APLIKASI PLAYER UNTUK MENJALANKAN FILE WAVE YANG TERKOMPRESI DENGAN METODE HUFFMAN

Karmela Saturnina Mega Wea, Willy Sudiarto R., Antonius Rachmat C.
Program Studi Teknik Informatika
Fakultas Teknik Universitas Kristen Duta Wacana
Jl. Dr. Wahidin Sudirohusodo 5 – 25, Yogyakarta 55224, Indonesia
Email: leni_wea@yahoo.com, willysr@gmail.com, anton@ukdw.ac.id

Abstrak:

File wave merupakan salah satu format file audio yang banyak dipakai dalam sistem operasi Windows untuk keperluan game dan multimedia. Semakin lama durasi sebuah file wave, semakin besar kapasitas media penyimpanan yang dibutuhkan untuk menyimpan data audio file wave tersebut. Kompresi data merupakan salah satu cara yang dapat dimanfaatkan untuk mengatasi masalah ukuran file wave yang besar. Metode kompresi yang digunakan adalah metode Huffman.

Untuk dapat menjalankan atau memainkan file wave terkompresi tersebut dibuat sebuah aplikasi player. Penggunaan metode Huffman dapat memperkecil file wave dengan penghematan rata-rata sebesar 18,04 %. Akan tetapi pada beberapa kasus kompresi file wave yang memiliki jumlah karakter unik mendekati 256 karakter dan banyak pola data yang mempunyai frekuensi kemunculan sama, dengan menggunakan metode ini justru menghasilkan ukuran file yang lebih besar dari ukuran file aslinya. File wave hasil kompresi dengan metode Huffman dapat dimainkan atau dijalankan dengan cara didekompresi terlebih dahulu.

Kata kunci: aplikasi player, WAVE, kompresi data, metode Huffman

1. Pendahuluan

Format wave **(*.WAV**) merupakan salah satu format *file* suara yang banyak dipakai dalam sistem operasi **Windows** untuk keperluan *game* dan *multimedia*. Format wave sebenarnya merupakan format kasar (*raw format*) dimana signal suara langsung direkam dan dikuantisasi menjadi data digital. Format dasar dari *file* ini secara default tidak mendukung kompresi dan dikenal dengan nama PCM (*Pulse Code Modulation*).

Semakin lama durasi sebuah *file* wave, semakin besar kapasitas media penyimpanan yang dibutuhkan untuk menyimpan data audio *file* tersebut. Media penyimpanan atau biasa dikenal dengan HDD (*Hard Disk Drive*) yang semakin besar tidak akan menjawab kebutuhan

teknologi informasi jika data berkas (*file*) yang digunakan juga semakin besar. Masalah tersebut dapat diatasi bila *file* wave tersebut dikompresi untuk mengurangi ukurannya. Pada dasarnya, algoritma kompresi terbagi atas *lossless compression* dan *lossy compression***.** Algoritma kompresi untuk jenis kompresi *lossless* yang banyak digunakan diantaranya adalah Huffman, RLE, LZ77, LZ78 dan LZW. Untuk menjalankan *file* wave yang terkompresi tersebut, dibutuhkan sebuah aplikasi player yang mampu melakukan dekompresi *file* secara on-the-fly.

Pada penelitian ini akan dirancang sebuah aplikasi untuk kompresi file wave dengan metode Huffman dan sebuah aplikasi *player* untuk menjalankan *file wave* baik yang terkompresi maupun tidak terkompresi. Agar pembahasan hanya terfokus pada kompresi file wave dengan metode Huffman dan player untuk menjalankan *file wave*, maka aplikasi dibuat dengan batasan masalah sebagai berikut:

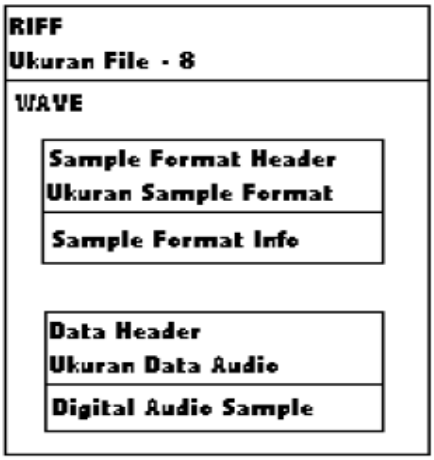
1. *File* Input hanya berupa *file* format Wave dengan audio format berjenis PCM (*Pulse Code Modulation*) dan hanya mendukung jumlah kanal maksimum 2 buah kanal (*mono* dan *stereo*).
2. Program tidak dapat melakukan pengubahan jumlah kanal (channel), bit per sample, dan sampling rate *file* Wave.
3. Program dapat menjalankan *file* Wave terkompresi tersebut dengan pilihan Play, Stop, dan Pause.

2. Landasan Teori

2.1 Format File WAV

File wave adalah format berkas suara yang diciptakan oleh perusahaan piranti lunak raksasa Microsoft dan telah menjadi standar berkas data suara pada PC maupun Macintosh. Format berkas WAV digunakan untuk menyimpan sinyal data suara digital ke dalam suatu berkas. Format ini sangat terkenal pada platform komputer IBM PC dan kompatibel karena mendukung bermacam-macam resolusi bit, sampling rate dan channel suara. Format ini adalah varian dari format RIFF yang menggunakan metoda Electronic Arts Interchange File Format versi Microsoft untuk menyimpan data suara ke dalam ‘potongan-potongan’ (chunks) (Goethals, 2004). Format RIFF WAVE tersebutlah yang selanjutnya disebut WAVE dan memiliki ekstensi WAV. Format ini dirancang sehingga data-data yang tersimpan terpecah-pecah ke dalam bagian-bagian deskriptif (*self-described*) yang independen. Setiap bagian memiliki prefiks yang menggambarkan data yang merupakan bagiannya. Prefiks ini berupa empat karakter yang disebut ChunkID untuk mendefenisikan tipe data di dalam bagiannya (Di Silvestro, Baribault).

Sesuai dengan struktur file RIFF, file WAV diawali dengan 4 byte yang berisi ‘RIFF’ lalu diikuti oleh 4 byte yang menyatakan ukuran dari file tersebut dan 4 byte lagi yang berisi ‘WAVE’ yang menyatakan bahwa file tersebut adalah file WAV. Berikutnya adalah informasi dari format sample yang menjadi sub-bagian dari bagian RIFF lalu diikuti sub-bagian data audionya.



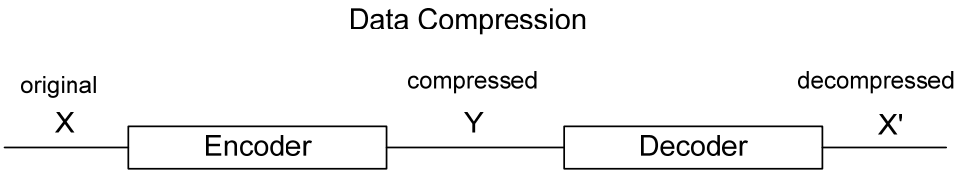
Gambar 1. Struktur File Wave

2.2 Kompresi Data

Kompresi data adalah proses pemadatan data yang bertujuan untuk memperkecil ukuran data sehingga selain dapat menghemat media penyimpanan, dapat memudahkan transfer data dalam jaringan (Howe, 1993). Kompresi data juga didefinisikan sebagai suatu proses mengubah suatu input data menjadi data lain dengan format berbeda dan dengan ukuran yang lebih kecil (Salomon, 2000, hlm.2).

File merupakan data digital yang berupa representasi atas bit ‘0’ dan ‘1’. Seringkali dalam sebuah *file* terjadi perulangan data atau *redundancy*. Semua metode kompresi melakukan pemadatan terhadap data berulang tersebut.

Berdasarkan hasil kompresi, algoritma kompresi terbagi atas *lossless compression* dan *lossy compression* (Sayood, 2005). Pada *lossy compression* terdapat data yang hilang setelah proses kompresi. Contoh standar yang menggunakan jenis *lossy compression* adalah JPEG (*Joint Picture Experts Group*) sebagai standar *image* gambar atau *still image*, MPEG (*Motion Picture Experts Group*) untuk *audio video* sepeti Video CD, MP3 (*MPEG-1 Layer 3*) untuk *audio*. Data hasil kompresi dengan lossy compression jika dikembalikan maka hasilnya tidak akan sama persis lagi dengan data orisinil. Berbeda dengan *lossy compression,* pada *lossless compression* tidak terdapat data yang hilang setelah proses kompresi dan data dapat dikembalikan seperti data semula. Contoh standar yang menggunakan jenis ini adalah Gzip, Unix Compress, WinZip, GIF (*Graphic Interchange Form*at) untuk *still image*, dan Morse Code.



Gambar 2. Skema Proses Kompresi dan Dekompresi

2.3 Algoritma Huffman

Algoritma Huffman ditemukan oleh Huffman pada tahun 1952 dan dipublikasikan pada paper yang berjudul *"A method for the construction of minimum-redudancy codes*". Algoritma Huffman dapat dipakai sebagai algoritma pencarian (*searching*) dan algoritma pemampat data (*data compression*) (Widyawardhana, 2000).

Algoritma Huffman dapat diterapkan untuk melakukan pencarian karena pohon Huffman dibentuk dengan meletakan item yang paling sering muncul di daun yang paling dekat dengan akar dan item yang paling jarang muncul ditempatkan pada daun yang paling jauh dengan akar. Pencarian dilakukan dengan membaca pohon tersebut satu per satu mulai dari akar, sehingga didalam proses pencarian item yang paling sering muncul dapat ditemukan. Dengan asumsi bahwa pencarian akan sering diperlukan untuk item yang sering muncul maka algoritma ini menjadi efektif.

Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (isi data yang dimasukan) menjadi sekumpulan *codeword*, algoritma Huffman termasuk ke dalam kelas algoritma yang menggunakan metode statis. Metodee statis adalah metode yang selalu menggunakan peta kode yang sama, metodee ini membutuhkan dua fase (*two-pass*): fase pertama untuk menghitung probabilitas kemunculan tiap simbol dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan di transmisikan.

Sedangkan berdasarkan teknik pengodean simbol yang digunakan, algoritma Huffman menggunakan metode *symbolwise*. Metode *symbolwise* adalah metode yang menghitung peluang kemunculan dari setiap simbol dalam satu waktu, di mana simbol yang lebih sering muncul diberi kode lebih pendek dibandingkan simbol yang jarang muncul.

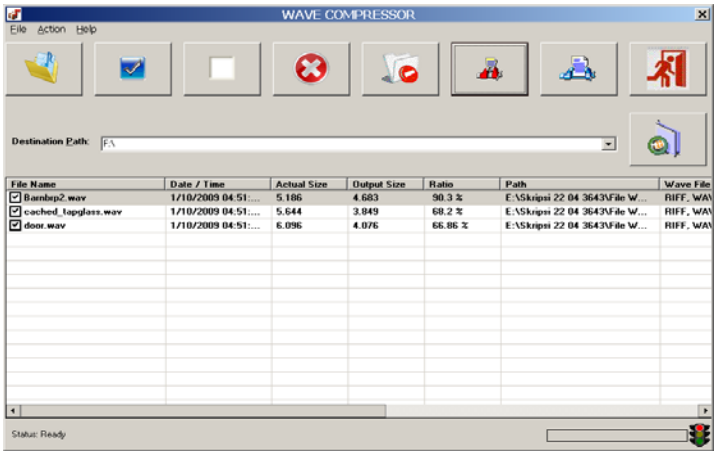
2.4 Rasio Kompresi

Dalam kompresi data, tujuan utama yang perlu diperhatikan adalah rasio kompresi yang semakin baik, dan proses kompresi dan pengembalian yang semakin cepat. Rasio kompresi secara matematis dapat ditulis sebagai berikut:

$$\text{Rasio Kompresi} = \frac{\text{Output Stream}}{\text{Input Stream}} \qquad \text{.....(1)}$$

3. Hasil dan Pembahasan

Percobaan akan dilakukan dengan menggunakan beberapa contoh kasus untuk melakukan proses kompresi dengan sistem. Percobaan pertama akan dilakukan pada *file wave* dengan ukuran kecil.

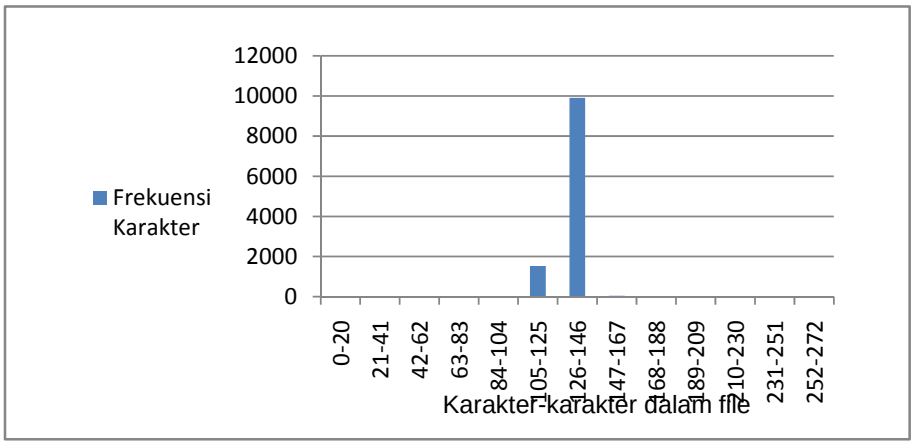


Gambar 3. Contoh Kasus Kompresi File Wave dengan Ukuran Kecil

Tabel 1. Hasil pengujian pada beberapa file wave dengan ukuran kecil

Nama File	Ukuran Asli File (Byte)	Ukuran File Hasil Kompresi (Byte)	Rasio	Space Savings	Waktu Kompresi (ms)
Bird4.wav	11.550	5.046	43,69%	56,31%	31
Cached_tapglass.wav	5.644	3.849	68,2%	31,80%	31
Door.wav	6.096	4.076	66,86%	33,14%	16
Ohman2.wav	3.908	3.369	86,21%	13,79%	16
Cached_fart.wav	6.728	4.050	60,2%	39,8%	16
Melano1.wav	6.204	5.448	87,81%	12,19%	15
Cached_dingsuperhi.wav	8.766	6.114	69,75%	30,25%	32
Effect.wav	8.354	6.693	80,12%	19,88%	16

Dari Tabel 1 dapat dilihat bahwa rasio kompresi terkecil pada beberapa *file* dengan ukuran kecil terdapat pada *file* Bird4.wav yaitu 43,69%.

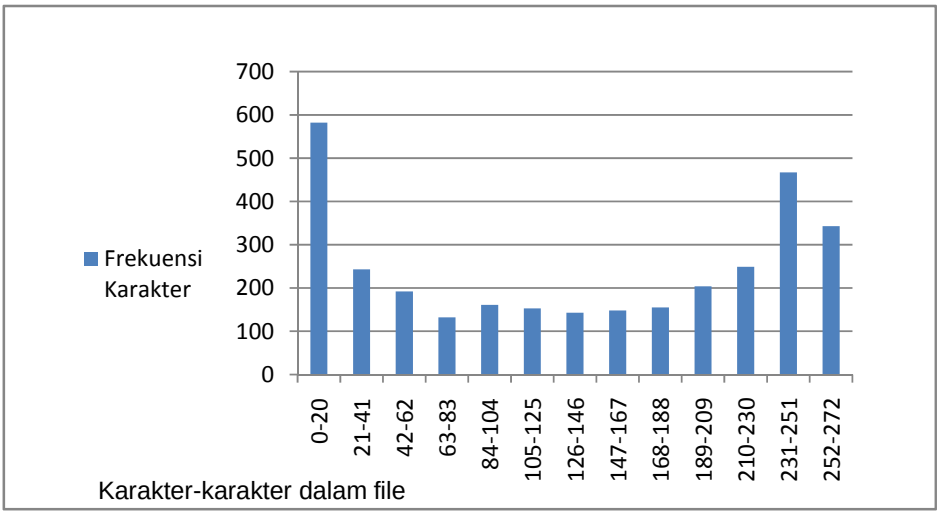


Gambar 4. Karakter-karakter yang ada dalam file Bird4.wav dalam bentuk ASCII

Tabel 2 menunjukkan hasil kompresi pada *file wave* dengan ukuran kecil yang rasio kompresinya lebih dari 100 %.

Tabel 2. Hasil Kompresi dengan rasio di atas 100 %

Nama File	Ukuran Asli File (Byte)	Ukuran File Hasil Kompresi (Byte)	Rasio	Waktu Kompresi (ms)
Cached_buy.wav	3.216	3.837	119,31%	32
Cached_sfx.wav	6.470	6.519	100,76%	31



Gambar 5. Karakter-karakter yang ada dalam file cached_buy.wav dalam bentuk ASCII

Pada *chunk* data *file* cached_buy.wav dapat dilihat bahwa jumlah karakter-karakter yang terdapat dalam *chunk* data *file* tersebut mendekati 256 karakter dan banyak pola data mempunyai frekuensi kemunculan sama.

Pengujian proses kompresi juga dilakukan terhadap *file wave* dengan ukuran di atas 5 MB.

Tabel 3. Hasil Pengujian Pada Beberapa File Dengan Ukuran Besar

Nama File	Ukuran Asli File (Byte)	Ukuran File Hasil Kompresi (Byte)	Rasio	Space Savings	Waktu Kompresi (ms)
Butterfly Fly Away.wav	4.901.212	4.554.855	91,44%	8,56%	7.562
You Are My No.1.wav	7.061.804	5.809.152	82,26%	17,74%	9.422
Only Hope.Wav	12.836.253	11.836.253	95,79%	4,21%	18.679
Eu Sei.Wav	26.053.679	25.178.610	96,64%	3,36%	40.985

Rasio kompresi untuk *file wave* dengan ukuran kecil maupun besar persentasenya hampir sama. Hal ini terjadi karena besar kecilnya rasio kompresi tidak bergantung pada ukuran *file*, tetapi pada isi data *file wave*. Semakin banyak perulangan data yang terdapat pada bagian *chunk* data *file Wave* maka rasio kompresi akan semakin rendah. Semakin rendah rasio kompresinya berarti semakin baik hasil kompresinya. Jika rasio kompresi di atas 100 % maka ukuran *file* hasil kompresi lebih besar daripada ukuran *file* aslinya.

Tabel 4 dan Tabel 5 menunjukan hasil kompresi pada file wave dengan ukuran yang sama tetapi memiliki *sample rate*, bit per sample dan jumlah channel yang berbeda.

Tabel 4.Hasil pengujian pada *file wave* yang mempunyai ukuran sama, jumlah channel sama dan *sample rate* sama tetapi beda bit per sample

Nama File	Ukuran File	Bit per sample	Rasio
Cached_buy.wav	4 Kb	16 bit	119,31%
Ohman2.wav	4 Kb	8 bit	86,21%
Cached_unleash.wav	21 Kb	16 bit	101,14%
Hammerwoodloop.wav	21 Kb	8 bit	36,47%

Tabel 5. Hasil pengujian pada *file wave* yang mempunyai ukuran sama, jumlah channel sama dan bit per sample sama tetapi beda sample rate

Nama File	Ukuran File	Sample Rate	Rasio
Bird1.wav	31 Kb	11 KHz	79,23%
Cached_xgameover.wav	31 Kb	24 KHz	92,63%
Cached_punch.wav	28 Kb	44 Khz	99,67%
Bird2.wav	28 Kb	11KHz	86,9%

Dari Tabel 4 dan Tabel 5 dapat dilihat bahwa rasio kompresi atau besar kecilnya *file* wave hasil kompresi pada *file wave* dengan ukuran yang sama dipengaruhi juga oleh bit per sample, jumlah channel dan sample rate. Saat proses kompresi, pada *file wave* dengan ukuran di atas 5 MB, rata-rata CPU *Usagenya* di atas 50% dengan *memory usage* rata-rata 22.408 K sampai 41.264 K .

Analisis juga dilakukan pada proses dekompresi yang terdapat pada aplikasi *player* untuk menjalankan *file wave* terkompresi dan tidak terkompresi. Pengujian dilakukan terhadap beban kerja sistem. Pada saat menjalankan *file* terkompresi dengan ukuran kecil rata-rata CPU *Usagenya* dibawah 10% dengan *memory usage* rata-rata 22.452 KB sampai 23.888 KB, sedangkan pada saat menjalankan *file wave* dengan ukuran di atas 5 MB, rata-rata CPU *Usagenya* di atas 50% dengan *memory usage* rata-rata 23.496 K sampai 38.888 K. Untuk *file* tidak terkompresi rata-rata CPU *Usagenya* di bawah 10% dengan *memory usage* rata-rata 21.896 K sampai 24.436 K. Dibutuhkan lebih banyak waktu untuk melakukan proses dekompresi pada *file wave* dengan ukuran di atas 5 MB dibandingkan dengan waktu yang dibutuhkan untuk melakukan proses dekompresi pada *file* dengan ukuran kecil, sehingga terdapat jeda waktu yang lebih banyak sebelum *file* dimainkan untuk *file wave* dengan ukuran besar.

Tabel 6. Hasil Pengujian Proses Dekompresi

Nama File	Ukuran File Asli (Byte)	Ukuran File Setelah Di Kompresi (Byte)	Ukuran File setelah Didekompresi (Byte)	Waktu Dekompresi (ms)
Barnbrp2.wav	5.186	4.683	5.186	15
Audidi.wav	22.616	19.832	22.616	78
Klaxon.wav	22.702	20.304	22.702	93
Butterfly Fly Away.wav	9.962.504	9.074.554	9.962.540	13.860

Tabel 6 menunjukan bahwa ukuran *file* hasil kompresi setelah didekompresi lagi hasilnya sama dengan ukuran *file* sebelum dilakukan proses kompresi.

File wave yang telah terkompresi hanya dapat dimainkan atau dijalankan dengan menggunakan aplikasi player yang telah yang dibuat.

Gambar 6. Form Player

Jika *file wave* yang terkompresi dimainkan atau dijalankan menggunakan windows media player maka akan muncul pesan "*Windows Media Player encountered an unknown error*" dan jika dijalankan menggunakan sebuah aplikasi PlayWave yang dibuat menggunakan Visual Basic 6 akan muncul pesan "*the specified format is not supported or cannot be translated. Use the capabilities function to determine the supported formats*". *File wave* yang telah dikompresi kemudian didekompresi lagi dapat dimainkan atau dijalankan di winamp dan media player lainnya. Kualitas suara *file wave* hasil kompresi yang didekompresi lagi secara subjektif, hampir sama dengan kualitas suara *file wave* sebelum dikompresi. Hal ini terjadi karena program tidak melakukan pengubahan pada *bitrate*, *samplerate*, dan jumlah *channel file wave* yang dikompresi. Kecepatan kompresi memang tidak dilakukan pengujian tetapi dari beberapa pengujian yang dilakukan tingkat kecepatan baik untuk proses kompresi dan dekompresi berbanding lurus dengan ukuran *file Wave*, artinya semakin besar ukuran *file Wave* yang diproses maka semakin lama proses berlangsung.

4. Kesimpulan

Hasil pembahasan di atas dapat disimpulan seperti berikut yaitu:

a. Metode Huffman dapat diterapkan dalam kompresi dan dekompresi *file wave* dengan rata-rata rasio kompresinya adalah 81,96%. Artinya ukuran *file wave* hasil kompresi berkurang 18,04% dari ukuran *file* aslinya.
b. Apabila isi *file* yang dikompress memiliki jumlah karakter unik mendekati 256 karakter dan banyak pola data yang mempunyai frekuensi kemunculan sama maka hasil kompresi kurang bagus.
c. Rasio kompresi pada *file wave* dengan ukuran sama dipengaruhi oleh jumlah *channel*, bit per *sample* dan *sample rate.*

d. Pada saat melakukan proses kompresi dan dekompresi pada *file wave* dengan ukuran besar, untuk setiap prosesnya memakan resource CPU rata-rata di atas 50 % dengan rata-rata *memory usage* 22.408 K sampai 41.264 K.
e. *File wave* yang sudah terkompresi dapat dimainkan atau dijalankan kembali oleh aplikasi *player* yang dibuat dengan cara didekompresi terlebih dahulu sebelum dimainkan atau dijalankan.

Daftar Pustaka

[1] Aburas, A.A., Rehiel,S.A (2008). Fingerprint patterns recognition system using huffman coding. *Proceedings of the World Congress on Engineering*, London: WCE,Vol III.

[2] Di Silvestro, L. L., Baribault, G. (1999). Waveform audio file format MIME sub-type registration <draft-ema-vpim-wav-00.txt>. Diakses dari http://64.170.98.42/html/draft-ema-vpim-wav-00.

[3] Goethals, A. (2004). Action plan background: AIFF / AIFF-C. Diakses dari http://www.fcla.edu/digitalArchive/pdfs/action_plan_bgrounds/aiff1_3_aiff-c1_0.pdf.

[4] Gunawan, I., Gunardi, K. (2005). Pembuatan perangkat lunak wave manipulator untuk memanipulasi file wav. Diakses dari http://puslit2.petra.ac.id/ejournal/index.php/inf/article/viewArticle/16318.

[5] Halvorson, M. (2000). Microsoft visual basic 6.0 profesional step by step. PT Elex Media Komputindo, Jakarta.

[6] Howe, D. (1993). Free on-line dictionary of computing. Diakses dari http://www.foldoc.org.

[7] Kabal, P. (2006). Audio file format specifications. Diakses dari http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html

[8] Merdiyan, M., Indarto, W. (2005). Implementasi algoritma run length, half byte dan huffman untuk kompresi file. Diakses dari http://journal.uii.ac.id/index.php/Snati/article/viewFile/1391/1171.

[9] Salomon, D. (2004). Data compression complete reference 3rd Edition. Springer, New York.

[10] Sayood, K. (2005). Introduction to data compression. Morgan Kaufmann, San Fransisco.

[11] Widhiartha, P. (2003). *Pengantar kompresi data*. Diakses http://ilmukomputer.org/wpcontent/uploads//2008/10/widhiartha_kompresidata.pdf

[12] Widyawardhana, A. (2004). Penggunaan metode huffman untuk decoding file audio terkompresi dalam format mp3 (Skripsi). *Fakultas Teknik Program Studi Teknik Informatika Universitas Kristen Duta Wacana*, Yogyakarta.