IMPLEMENTASI ALGORITMA J-BIT ENCODING PADA KOMPRESI FILE TEKS

Tedi Mizwar[1], Guidio Leonarde Ginting[2], Mesran[2], Alwin Fau[2], Soeb Aripin[2], Dodi Siregar[3]

[1] Mahasiswa Program Studi Teknik Informatika STMIK Budi Darma, Medan, Indonesia
[2] Dosen Tetap STMIK Budi Darma, Medan, Indonesia
[3] Dosen Tetap STT Harapan, Medan, Indonesia

Abstrak

Data merupakan hal yang penting dan memiliki peranan yang sangat penting dalam kegiatan penggunaan komputer. Pada umumnya ukuran data yang berada pada saat ini memiliki ukuran yang sangat besar. Sehingga pada saat mengirim data yang berukuran besar melalui jaringan tentu akan mengakibatkan pemborosan pada penggunaan *resource* jaringan ketika data tersebut diakses oleh pengguna lain dan membuat media penyimpanan lebih kurang efisien dalam menyimpan data-data tersebut. Pada penelitian ini digunakan algoritma kompresi J-BIT untuk memperkecil ukuran file teks, sehingga akan mengurangi ukuran file pada media penyimpanan.

Kata kunci: *Kompresi, File Teks, J-BIT*

Abstract

Data is important and has a very important role in the activities of computer use. In general, the size of data that is at this time has a very large size. So at the time of sending large data through the network would lead to waste in the use of network resources when the data is accessed by other users and make the storage media more efficient in storing the data. In this study used J-BIT compression algorithm to reduce the size of text files, thus reducing the file size on storage media.

Keywords: *Compression, File Text, J-BIT*

1. PENDAHULUAN

Semakin berkembangnya teknologi pada komputer maka semakin banyak pula data-data atau *file-file* yang ingin kita simpan, tapi kadangkala kapasitas *memory* yang kita miliki tidak sebanding dengan data yang akan kita simpan. Oleh karena itu data-data yang akan disimpan perlu dikompres terlebih dahulu supaya ukurannya menjadi lebih kecil.

Data merupakan hal yang penting dan memiliki peranan yang sangat penting dalam kegiatan penggunaan komputer. Pada umumnya ukuran data yang berada pada saat ini memiliki ukuran yang sangat besar. Sehingga pada saat mengirim data yang berukuran besar melalui jaringan tentu akan mengakibatkan pemborosan pada penggunaan *resource* jaringan ketika data tersebut diakses oleh pengguna lain dan membuat media penyimpanan lebih kurang efisien dalam menyimpan data-data tersebut.

Kompresi merupakan pengurangan ukuran data menjadi ukuran yang lebih kecil dari aslinya. Pengkompresian data ini sangat menguntungkan manakala terdapat suatu data yang berukuran besar dan data di dalamnya mengandung banyak pengulangan karakter. Adapun teknik dari kompresi ini adalah dengan mengganti karakter yang berulang-ulang tersebut dengan suatu pola tertentu sehingga data tersebut dapat meminimalisasi ukurannya.

Masalah kompresi melibatkan proses bagaimana mengoptimalkan algoritma untuk menghapus berbagai redudansi dari suatu objek data. Terdapat beberapa permasalahan umum seperti jumlah bit total yang dikompresi, bagaimana mengembalikan ke bentuk string atau bit asli dan berapa banyak redundasi di ekstraksi dari data asli, pada penelitian skripsi ini penulis menggunakan algoritma J-Bit Encoding (JBE) untuk melakukan kompresi terhadap file teks.

J-Bit encoding (JBE) merupakan sebuah algorima yang mengoptimalkan masukan untuk algoritma kompresi lainnya [1]. Konsep utama dari algoritma J-Bit Encoding adalah membagi input yang ada kedalam 2(dua) buah bentuk data dimana bentuk pertama memiliki keseluruhan *zero byte*, dan bentuk kedua memiliki keseluruhan informasi *nonzero byte* [2].

Terdapat beberapa penelitian terkait dengan algoritma J-Bit Encoding, seperti yang dilakukan I Made Agus Dwi Suarjaya dengan judul penelitian *A New Algorithm for Data Compression Optimization* [2], hasil penelitian ini mengambil bentuk penelitian bagaimana algoritma J-Bit Encoding dikombinasikan dan dibandingkan dengan algoritma lain, dan dari hasil pengujian yang dilakukan algoritma J-Bit Encoding bisa dikombinasikan dengan baik untuk beberapa algoritma kompresi berbasis teks seperti LZW, LZ77, *Run Length Encoding* dan *Burrows-Wheeler*

2. LANDASAN TEORI

2.1 Kompresi

Kompresi memiliki tujuan untuk memperkecil ukuran data dengan mempertimbangkan kualitas data yang masih memadai untuk dapat dinikmati. Umumnya file memiliki banyak duplikasi data didalamnya, yaitu karakter-karakter atau binary yang

mempunyai intensitas kemunculan yang sama dalam setiap kata.

Terjadinya duplikasi data bisa memungkinkan apabila ukuran file data sangat besar. Semakin besar jumlah file data, maka bisa semakin kecil pula hasil file yang terkompresi terjadi [6].

2.2 Algoritma J-BIT

J-Bit *encoding* (JBE) merupakan algorima yang mengoptimalkan masukan untuk algoritma kompresi lainnya seperti algoritma LZW, *Run Length* dan *Half Byte* [8]. Konsep utama dari algoritma J-Bit *Encoding* adalah membagi *input* yang ada kedalam 2 (dua) buah bentuk data dimana bentuk pertama memiliki keseluruhan *zero byte*, dan bentuk kedua memiliki keseluruhan informasi *nonzero byte* [9].

Berikut adalah beberapa langkah untuk melakukan kompresi file teks dengan menggunakan algoritma J-Bit *Encoding* [9].

1. Baca *byte per byte* dari *file*
2. Tentukan nilai dari *byte* yang dengan *nonzero* atau *zero byte*
3. Tuliskan *nonzero byte* dari data I dan berikan bit '1' kedalam *temporary byte* atau berukan bit '0' into kedalam *temporary byte data* untuk nilai *zero*
4. Lakukan proses 1-3 sampai semua *temporary byte* berisi 8 bit *data*
5. Jika *temporary byte data* sudah terisi sebanyak 8 bit, lakukan penulisan *temporary data* untuk *byte* kedalam data II
6. Bersihkan t*emporary byte data*
7. Lakukan proses 1-6 secara berulang hingga akhir *byte* dari *file*
8. Lakukan proses penulisan dengan menggabungkan data I dan data II
   a. Tuliskan *original byte* dari *file* asli
   b. Tuliskan data I.
   c. Tuliskan data II.

3. ANALISA DAN PEMBAHASAN

Analisis sistem yang dibahas pada penelitian mengenai proses kompresi file teks dengan menggunakan algoritma J-Bit Encoding, proses pengujian kompresi dilakukan pada file teks dengan ekstensi *.txt (notepad).

Proses kompresi dengan menggunakan algoritma LZW, *Huffman*, *Deflate*, *Half Byte* dan beberapa algoritma kompresi lainnya membutuhkan proses kompresi yang rumit dan kompleks baik dari perhitungan manual hingga aplikasinya.

Untuk pengujian kompresi algoritma J-Bit *Encoding* penulis memberikan sampel kata sebagai berikut:

P = STMIK BUDIDARMA = 15 Byte

Kalimat ***P*** kemudian dirubah kedalam bentuk bilangan biner terlebih dahulu sebelum melakukan proses kompres, berikut adalah biner dari setiap karakter dari kalimat ***P***

Tabel 1. Biner P

No	Karakter	ASCII	Biner
1	S	083	0101 0011
2	T	084	0101 0100
3	M	077	0100 1101
4	I	073	0100 1001
5	K	075	0100 1011
6	SPACE	032	0010 0000
7	B	066	0100 0010
8	U	085	0101 0101
9	D	068	0100 0100
10	I	073	0100 1001
11	D	068	0100 0100
12	A	065	0100 0001
13	R	082	0101 0010
14	M	077	0100 1101
15	A	065	0100 0001

Proses berikutnya adalah menentukan *zero byte* dan *non zero byte* dari setiap karakter dari kalimat ***P***, berikut adalah proses kompresi untuk ***P***

Tabel 2. Proses Byte I

Original	Data I
0101 0011	
0101 0100	
0100 1101	
0100 1001	
0100 1011	000000000000000
0010 0000	1111101111111111
0100 0010	000001000000000
0101 0101	110000010000100
0100 0100	001110000100010
0100 1001	011000011010010
0100 0100	100010100000100
0100 0001	101110010101011
0101 0010	
0100 1101	
0100 0001	
120 bit	

Langkah berikutnya adalah menentukan nilai Bit 0 atau 1 yang dibaca untuk dimasukkan kedalam *temporary byte data*, dan pada kasus ini peneliti memilih byte 1, maka proses berikutnya adalah menulis semua bit 1 kedalam *non zero byte* dan hasilnya sebagai berikut dengan:

Data I: 000000000000000 1111101111111111 000001000000000 110000010000100

001110000100010 011000011010010
100010100000100 101110010101011 =**120 Bit**

Temporary Data Byte : 111110111111111 000001
110000010000100 001110000100010
011000011010010 100010100000100
101110010101011 = **96 Bit**

Temporary Data Byte :
11111011111111100000111000001000010 0
00111000010001001100001101001010001010000
0
100101110010101011

Kemudian *Temporary Byte Data* dibagi kedalam kelompok 8 bit untuk dimasukkan dan diproses kedalam Data II, dan hasilnya sebagai berikut:

Tabel 3. Bit Data II

Temporary Byte Data	Data II
	11111011
	11111110
	00001110
	00001000
111110111111111000011	01000011
100000100001000011100	10000100
001000100110000110100	01001100
101000101000001001011	00110100
10010101011	10100010
	10000010
	01011100
	10101011

Data II merupakan hasil akhir dari proses kompresi dari kalimat ***P*** = **STMIK Budidarma**, hasil biner ***J*** tersebut dirubah kedalam bentuk *ASCII* akan menghasilkan *encoding* teks sebagai berikut : ��C�L4��\�, hasil kompresi menjadi 12 *byte* dan hasil kompresi lebih kecil dari *file original* 15 *byte*, proses *decoding* adalah mengembalikan proses kedalam bentuk asli, langkah awal adalah merubah ��C�L4��\� kedalam bentuk biner dan hasilnya sebagai berikut:
11111011111111100001110000010000100001110 0
00100010011000011010010100010100000010010 11
10010101011
Kemudian berikutnya adalah merubah biner yang ada kedalam bentuk 15 bit dimulai dari sebelah kanan dan hasilnya sebagai berikut:
111101111111110 000100000000000
110000010000100 001110000100010
011000011010010 100010100000100
101110010101011

Setelah di konversi dengan melakukan proses data *byte* I dan data *byte* II seperti pada proses kompresi, maka hasil biner ditambahkan *zero byte* dan hasilnya sebagai berikut:

000000000000000 111101111111110
000100000000000 110000010000100
001110000100010 011000011010010
100010100000100 101110010101011

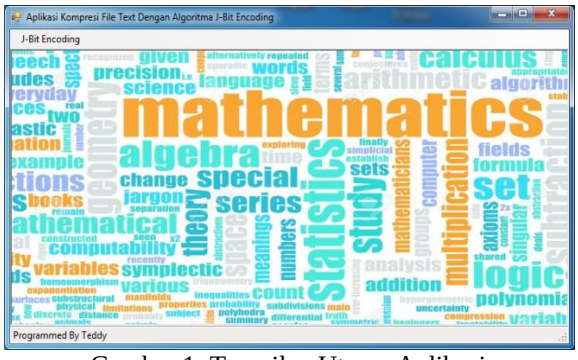
Ketika proses konversi ke *ASCII* dilakukan akan mendapatkan hasil sebagai berikut:

STMIK BUDIDARMA

4. IMPLEMENTASI

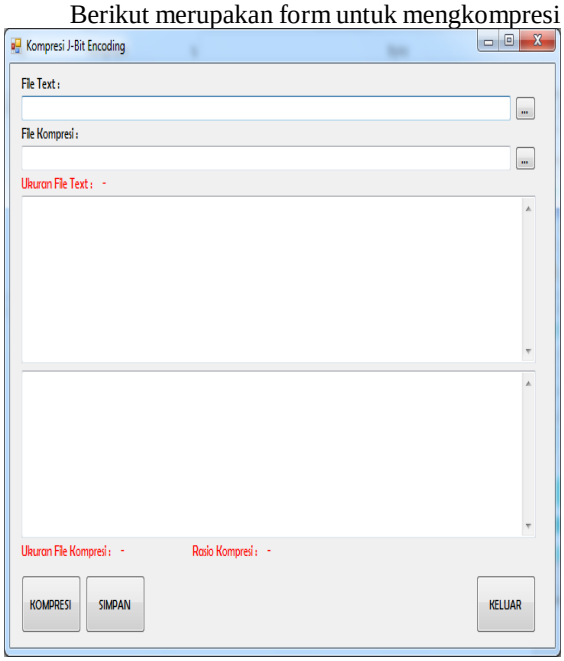
Pengujian program digunakan untuk melihat hasil yang sudah dibahas pada bab sebelumnya, berikut adalah beberapa tampilan aplikasinya

1. Form Utama
   Form utama merupakan form yang tampil ketika aplikasi dijalankan pertama sekali.



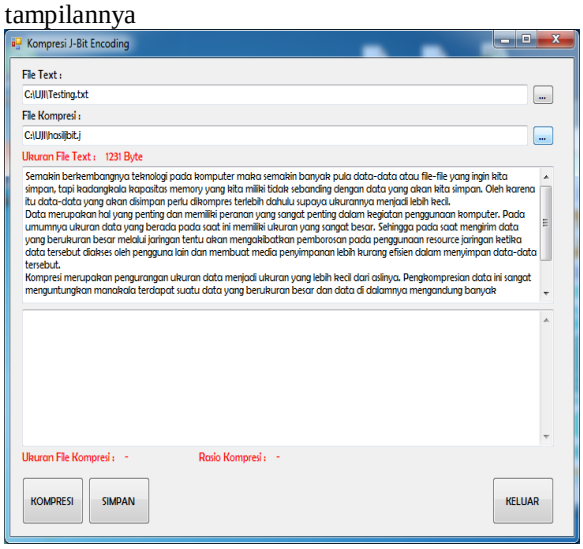
Gambar 1. Tampilan Utama Aplikasi

2. Form Kompresi
   Berikut merupakan form untuk mengkompresi



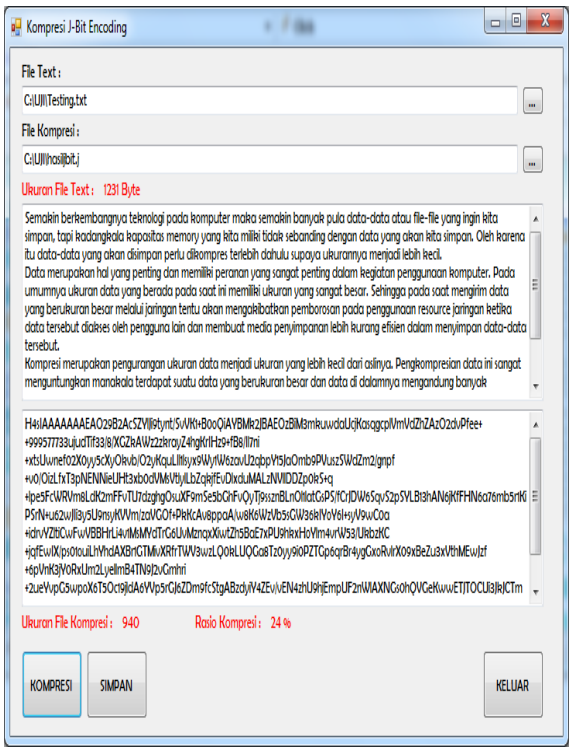
Gambar 2. Form Kompresi File

Gambar 2 diatas merupakan form kompresi yang digunakan untuk proses kompresi, langkah awal adalah menentukan lokasi *file* text dan lokasi *file* kompresi yang merupakan lokasi untuk menentukan dimana *file* hasil kompresi akan disimpan, berikut tampilannya



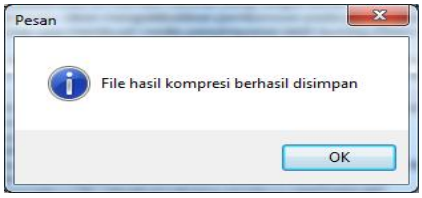
Gambar 3 Proses Pengujian Kompres

Gambar 3 merupakan tampilan hasil penentuan lokasi *file text* dan lokasi *file* kompresi, kemudian pada gambar juga tampak *plaintext* asli sebelum di kompresi, untuk melakukan proses kompresi dilakukan dengan menekan tombol kompresi dan hasilnya sebagai berikut:



Gambar 4 Hasil Proses Pengujian Kompresi

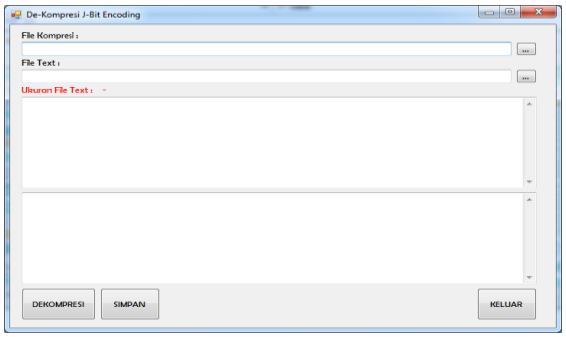
Hasil kompresi diatas menampilkan proses kompresi dari ukuran asli 1231 *Byte* menjadi 940 *byte* dengan ratio kompresi 24 %, *ratio* kompresi merupakan persentase kompresi yang berhasil dilakukan, untuk menyimpan hasil kompresi bisa dilakukan dengan menekan tombol simpan dan menampilkan pesan sebagai berikut:



Gambar 5. File Kompresi disimpan

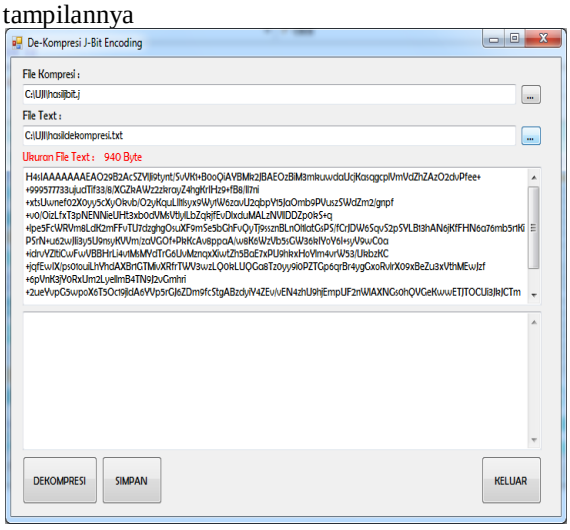
3. Form Dekompresi

Proses berikutnya adalah melakukan dekompresi dengan memilih menu J-Bit *Encoding* dan Dekompresi *file*, berikut tampilannya



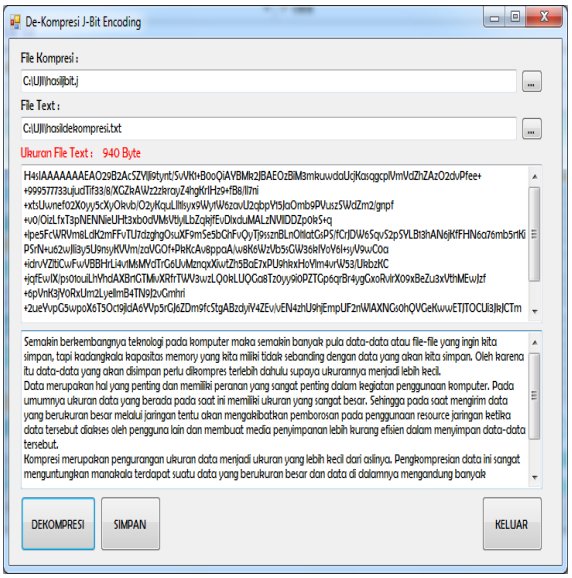
Gambar 6. Form Dekompresi

Gambar 6 merupakan form dekompresi yang digunakan untuk proses dekompresi, langkah awal adalah menentukan lokasi *file* kompresi dan lokasi *file text* yang merupakan lokasi untuk menentukan dimana *file* hasil dekompresi akan disimpan, berikut tampilannya



Gambar 7 Pengujian Dekompresi

Gambar 7 merupakan tampilan hasil penentuan lokasi *file* kompresi dan lokasi *file* dekompresi, kemudian pada gambar juga tampak *string* hasil kompresi sebelum di dekompresi, untuk melakukan proses dekompresi dilakukan dengan menekan tombol dekompresi dan hasilnya sebagai berikut:



Gambar 8. Hasil Pengujian Dekompresi

Gambar 8 merupakan hasil pengujian dekompresi dan hasilnya sesuai dengan yang diharapkan.

Pengujian kompresi dilakukan terhadap beberapa macam *file text* dengan konten yang berbeda, berikut adalah hasil pengujiannya.

Tabel 4. Hasil Pengujian

No	Jumlah Karakter	Ukuran Asli (Byte)	Hasil Kompresi (Byte)	Ratio Kompresi
1	1231	1231	940	24%
2	1094	1094	880	20%
3	3105	3105	2080	33%
4	4847	4847	2472	49%
5	12964	12964	6604	49%

Dari tabel uji di atas dapat dilihat bahwa semakin banyak karakternya maka proses kompresi akan semakin besar.

5. KESIMPULAN

Dari hasil penelitian diatas dapat disimpulan, yaitu:

1. Aplikasi kompresi yang dirancang mampu melakuan kompresi dengan baik terutama pada dokumen *text*
2. Sistem yang dirancang melakukan kompresi dengan menerapkan algoritma J-Bit *Encoding* tanpa harus dikombinasikan dengan algoritma lain seperti Huffman, LZW, LZ77
3. *Ratio* kompresi yang dihasilkan cukup baik, semakin besar *plaintext* maka rasio kompresi juga akan semakin tinggi.

REFERENSI

[1] Neti, "PERANCANGAN APLIKASI RENTAL MOBIL PADA CV KARYA BERSAMA PALEMBANG," *MDP*, pp. 1-10, 2008.

[2] Haviluddin, "Memahami Penggunaan UML (Unified Modelling Language)," *Informatika Mulawarman*, vol. 6, no. 1, pp. 1-15, 2011.

[3] H. Purwanto, "Penerapan Algoritma Huffman Pada Kompresi File Wave," vol. 2, no. 1, 2015.

[4] A. Satyapratama, Widjianto and M. Yunus, "Analisis Perbandingan Algoritma LZW dan HUFFMAN Pada Kompresi File Gambar BMP dan PNG," vol. 6, no. 2, 20115.

[5] S. Ambadekar, K. Gandhi, J. Nagaria and R. Shah, "Advanced Data Compression Using J-bit Algorithm," *International Journal of Science and Research (IJSR)*, vol. 4, no. 3, pp. 1366-1368, 2015.

[6] I. M. A. D. Suarjaya, "A New Algorithm for Data Compression Optimization," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 3, no. 8, pp. 14-17, 2012.

[7] Z. Amin, "DESAIN DAN IMPLEMENTASI TUNNELING IPSEC BERBASIS UNIX DENGAN ESP (ENCAPSULATING SECURITY PAYLOAD) (STUDI KASUS : PT. SUMEKS TIVI PALEMBANG & PT. SUMATERA EKSPRES)," *JURNAL TEKNOLOGI DAN INFORMATIKA (TEKNOMATIKA*, vol. 2, no. 2, pp. 116-126, 2012.

[8] S. Ambadekar, K. Gandhi, J. Nagaria and R. Shah, "Advanced Data Compression Using J-bit Algorithm," *International Journal of Science and Research (IJSR)*, vol. 4, no. 3, pp. 1366-1368, 2015.

[9] I. M. A. D. Suarjaya, "A New Algorithm for Data Compression," *International Journal of Advanced Computer Science and Applications* , vol. 3, no. 8, pp. 14-17, 2013.

[10] M. K. Grewal and S. Kaur, "Encryption and Compression of Audio-Video Data Using Enhanced AES and J-Bit Algorithm," *International Journal for Scientific Research & Development*, vol. 2, no. 8, pp. 159-165, 2014.

[11] S. D. Nasution, G. L. Ginting, M. Syahrizal and R. Rahim, "Data Security Using Vigenere Cipher and Goldbach Codes Algorithm," *International Journal of Engineering Research & Technology (IJERT)*, vol. 6, no. 1, pp. 360-363, 2017.

[12] S. D. Nasution and Mesran, "Goldbach Codes Algorithm for Text Compression," *IJournals Int. J. Softw. Hardw. Res. Eng.*, vol. 4, no. December, pp. 43–46, 2016.