PERANCANGAN APLIKASI KOMPRESI PADA APLIKASI PEMUTAR AUDIO DENGAN MENGGUNAKAN METODE GOHECOTWO COMPRESSION

Misael Oktavianda Harefa[1], Rivalri Kristianto Hondro[2], Anda Yanny[3]
[1,2,3]Universitas Budi Darma, Medan
[1]misael031099@gmail.com, [2]rivalryhondro@gmail.com,
[3]andayanny31@gmail.com

ABSTRAK

Banyak orang menikmati mendengarkan musik di perangkat mobile mereka, hal ini dikutip dari kompas tekno. File audio dapat memakan banyak ruang penyimpanan, terutama file audio berkualitas tinggi. Permasalahan lain saat ini juga meskipun banyak aplikasi pemutar audio yang tersedia, namun masih sedikit aplikasi yang menyediakan fitur kompresi audio. Hal ini membuat pengguna kesulitan untuk mengompresi file audio mereka. aplikasi kompresi audio yang ada saat ini tidak banyak menggunakan metode kompresi yang optimal. Hal ini dapat mengakibatkan kualitas audio yang buruk atau tingkat kompresi yang rendah. Permasalahan lain saat ini juga meskipun banyak aplikasi pemutar audio yang tersedia, namun masih sedikit aplikasi yang menyediakan fitur kompresi audio. Hal ini membuat pengguna kesulitan untuk mengompresi file audio mereka. Selain itu, aplikasi kompresi *audio* yang ada saat ini tidak banyak menggunakan metode kompresi yang optimal. Hal ini dapat mengakibatkan kualitas *audio* yang buruk atau tingkat kompresi yang rendah. Salah satu solusi untuk mengatasi masalah ini adalah dengan menggunakan aplikasi kompresi audio Aplikasi kompresi audio dapat mengurangi ukuran file audio tanpa mengurangi kualitas audio secara signifikan Hal ini memungkinkan pengguna untuk menyimpan lebih banyak file audio di perangkat mobile mereka.

Kata Kunci : file audio, gohecotwo compression

ABSTRACT

Many people enjoy listening to music on their mobile devices, as cited from Kompas Tekno. Audio files can take up a lot of storage space, especially high-quality audio files. Another issue is that although there are many audio player applications available, only a few offer audio compression features, making it difficult for users to compress their audio files. The existing audio compression applications often do not utilize optimal compression methods, which can result in poor audio quality or low compression rates. Moreover, although many audio player applications are available, few provide audio compression features, making it hard for users to compress their files. Additionally, the current audio compression applications do not often use optimal compression methods, leading to poor audio quality or low compression efficiency. One solution to this problem is using an audio compression application that can reduce the size of audio files without significantly reducing the audio quality. This allows users to store more audio files on their mobile devices.

Keywords: audio files, gohecotwo compression

A. Pendahuluan

Saat ini, banyak orang menikmati mendengarkan musik di perangkat mobile mereka, seperti yang dikutip dari Kompas Tekno. Namun, file audio berkualitas tinggi dapat memakan banyak ruang penyimpanan. Masalah ini terutama dirasakan oleh pengguna perangkat mobile dengan kapasitas penyimpanan yang terbatas. Pengguna sering kali harus memilih antara kualitas audio atau jumlah file yang dapat disimpan, yang tentu saja menjadi dilema. Selain itu, meskipun banyak aplikasi pemutar audio tersedia, hanya sedikit yang menyediakan fitur kompresi audio. Hal ini menyulitkan pengguna yang ingin menghemat ruang penyimpanan dengan cara mengompresi file audio. Aplikasi kompresi yang ada juga kerap menggunakan metode yang kurang optimal, menyebabkan kualitas audio menurun atau tingkat kompresi yang tidak signifikan, sehingga manfaat kompresi menjadi terbatas (Purwanto, 2018). Aplikasi kompresi adalah perangkat lunak yang digunakan untuk mengurangi ukuran file digital, seperti audio, video, gambar, atau dokumen, dengan mengompresi data di dalamnya (Ardiyanto, 2012);(Ritonga, 2022). Tujuan utama dari aplikasi kompresi adalah untuk menghemat ruang penyimpanan dan mempermudah transfer file, tanpa mengorbankan kualitas atau dengan pengurangan kualitas yang minimal (Mulyanta, 2006).

Salah satu solusi untuk mengatasi masalah ini adalah dengan menggunakan aplikasi kompresi audio. Aplikasi kompresi audio adalah perangkat lunak yang dirancang untuk mengurangi ukuran file audio tanpa mengurangi kualitas suara secara signifikan (Situmorang, 2023);(Omeri, 2015);(Setiawan, 2023). Proses ini dilakukan dengan menggunakan algoritma kompresi yang bekerja untuk mengurangi jumlah data yang diperlukan untuk merepresentasikan audio. Aplikasi ini sangat berguna bagi pengguna yang ingin menghemat ruang penyimpanan di perangkat mereka, seperti smartphone, tablet, atau komputer, terutama ketika berurusan dengan file audio berkualitas tinggi yang biasanya memiliki ukuran besar (Munawar, 2023). Aplikasi ini dapat mengurangi ukuran file tanpa mengorbankan kualitas suara secara

signifikan (Syahfitri, 2024). Dengan aplikasi tersebut, pengguna dapat menyimpan lebih banyak file audio di perangkat mereka tanpa harus khawatir kehilangan kualitas suara. Penelitian oleh Panjaitan et al., (020) dengan judul “Penerapan Algoritma Gopala-Hemachandra Code2 (GH-2(n)) Pada Kompresi File Audio” juga menekankan pentingnya kompresi audio untuk efisiensi penggunaan ruang penyimpanan dan aksesibilitas audio yang lebih cepat tanpa mengurangi informasi penting.

Metode kompresi audio sangat beragam, dengan kelebihan dan kekurangan masing-masing. Salah satu metode yang menjanjikan adalah GoHeCoTwo Compression. Metode ini menawarkan tingkat kompresi yang lebih tinggi tanpa mengurangi kualitas audio. Penelitian yang dilakukan oleh Childers, (2020) dalam makalah berjudul "Gopala-Hemachandra Codes Revisited" menunjukkan bahwa metode ini mampu menghasilkan kompresi yang lebih efisien dengan tetap menjaga kualitas suara yang mendekati aslinya. Penerapan GoHeCoTwo Compression dalam aplikasi kompresi audio diharapkan dapat memberikan solusi yang efisien untuk masalah penyimpanan dan kualitas suara. Aplikasi ini memungkinkan pengguna untuk mengompresi file audio mereka dengan mudah, serta membantu mereka menghemat ruang penyimpanan tanpa harus mengorbankan kualitas suara.

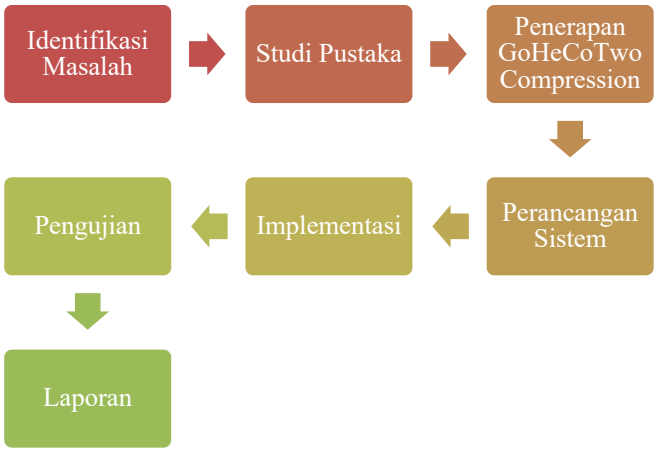
Dalam proses pengembangannya, aplikasi ini telah berhasil diintegrasikan dengan metode kompresi audio yang efisien, menghasilkan pengalaman pengguna yang lebih baik. Dengan kompresi audio yang optimal, pengguna dapat menyimpan lebih banyak file di perangkat mereka, yang tentu saja sangat membantu bagi mereka yang memiliki keterbatasan ruang penyimpanan. Secara keseluruhan, hasil dari penelitian ini memperlihatkan bahwa penerapan metode GoHeCoTwo Compression dalam aplikasi kompresi audio dapat menjadi solusi efektif untuk masalah kapasitas penyimpanan dan kualitas audio. Aplikasi ini memberikan kesempatan bagi pengguna untuk menikmati lebih banyak musik berkualitas tinggi tanpa khawatir kehabisan ruang.

Berdasarkan uraian penjelasan singkat latar belakang masalah di atas, maka berikut judul penelitian yang ditetapkan, sebagai berikut "Perancangan Aplikasi Kompresi Pada Aplikasi Pemutar Audio Dengan Menggunakan Metode *GoHeCoTwo Compression*".

B. Metode Penelitian

Kerangka Kerja Penelitian

Penelitian ini merupakan metode penelitian deskriptif yang bertujuan untuk memberikan gambaran, deskripsi, dan verifikasi terkait dengan perfoma penerapan *GoHeCoTwo Compression* dalam proses mengkompresi *file* data digital jenis audio. Berikut bentuk kerangka kerja penelitian yang dilakukan oleh penulis:



Gambar 1 Kerangka Kerja Penelitian

Berdasarkan diagram kerangka kerja penelitian, tahapan-tahapan yang dilakukan adalah sebagai berikut: Pada tahap identifikasi masalah, penulis memprediksi, menyimpulkan, dan menjelaskan akar penyebab masalah terkait kapasitas penyimpanan yang penuh untuk menciptakan lebih banyak ruang penyimpanan. Studi pustaka dilakukan untuk memahami objek penelitian melalui referensi dari buku, jurnal, dan sumber lainnya. Selanjutnya, penerapan GoHeCoTwo Compression dilakukan dengan kompresi bersifat lossless, dimana data dapat didekompresi kembali dengan hasil yang sama persis seperti data asli sebelum kompresi. Perancangan sistem mencakup pembuatan rancangan form aplikasi kompresi dan dekompresi menggunakan Matlab R2022a. Pada

tahap implementasi, file audio sebelum dan sesudah dikompresi dibandingkan, menunjukkan rasio perbandingan mencapai 50%. Pengujian dilakukan untuk menguji apakah aplikasi kompresi dan dekompresi sesuai dengan penerapan yang dirancang. Terakhir, dokumentasi menyimpan seluruh hasil pengujian dan perbandingan performa metode kompresi.

Sampel Data

Sampel data yang digunakan dalam penelitian ini adalah jenis *file* audio dengan ekstensi MP3. Adapun spesifikasi sampel data *file* audio tersebut dapat dilihat pada tabel berikut ini.

Tabel 1 Spesifikasi *File* Audio MP3

Nama *File*	Size	Type
File_Audio_1	1,51 MB	MP3

Tampilan *properties* berkas audio dengan nama berkas "File_Audio_1", sebagai data sampel yang digunakan dalam penelitian ini.

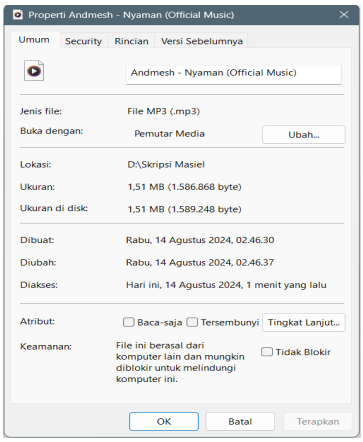
Gambar 2 Sample Data *File* Audio

C. Hasil Penelitian dan Pembahasan Penetapan Data Kriteria

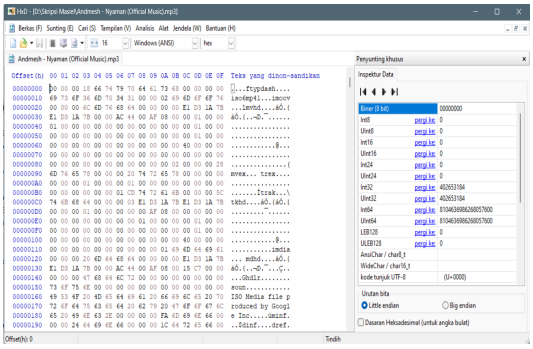
Penerapan Metode *GoHeCoTwo Compression*

Penelitian ini membahas proses kompresi dan dekompresi file audio menggunakan metode GoHeCoTwo Compression, yang merupakan salah satu teknik kompresi lossless. Untuk menganalisis file audio, langkah pertama adalah mengambil sampel file audio dalam format *.Mp3 dan membaca nilai hexadesimal dari data file tersebut. Pembacaan file *.Mp3 dilakukan untuk mengumpulkan data dalam bentuk bilangan hexadesimal. Proses manual pengambilan nilai sampel audio dilakukan menggunakan software HxD, kemudian data hexadesimal ini

dikompresi menggunakan metode GoHeCoTwo Compression. Penelitian ini akan menyajikan informasi lengkap mengenai objek file audio yang diambil sampelnya sebelum dilakukan proses kompresi.



Gambar 3 Sampel Data



Gambar 4 Hexadecimal

Tabel 2 Nilai Hexa yang belum dikompresi

No	Hexadecimal	Biner	Frekuensi	Bit	Bit*Freq
1	00	00000000	21	8	168
2	01	00000001	3	8	24
3	E1	11100001	1	8	8
4	D3	11010011	1	8	8
5	A1	10100001	1	8	8
6	78	01111000	1	8	8
7	AC	10101100	1	8	8
8	44	01000100	1	8	8
9	AF	10101111	1	8	8
10	08	00001000	1	8	8

Gambar 5 Tampilan Bilangan *Hexadecimal*

Kompresi Metode *GoHeCoTwo Compression*

Dalam melakukan analisa kompresi *file audio* dengan menerapkan Metode *GoHeCoTwo Compression* menggunakan sampel yaitu nilai *hexadecimal* dari *file audio* "sampel *file audio* *.Mp3" yang berisikan nilai *Hexadecimal* : E1 D3 1A 78 00 00 AC 44 00 AF 08 00 00 01 00 00 01 00 00 00 00 00 00 00 00 00 00 00 01 00 00= 32 *Byte*.

Jumlah	256 *bit*

Berdasarkan tabel, satu karakter bernilai delapan bit bilangan biner, sehingga 32 karakter pada string memiliki nilai biner sebanyak 256 bit. Metode GoHeCoTwo Compression bekerja dengan menghitung frekuensi kemunculan setiap karakter yang akan dikompresi. String bit sebelum dikompresi adalah: 00000000 00000001 11100001 11010011 10100001 01111000 10101100 01000100 10101111 00001000. Setelah menentukan kemunculan setiap karakter dan mencari nilai biner dari masing-masing karakter, langkah selanjutnya adalah mengurutkan karakter berdasarkan frekuensi kemunculannya. Bilangan biner pada setiap karakter kemudian diganti dengan kode kebenaran dari metode GoHeCoTwo Compression yang diurutkan berdasarkan frekuensi kemunculan tersebut.

Langkah selanjutnya adalah membentuk tabel kode Gopala-Hemachandra Code 2 (GH-2(n)). Aturan pembentukan kode ini didasarkan pada teori yang dijelaskan sebelumnya, yaitu dengan membentuk variasi angka Fibonacci orde kedua terlebih dahulu. Setelah itu, tabel kode Gopala-Hemachandra dapat dibuat sesuai dengan aturan yang telah ditetapkan. Proses ini membantu dalam menghasilkan kode yang akan digunakan dalam kompresi data, dengan rincian lebih lanjut dapat dilihat pada tabel yang disajikan:

Tabel 3 *Fibonacci* orde kedua

Deret Fibonacci	**F0**	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**
Fibonacci Orde Kedua	-2	3	1	4	5	9	14	23	37	60	97

Dimana
n = 1 (0011)
n = 2 (10011)
n = 3 (100011)
n = 4 (00011)
n = 5 (0000011)

Dimana
n =6 (001011)
n =7 (1000011)
n = 8 (010011)
n = 9 (00000)
n = 0010011

Pembentukan kode Fibonacci melibatkan penambahan angka "1" pada setiap akhir proses, yang diletakkan di posisi paling kanan dari

kode Fibonacci yang ada. Sebagai contoh, untuk angka \(n = 1 \) (F0), kode Fibonacci yang dihasilkan adalah "0011". Selanjutnya, kode Gopala-Hemachandra Code 2 (GH-2(n)) dibentuk berdasarkan kode Fibonacci ini, dengan rincian proses dapat dilihat pada tabel yang disajikan di bawah:

Tabel 4 Kode *Gopala-Hemachandra Code 2 (GH-2*(*n)*

n	**GH2(N)**
1	0011
2	10011
3	100011
4	00011
5	0000011
6	001011
7	1000011
8	010011
9	0000011
10	0010011

Hasil *encode* dari algoritma *GoHeCoTwo Compression* pada Nilai heksa “E1 D3 1A 78 00 00 AC 44 00 AF 08 00 00 01 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00” dapat dilihat pada tabel dibawah ini:

Tabel 5 Heksa yang sudah dikompresi dengan algortima *Gopala-Hemachandra Code 2*(GH-2(*n*))

Heksa	**Freq**	**Gopala-Hemachandra Code**	**Bit**	**Bit * Ferq**
00	21	0011	4	84
01	3	10011	5	15
E1	1	100011	6	6
D3	1	00011	5	5
A1	1	0000011	7	7
78	1	001011	6	6
AC	1	1000011	7	7
44	1	010011	6	6
AF	1	0000011	7	7
08	1	0010011	7	7
Total				150 *bit*

Tabel 6 *String bit* hasil kompresi dengan algoritma *GopalaHemachandra Code* 2 (*GH-2*(*n*)

E1	**D3**	**1A**	**7B**	**00**	**00**
100 011	000 11	000 001 1	001 011	001 1	00 11

AC	**44**	**00**	**AF**	**08**	**00**
100 001 1	010 011	001 1	000 001 1	001 001 1	00 11
00	**01**	**00**	**00**	**01**	**00**
001 1	100 11	001 1	001 1	100 11	00 11
00	**00**	**00**	**00**	**00**	**00**
001 1	001 1	001 1	001 1	001 1	00 11

00	**00**	**00**	**00**	**00**	**01**
0011	0011	0011	0011	0011	100111
00	**00**				
0011	0011				

Sebelum hasil akhir kompresi diperoleh, dilakukan penambahan padding bit dan flagging bit. Jika panjang string bit dibagi 8 menghasilkan sisa 0, tambahkan "00000001" di akhir bit. Jika sisa hasil bagi panjang string bit terhadap 8 adalah \(n = 1, 2, 3, 4, 5, 6, 7, 8, 9 \), tambahkan bilangan biner dari \(9 - n \) di akhir bit. Sebagai contoh, untuk string bit dengan panjang 150 bit, yang tidak habis dibagi 8 dengan sisa 6 bit, tambahkan 0 sebanyak \(7 - n + 1 \) di akhir string bit dan kemudian tambahkan bilangan biner dari \(9 - n \) di bit akhir untuk menyelesaikan padding dan flagging bit.

7-*n* + 1

7-6+1 =01

Bit akhir 9-6=3 =00000011

7 Perhitungan penambahan *bit*

1000 11	000 11	000 0011	001 011	001 1	00 11
1000 01	010 011	001 1	000 0011	001 001 1	00 11
1					
00 11	100 11	001 1	001 1	100 11	00 11
00 11	001 1	001 1	001 1	001 1	00 11
00 11	001 1	001 1	001 1	001 1	10 01 1
00 11	001 1	01 00000011			

Total panjang *bit* keseluruhan setelah ada penambahan *bit* adalah 150 + 7 + 8 = 165. Selanjutnya lakukan pemisahan *bit* menjadi beberapa kelompok. Setiap kelompok terdiri dari 8 *bit* seperti tabel di bawah ini:

Tabel 8 Pembagian *String bit*

100 011 00	011 000 00	110 010 11	001 100 11	100 001 10	100 110 01
100 000 11	001 001 10	011 001 11	001 100 11	001 110 01	100 110 01
100 110 01	100 110 01	100 110 01	100 110 01	100 110 01	110 011 00
110 011 01	000 000 11				

Berdasarkan pembagian kelompok nilai biner yang telah dilakukan,

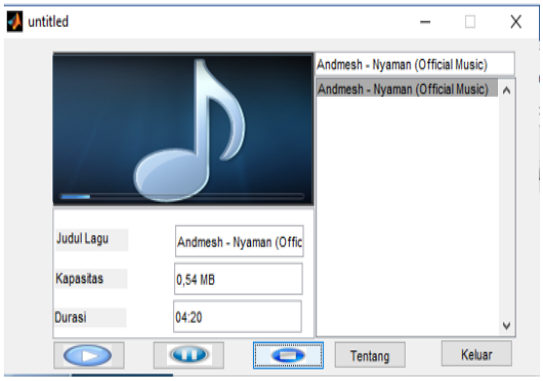
diperoleh nilai biner baru yang sudah terkompresi beserta nilai biner penambahan bit yang terdiri dari 20 kelompok. Setelah proses pembagian dan penambahan bit, nilai biner terkompresi tersebut dianalisis menggunakan Kode ASCII untuk mengidentifikasi nilai biner yang telah terkompresi. Informasi mengenai nilai biner yang sudah terkompresi dapat dilihat pada Tabel 9 di bawah ini.

Tabel 9 Nilai Desimal Terkompresi

No	Biner	Nilai Desimal	Karakter
1	10001100	142	Ş
2	01100000	96	‘
3	11001011	203	Ë
4	00110011	51	3
5	10000110	134	†
6	10011001	153	™
7	10000011	131	Ƒ
8	00100110	38	&
9	01100111	103	g
10	00110011	51	3
11	00111001	61	=
12	10011001	153	™
13	10011001	153	™
14	10011001	153	™
15	10011001	153	™
16	10011001	153	™
17	10011001	153	™
18	11001100	204	Ì
19	11001101	205	Í
10	00000011	3	

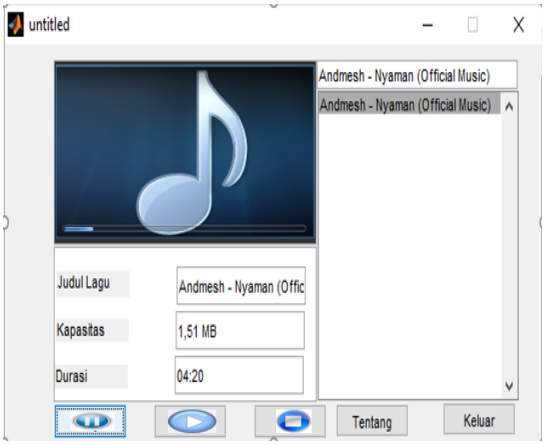
Hasil Pengujian

Pengujian sistem bertujuan untuk mengetahui bagaimana sistem yang telah dibangun akan berjalan. Sistem ini dirancang menjadi sesederhana mungkin agar pengguna dapat dengan mudah mengaksesnya. Dalam hal ini, penulis memberikan *print out* saat sistem sedang berjalan. Pada *form* ini akan menampilkan proses pengkompresian *file* audio kapasitas *.Mp3 menggunakan algoritma *GoHeCoTwo Compression*, yang dimana *user* menginputkan terlebih dahulu *file* audio yang akan dikompresi.



Gambar 6 Kompresi *File audio*

Pada *Form* dekompresi ini menampilkan *form* dimana sistem dapat melakukan proses dekompresi *file* audio yang telah di*input*. Setelah *file* audio didekompresi maka menghasilkan *file* audio terdekompresi.



Gambar 7 Aplikasi Dekompresi File Audio

Hasil pengujian Kompresi beberapa *file*

Tabel 10 Hasil Pengujian Kompresi *file*

	Sebelum Kompresi			Seseudah Kompresi		
	Nama *file*	Ukuran	Ekstensi	Nama file	Ukuran	CR
1.	Kpku	0,123 MB	*.docx	Kpku	0,020 MB	16 %

Berdasarkan hasil pengujian yang ditunjukkan dalam tabel di atas, terlihat bahwa ada perbedaan ukuran file sebelum dan setelah dikompresi. Perbedaan ini dapat diukur melalui Compression Ratio. Selain itu, kecepatan kompresi file dipengaruhi oleh ukuran awal file, di mana ukuran file yang lebih besar cenderung mempengaruhi kecepatan proses kompresi.

Hasil Pengujian Dekompresi *file*

Tabel 11 Hasil Pengujian Kompresi *file*

No	Sebelum dekompresi			Sesudah dekompresi	
	Nama *file*	Ukuran	Ekstensi	Nama file	Ukuran
1.	Kpku	0,020 MB	*.docx	Kpku	0,123 MB

D. Kesimpulan

Berdasarkan pembahasan dan evaluasi dari bab-bab sebelumnya, serta analisis terhadap kompresi file audio berformat *.Mp3, dapat disimpulkan bahwa prosedur kompresi menggunakan algoritma GoHeCoTwo Compression telah berhasil diterapkan dengan baik. Algoritma ini memungkinkan file audio dengan ukuran besar untuk dikompresi menjadi ukuran yang lebih kecil, sesuai dengan teknik kompresi yang diharapkan. Proses kompresi berjalan secara efektif, menunjukkan bahwa algoritma ini mampu menangani file audio dengan efisiensi tinggi tanpa mengorbankan kualitas secara signifikan. Penerapan algoritma GoHeCoTwo Compression dilakukan melalui aplikasi yang telah dirancang dan dibangun menggunakan software Matlab13, yang dirancang secara khusus untuk mendukung proses kompresi ini. Aplikasi tersebut terbukti berfungsi dengan baik, sehingga dapat mempermudah proses kompresi file audio yang berformat *.Mp3 dengan hasil yang memadai. Hal ini menunjukkan bahwa algoritma dan aplikasi yang dikembangkan memberikan solusi praktis dan efisien untuk kompresi audio, terutama bagi pengguna yang memerlukan penyimpanan lebih hemat tanpa harus kehilangan kualitas suara yang signifikan.

DAFTAR PUSTAKA

Ardiyanto. (2012). Kompresi Citra Dengan Source Coding Menggunakan Metode DM (Delta Modulation). In *(Doctoral dissertation, Universitas Muhammadiyah Surakarta)*.

Childers. (2020). Gopala-Hemachandra codes revisited. In *arXiv preprint arXiv:2004.00821 .*

Mulyanta. (2006). Dari Teori Hingga Praktik: Pengolahan Digital Image dengan Photoshop CS2 . In *Penerbit Andi.*

Munawar. (2023). Konsep Dasar Pengenalan Ilmu Komputer . In *Cendikia Mulia Mandiri*.

Omeri, N. (2015). Pentingnya Pendidikan Karakter Dalam Dunia Pendidikan. *Jurnal Ilmiah Manajemen Pendidikan Program Pascasarjana*, *9*(3), 464–468.

Panjaitan, S. M., Nasution, S. D., & Purba, B. (2020). Penerapan Algoritma Gopala-Hemachandra Code2 (GH-2 (n)) Pada Kompresi File Audio. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, *4*(1), 170–177.

Purwanto. (2018). Penerapan algoritma huffman pada kompresi file wave. *Jurnal Sistem Informasi*, *4*(1).

Ritonga, S. N. (2022). Implementasi Algoritma Golmb Code Dalam Kompresi File Video. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, *5*(1), 365–373.

Setiawan. (2023). PENDIDIKAN MULTIMEDIA: Konsep dan Aplikasi pada era revolusi industri 4.0 menuju society 5.0. In *PT. Sonpedia Publishing Indonesia.*

Situmorang, T. B. (2023). Perancangan Aplikasi Kompresi File MP3 Dengan Menggunakan Algoritma Lempel Ziv Welch (LZW) JURNAL MEDIA INFORMATIKA [JUMIN]. *JURNAL MEDIA INFORMATIKA [JUMIN]*, *5*(1), 11–21.

Syahfitri, E. (2024). Implementasi Algoritma Even-Rodeh dalam Mengkompresi File Audio. *VIRTUAL : Jurnal Teknik Informatika Dan Komputer*, *1*(2), 29–34.