan data dari masukan file audio wav. Proses pembacaan akan dimasukkan ke dalam array sehingga file audio wav dapat diproses dan disimpan dalam penyimpanan ini. Informasi header file wav dapat diperoleh dengan melakukan pembuatan tipe data dasar seperti *chunk size* yang bertipe *cardinal, sample rate* yang bertipe *longint*, dst.

3.2.2 Thresholding

Pada proses ini dilakukan pengambilan *peak* dominan dari sinyal dengan terlebih dahulu melakukan *powering* dan penentuan standar deviasi yang berguna sebagai pendeteksi awal dan akhir sinyal dalam *frame*.

3.2.3 Frame Blocking

Frame Bloking merupakan pembagian sinyal audio menjadi beberapa frame yang akan memudahkan dalam perhitungan dan analisa sinyal, satu frame terdiri dari beberapa sampel tergantung tiap berapa detik suara yang akan disample dan besarnya frekuensi samplingnya. Pada proses ini dilakukan pemotongan sinyal dalam slot-slot tertentu agar dianggap invariant. Berikut ini rumus yang digunakan untuk pengambilan sample tiap frame :

$$SFr = Fs * (t / 1000) \quad \text{..... [6]}$$

Dimana :

SFr : Sample per Frame

Fs : Frekuensi sample / sample rate dari format audio wav

t : waktu pengambilan (ms)

3.2.4 Windowing

Pada dasarnya *windowing* mempunyai beberapa jenis *windowing*, diantaranya : *Hamming*, *Hanning*, *Bartlet*, *Rectangular* dan *Blackman*. Dalam proyek akhir ini menggunakan jenis *windowing hamming* karena *window hamming* memiliki main lobe paling besar dan side lobe paling kecil. Proses *windowing* berada pad aproses FFT, karena fungsi *windowing* berada dalam komponen DSPlab sehingga pemanggilan fungsi *window* sudah tersedia dalam komponen tersebut.

3.2.5 FFT

Nilai awal yang sudah didapatkan melaui proses *thresholding* menjadi parameter pada fungsi FFT. Dalam pemprosesannya, mulai dari titik awal sinyal akan diambil sample sebanyak *buffer size* dari FFT yang digunakan untuk dimasukkan ke dalam veriabel array RealIn yang akan dihitung megnitudenya.

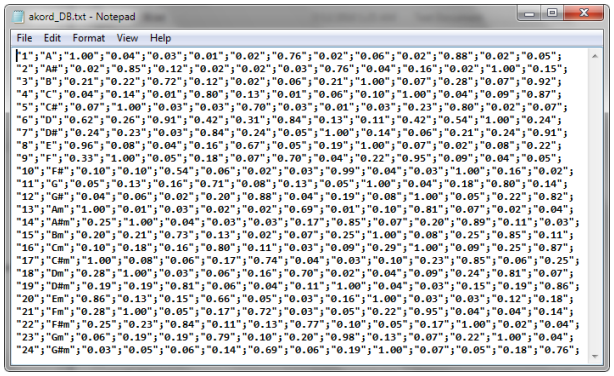
3.2.6 EPCP

Saat melakukan proses EPCP, akan medeteksi puncak yang ada pada proses FFT. Nilai puncak-puncak tersebut akan dimaukkan ke dalam 12 *class* PCP berdasarkan pembagian range frekuensi nada. Nilai puncak yang didapatkan dari proses FFT merupakan frekuensi dari sinyal yang terproses.

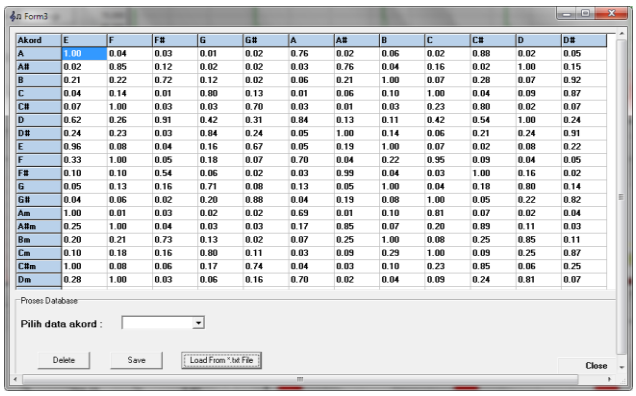
3.3 PENGUJIAN

3.3.1 Database

Dalam pengujian, untuk penggunaan *database* memanfaatkan notepad yaitu file yang berekstensi *.txt. Dalam penyimpanan data ke dalam txt ini memiliki syarat yaitu setiap data yang dituliskan dipisahkan dengan tanda petik (“) dan titik koma(;). Format tersebut bertujuan untuk memudahkan penulisan data dan pembacaan data yang disimpan berdasarkan index array. Berikut gambar penjelasan untuk format *database*:



Gambar 4 Format Database



Gambar 5 Tampilan Format Database

3.3.2 Waktu Komputasi

Dalam menjalankan program ini membutuhkan waktu untuk melakukan proses komputasi yang terjadi pada saat pengambilan data file audio wav. Proses komputasi memerlukan beberapa waktu yang berbeda pada setiap masukan file audio wav. Pembacaan waktu komputasi yang berbeda-beda bukan karena perbedaan 24 akord mayor dan minor melainkan berdasakan durasi perekaman file audio wav, semakin panjang durasi suatu file wav maka semakin lama waktu perhitungan yang dibutuhkan. Dibawah ini adalah tabel 24 akord dengan masing-masing rata-rata waktu komputasinya :

Tabel 1. Tabel Waktu Komputasi

Akord	Waktu Komputasi (detik)										Rata-rata
	1	2	3	4	5	6	7	8	9	10	
C	18	19	20	18	20	19	22	21	18	18	19.3
Cm	21	18	19	18	19	18	24	19	20	17	19.3
C#	24	19	20	18	17	16	16	16	18	16	18
C#m	20	16	16	16	16	18	18	19	18	18	17.5
D	23	21	19	19	20	20	18	21	20	19	20
Dm	23	20	18	27	24	24	24	23	24	24	23.1
D#	16	23	22	23	23	20	22	20	21	21	21.1
D#m	21	20	19	21	22	22	17	17	17	23	19.9
E	23	17	18	26	26	25	16	19	17	17	20.4
Em	16	18	17	16	16	17	20	17	19	18	17.4
F	19	16	18	17	22	19	15	15	15	18	17.4
Fm	20	17	16	14	15	13	13	13	13	14	14.8
F#	23	15	15	15	14	14	15	14	13	14	15.2
F#m	21	13	13	23	25	24	25	25	25	25	21.9
G	20	26	22	23	22	15	14	14	14	14	18.4
Gm	23	15	14	14	15	14	16	24	21	21	17.7
G#	18	15	18	19	27	25	20	23	18	23	20.6
G#m	25	26	20	17	22	26	17	15	20	14	20.2
A	36	18	18	17	16	16	18	15	17	16	18.7
Am	25	17	16	17	17	15	17	16	22	17	17.9
A#	34	18	14	14	15	16	15	19	18	16	17.9
A#m	22	15	16	16	17	19	17	16	17	20	17.5
B	26	22	16	15	15	14	16	15	18	17	17.4
Bm	26	17	15	16	17	15	17	19	16	17	17.5

3.3.3 Standart Deviasi

Nilai standart deviasi didapatkan dari setiap data wave yang dikuadratkan dibagi dengan jumlah data wave. Standart deviasi ini merupakan batas toleransi kesalahan. Sedangkan nilai rata-rata didapatkan dari jumlah semua data wave dibagi dengan banyaknya jumlah data wave. Berikut hasil dari masing-masing akord dengan nilai rata-rata dan standart deviasinya.

Tabel 2. Tabel Standart Deviasi

Akord	Rata-Rata	Standart Deviasi
C	31.66	5554.95
Cm	25.56	6052.95
C#	36.10	5563.12
C#m	13.34	5711.76
D	27.73	4849.24
Dm	31.03	5396.77
D#	24.01	5954.93
D#m	28.07	5839.39
E	39.17	6537.41
Em	24.16	6581.60
F	30.30	5678.68
Fm	58.63	6029.23
F#	26.53	4947.56
F#m	36.10	5159.90
G	32.46	6682.76
Gm	33.61	5461.89
G#	29.03	5558.04
G#m	12.82	6530.89
A	26.77	4599.72
Am	31.93	4882.60
A#	27.81	4731.10
A#m	29.88	5693.33
B	33.34	6037.55
Bm	28.06	5932.15

4. PENUTUP

4.1 KESIMPULAN

Berdasarkan pada hasil pengujian sistem dan analisa hasil, didapatkan kesimpulan , yaitu:

1. Hasil waktu komputasi yang dilakukan berdasarkan tiap-tiap akord yang terdiri dari 5 frame memiliki waktu komputasi yang berbeda-beda disebabkan oleh panjang data file audio wav pada saat perekaman. Semakin panjang file audio wav maka proses perhitungan pun akan semakin lama.
2. Tiap 24 akord mayor dan minor memiliki nilai standart deviasi dan nilai rata-rata yang berbeda.
3. Pada sistem *database* masih memakai 2 pembatas yaitu titik koma (;) dan tanda petik (“) dengan menggunakan metode split untuk pengambilan datanya.

5. REFERENSI

[1] Febrianzah Junaidy Permana, “Pembuatan Database Software Digital Musik Mentor”,Surabaya, Agustus 2009

[2] Fandy Akbar, “Pembuatan Software Digital Musik Mentor”, Surabaya, Agustus 2009.

[3] Kyogu Lee, *”Automatic Chord Recogniton from Audio using a Supervised HMM trained with Audio-from-Symbolic Data”*,*Sunnyvale*,2001

[4] http://id.wikipedia.org/wiki/Teori_musik

[5] http://yoyokpm._files.wordpress.com_/2008/04/teori_musik1.pdf

[6] Steven W. Smith, “*The Scientist and Engineer’s Guide to Digital Signal Processingsecond Edition”, California Technical Publishing* San Diego, California.