Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 1, No. 1, Januari 2017, hlm. x-x http://j-ptiik.ub.ac.id

Implementasi Kompresi Data Dengan Menggunakan Zlib Data Compression dan Encoding Base64 Pada Sistem Paratransit Trip Data Collection Berbasis Esp32

Ilham Sentosa, Barlian Henryranu Prasetio, Aryo Pinandito

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: Ilhamsentosa@student.ub.ac.id, aryo@ub.ac.id, barlian@ub.ac.id

Abstrak

Dalam sejarah perkembangan manusia, pergerakan dari satu tempat ke tempat lain guna memenuhi kebutuhan telah menjadi aspek penting dalam kehidupan sehari-hari, termasuk di kota Malang yang memiliki karakteristik moda transportasi tidak teratur. Penelitian ini menyelidiki pengurangan lebar pita selama transmisi data, termasuk data GPS dan waktu, menggunakan teknik kompresi Zlib dan encoding Base64 untuk aplikasi seluler yang membutuhkan jaringan berbayar. Data dikirim melalui modul SIM800L dan ESP32 sebagai mikrokontroler. Hasilnya menunjukkan bahwa ukuran data yang dikirim berkurang 21% dari ukuran aslinya, namun memerlukan memori tambahan dan waktu pemrosesan hingga 4ms dalam sistem ESP32. Teknik ini dapat mengurangi biaya penggunaan jaringan berbayar dengan mengirim lebih banyak data dengan biaya yang sama atau lebih rendah, meskipun dengan peningkatan penggunaan memori dan waktu pemrosesan.

Kata kunci: *zlib compression, base64 encode, ESP32*

Abstract

Throughout human history, the movement from one place to another to meet needs has been an essential aspect of daily life, including in Malang, where transportation modes exhibit irregular characteristics. This study investigates the reduction of bandwidth during data transmission, including GPS and time data, using Zlib compression techniques and Base64 encoding for mobile applications requiring paid networks. Data is transmitted through the SIM800L module and ESP32 as the microcontroller. The results show that the data size sent is reduced by 21% from its original size, but it requires additional memory and processing time of up to 4ms in the ESP32 system. This technique can reduce the cost of using paid networks by transmitting more data at the same or lower cost, albeit with increased memory usage and processing time..

Keywords: *zlib compression, base64 encode, ESP32*

1. PENDAHULUAN

1.1 Latar Belakang

Dalam sejarah perkembangan manusia terhadap perkembangan kota dapat kita lihat bahwa manusia selalu berhasrat untuk berpergian dari suatu tempat ke tempat lain guna medapatkan keperluan yang dibutuhkan. Dalam hal ini manusia sangat membutuhkan suatu sarana transportasi yang disebut moda atau angkutan. Kebutuhan akan sarana transportasi dari waktu ke waktu terus mengalami peningkatan akibat semakin banyaknya kegiatan-kegiatan yang membutuhkan jasa transportasi sehingga bertambah pula intensitas pergerakan lalu lintas antar kota.

Moda atau angkutan di Malang memiliki karakteristik yang tidak teratur sendiri, sama seperti di kota-kota lain di dunia. Sulit untuk mengenali waktu operasional yang tepat, ketersediaan, dan estimasi waktu perjalanan saat menggunakan angkutan ini. Informasi tentang angkutan ini hanya dapat diperoleh dengan melakukan prediksi karena

karakteristik yang tidak teratur tersebut. (Kharisma et al., 2021)

Dalam melakukan transmisi data untuk mengenali waktu operasional yang tepat, ketersediaan, dan estimasi waktu perjalanan saat menggunakan angkutan ini dibutuhkan aplikasi seluler yang menggunakan jaringan berbayar. Oleh karena itu, mengurangi lebar pita yang diperlukan selama transmisi disarankan agar bermanfaat. Teknik kompresi data secara luas dikenal dapat mengurangi ukuran data, tetapi langkah tambahan ini dapat menghasilkan efek samping seperti peningkatan penggunaan memori dan waktu pemrosesan. Penelitian ini bertujuan untuk menyelidiki bagaimana proses kompresi data, yang khususnya menggunakan Zlib dan enkoding Base64, dapat menambah beban tambahan pada seluruh proses pengiriman data. (Pinandito et al., 2023)

Berdasarkan uraian paragraf tersebut mengurangi lebar pita data dengan teknik kompresi menggunakan Zlib dan encoding Base64 yang akan dikirim bermanfaat untuk mengurangi biaya yang diperlukan dalam penggunaan jaringan berbayar sehingga dapat mengirim dan menerima lebih banyak data dengan biaya yang sama atau bahkan lebih rendah

1.2 Rumusan Masalah

Dari penjelasan latar belakang tersebut, maka dapat dirumuskan suatu permasalahan sebagai berikut:

1. Berapa ukuran data yang ditransmisi setelah dilakukan kompresi data?
2. Berapa memori yang dibutuhkan ESP32 dalam melakuka kompresi dan encoding?
3. Berapa waktu yang dibutuhkan untuk melakukan proses kompresi data?

2. STUDI LITERASI

2.1 Metode Kompresi Zlib

1. Pengertian Kompresi Data

Kompresi data adalah proses mengurangi ukuran file atau data dengan cara menghilangkan redundansi tanpa menghilangkan informasi yang esensial. Tujuan utama dari kompresi data adalah untuk menghemat ruang penyimpanan dan mempercepat proses transmisi data. Kompresi data dapat dibagi menjadi dua kategori utama: kompresi lossy (kehilangan data) dan kompresi lossless (tanpa kehilangan data) (Sayood, 2012).

2. Kompresi Lossless

Metode kompresi lossless adalah teknik di mana data asli dapat direkonstruksi sepenuhnya dari data terkompresi tanpa kehilangan informasi. Contoh algoritma kompresi lossless termasuk Huffman coding, Run-Length Encoding (RLE), dan Lempel-Ziv-Welch (LZW). Metode kompresi lossless sering digunakan untuk teks, kode program, dan data lain yang tidak dapat mentoleransi kehilangan informasi (Sayood, 2012).

2.2 Encoding Base64

Base64 adalah metode encoding yang digunakan untuk mengonversi data biner menjadi representasi teks yang terdiri dari 64 karakter. Karakter-karakter ini mencakup huruf besar (A-Z), huruf kecil (a-z), angka (0-9), serta dua simbol tambahan, yaitu "+" dan "/". Base64 sering digunakan untuk mentransmisikan data biner melalui media yang hanya mendukung format teks, seperti email atau penyertaan data dalam URL (Josefsson, 2003).

Proses encoding Base64 bekerja dengan cara membagi data biner menjadi blok-blok 24-bit, yang kemudian dipecah menjadi empat kelompok 6-bit. Setiap kelompok 6-bit tersebut kemudian dipetakan ke salah satu dari 64 karakter yang tersedia. Jika data asli tidak cukup untuk membentuk blok 24-bit penuh, padding karakter "=" ditambahkan di akhir hasil encoding untuk memastikan panjang yang konsisten (Josefsson, 2003).

3. METODOLOGI PENELITIAN

3.1 Merancang Prototype dari Sebuah Sistem

Dalam penelitian ini dibutuhkan perangkat keras dan perangkat lunak antara lain:

Perangkat keras: ESP32, Gravity: GNSS

positioning module DFRobot, RTC (Real Time Clock) Sensor, Baterai 18650.
Perangkat lunak: Library Encode dan Decode Base64 dan Library Data Compression Zlib untuk ESP32.

3.2 Implementasi Teknik Kompresi ZLIB dan Encoding Base64

Untuk mengimplementasikan metode kompresi menggunakan library ZLIB pada mikrokontroler ESP32, pertama-tama kita perlu mengintegrasikan library ZLIB ke dalam proyek ESP32. Langkah awal adalah mengimpor library ZLIB dan melakukan konfigurasi awal untuk penggunaan kompresi data. Setelah itu, data yang ingin dikompresi diambil dan diproses menggunakan fungsi-fungsi kompresi yang disediakan oleh ZLIB. Setelah data berhasil dikompresi, tahap selanjutnya adalah meng-encode hasil kompresi tersebut menggunakan metode Base64 untuk memastikan data dapat ditransmisikan dengan aman dan kompatibel melalui berbagai protokol komunikasi. Proses encoding Base64 dapat dilakukan dengan menggunakan fungsi-fungsi yang tersedia di library standar Base64. Setelah data dikompresi dan di-encode, data siap untuk dikirim melalui berbagai saluran komunikasi modul GSM yang terhubung ke ESP32.

4. HASIL DAN ANALISIS

4.1 PENGUKURAN DATA LENGTH

Tabel ini menyajikan perbandingan panjang data antara dua format penyimpanan JSON: uncompressed dan compressed dengan tambahan base64 encoding. Pada kolom pertama, ditampilkan ukuran data dalam format JSON asli tanpa kompresi. Kolom kedua menunjukkan ukuran data setelah JSON tersebut dikompresi menggunakan algoritma kompresi yang efisien. Terakhir, kolom ketiga mencantumkan ukuran data setelah JSON yang telah dikompresi tersebut diubah menjadi format base64. Hasil perbandingan ini memberikan gambaran tentang efisiensi pengurangan ukuran data melalui proses kompresi dan encoding.

DATA	OUTPUT	LENGTH
JSON ENCODED UNCOMPRESSED DATA	"{\"d\":["{\"lat\":\"-7.2316166117111\",\"lng\":\"112.7964279259715\",\"vid\":\"0001\",\"vno\":\"18564001\",\"lid\":\"0001\"},{\"lat\":\"-7.9861934912960\",\"lng\":\"112.1649307967923\",\"vid\":\"0001\",\"vno\":\"18564001\",\"lid\":\"0001\"}]}";	201 Bytes
ZLIB COMPRESSION AND BASE64 ENCODE DATA	eJyVjcEKg0AMRP2UkrOVTHY3Mf5K6aFQKIXFXsSL+O9GPLQee50382ahJw2X20L1MdFAV+skQaEKGABqqY6vAIB05prFXIobSpD5HVti5r02j5+91hfNR1C/dG1//N4rPGWHuPLJD82eOF7iI/3lv6/NBjnVLfY=	160 Bytes

Table 1 Perbandingan ukuran data kompresi dan tanpa kompresi

Tabel ini menampilkan sampel data yang memperlihatkan tingkat persentase kompresi dari berbagai skenario. Kolom pertama menunjukkan ukuran data tanpa kompresi, sementara kolom kedua menampilkan waktu yang diperlukan untuk melakukan kompresi data tersebut. Kolom ketiga berisi ukuran data setelah proses kompresi selesai. Selanjutnya, kolom keempat mencantumkan rasio kompresi, yang dihitung dengan membandingkan ukuran data sebelum dan sesudah kompresi. Informasi dalam tabel ini memberikan wawasan mengenai efektivitas metode kompresi yang digunakan, menunjukkan berapa persen ukuran data dapat dikurangi

serta efisiensi waktu yang dibutuhkan untuk mencapai pengurangan tersebut.

JSON DATA LENGTH	GENERATION TIME	COMPRESSED LENGTH	COMPRESSION RATE
681	1	204	30%
777	1	220	28%
873	1	235	27%
969	1	254	26%
1065	1	274	26%
1161	1	292	25%
1257	1	310	25%
1352	1	328	24%
1449	1	347	24%
1545	1	364	24%
1641	2	383	23%
1737	1	400	23%
1833	1	418	23%
1929	2	436	23%
2025	2	452	22%
2121	2	469	22%
2217	2	487	22%
2313	2	505	22%
2409	2	522	22%
2505	2	541	22%
2601	2	559	21%
2697	2	577	21%
2793	2	591	21%
2889	2	613	21%
2985	2	631	21%
3081	2	646	21%
3177	3	665	21%
3273	3	683	21%
3369	3	704	21%
3465	3	719	21%

Table 2 Rasio Kompresi zlib

Hasil statistik dari analisis R menunjukkan perbandingan antara data tanpa kompresi dan data yang sudah terkompresi. Analisis ini menghasilkan nilai p-value < 2.2e-16, yang menunjukkan bahwa perbedaan antara kedua set data tersebut sangat signifikan secara statistik. Nilai p-value yang sangat kecil ini mengindikasikan bahwa kompresi data secara substansial mempengaruhi ukuran data. Kompresi data memberikan dampak yang signifikan dalam mengurangi ukuran data secara statistik.

```
        Wilcoxon signed rank test with continuity correction

data:  raw$ORI_SIZE and raw$COMP_SIZE
V = 5995, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

Gambar 1 hasil dari p-value antara data yang dikompresi dan tanpa kompresi

4.2 Waktu dan Memori Dalam Kompresi dan Enkode

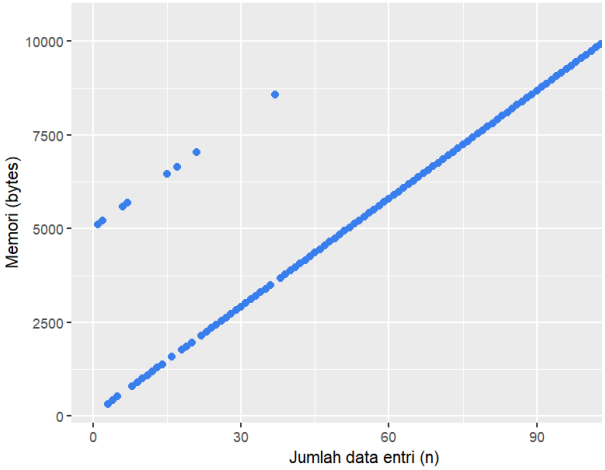
Tabel ini menampilkan penggunaan memori yang harus dipersiapkan selama proses kompresi dan encoding pada sistem ESP32. Dalam penelitian ini dilakukan penambahan objek dalam JSON. Kolom pertama menunjukkan jumlah objek JSON yang diproses, kolom kedua mencantumkan waktu yang dibutuhkan untuk kompresi dalam milidetik, kolom ketiga menunjukkan waktu yang dibutuhkan untuk encoding dalam milidetik, dan kolom terakhir menampilkan waktu total yang diperlukan untuk kedua proses tersebut. Data dalam tabel ini memberikan informasi penting mengenai durasi waktu yang perlu disiapkan untuk menjalankan proses kompresi dan encoding secara efektif pada sistem ESP32.

JSON OBJECT	COMPRESSED MEMORY	ENCODE MEMORY	TOTAL MEMORY
1	5116	144	5260
2	5212	180	5392
3	328	204	532
4	424	228	652
5	520	244	764
6	5596	272	5868
7	5692	292	5984
8	808	316	1124
9	904	336	1240
10	1000	360	1360
11	1096	388	1484
12	1192	412	1604
13	1288	436	1724
14	1384	460	1844
15	6460	484	6944
16	1576	508	2084
17	6652	532	7184
18	1768	556	2324

19	1864	580	2444
20	1960	604	2564

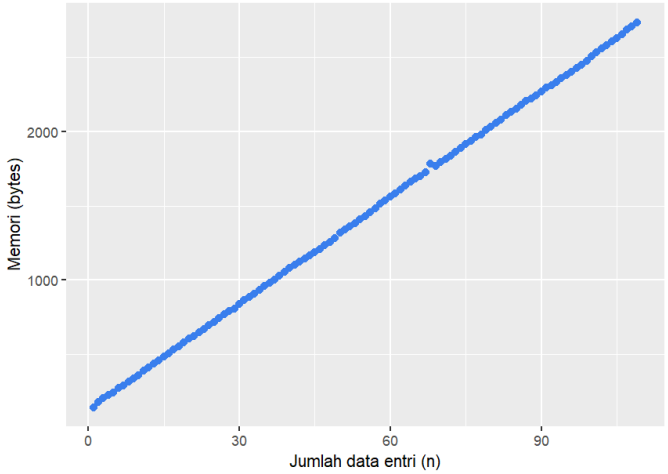
Table 3 Penggunaan Memori dalam proses kompresi dan encode

Grafik ini menampilkan penggunaan memori oleh sistem ESP32 selama proses kompresi menggunakan zlib compression. Sumbu horizontal pada grafik menunjukkan waktu atau tahapan proses kompresi, sedangkan sumbu vertikal menunjukkan jumlah memori yang digunakan dalam bytes. Data dalam grafik memperlihatkan fluktuasi penggunaan memori selama berbagai tahap kompresi, mulai dari inisiasi proses, kompresi data, hingga menghasilkan keluaran data.



Gambar 2 penggunaan memori dalam proses kompresi berbanding dengan banyak data

Grafik ini menampilkan penggunaan memori oleh sistem ESP32 selama proses encoding. Sumbu horizontal pada grafik menunjukkan waktu atau tahapan proses encoding, sedangkan sumbu vertikal menunjukkan jumlah memori yang digunakan dalam bytes. Data dalam grafik memperlihatkan fluktuasi penggunaan memori selama berbagai tahap proses encoding, mulai dari inisialisasi hingga menghasilkan output data.



Gambar 3 penggunaan memori dalam proses encode berbanding dengan banyak data

Tabel ini menampilkan perbandingan waktu yang dibutuhkan untuk proses pengiriman data antara dua skenario: tanpa kompresi dan dengan kompresi + encode. Kolom pertama menujukan jumlah JSON objek. Kolom kedua menunjukkan waktu pengiriman data tanpa kompresi dalam milidetik. Kolom ketiga mencantumkan waktu yang dibutuhkan untuk pengiriman data dengan kompresi dan encoding dalam milidetik. Kolom keempat menunjukkan penambahan waktu yang dibutuhkan untuk proses kompresi dan encoding.

JSON OBJECT	UNCOMPRESSED TIME	COMPRESSED + ENCODE	ADDITIONAL TIME
1	364	365	1
2	418	419	1
3	536	537	1
4	857	859	2
5	363	364	1
6	383	384	1
7	495	497	2
8	389	391	2
9	682	683	1
10	277	279	2
11	488	490	2
12	466	468	2
13	796	798	2
14	578	580	2
15	515	517	2

16	566	568	2
17	567	570	3
18	310	312	2
19	463	465	2
20	832	836	4

Table 4 Tabel waktu yang dibutuhkan dalam proses kompresi

Hasil analisis R ini membandingkan waktu tambahan yang dibutuhkan oleh ESP32 untuk memproses kompresi dan encoding data sebelum pengiriman, dibandingkan dengan waktu pengiriman data tanpa kompresi. Analisis ini menghasilkan p-value < 2.2e-16, yang menunjukkan bahwa perbedaan waktu antara kedua kondisi tersebut sangat signifikan secara statistik. Nilai p-value yang sangat kecil ini mengindikasikan bahwa proses kompresi dan encoding mempengaruhi waktu pemrosesan secara signifikan pada ESP32. Perbandingan ini menyoroti dampak waktu tambahan yang dibutuhkan untuk kompresi dan encoding dalam konteks efisiensi pengiriman data.

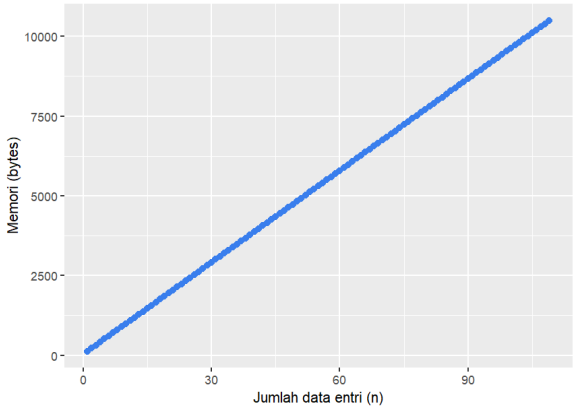
```
        Wilcoxon signed rank test with continuity correction

data:  raw$TIME_SEND_COMP_JSON and raw$TIME_SEND_UNCOMP_JSON
V = 5995, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

Gambar 4 hasil dari p-value dari waktu yang dibutuhkan antara dengan proses kompresi dan tanpa kompresi

4.3 Waktu dan Memori Dalam Dekode dan Dekompresi

Proses decoding dan dekompresi data yang diterima dari server oleh ESP32 melibatkan dua tahapan kritis: decoding data yang telah di-encode dan kemudian mendekompresi data tersebut untuk mengembalikannya ke bentuk asli. Selama proses decoding, ESP32 menggunakan sejumlah memori yang bervariasi sesuai dengan kompleksitas dan ukuran data yang diterima. Grafik penggunaan memori menunjukkan fluktuasi selama berbagai tahap decoding, mulai dari inisialisasi hingga penyelesaian. Tahap ini penting untuk memastikan bahwa data yang diterima dapat diubah kembali menjadi format yang dapat digunakan oleh aplikasi, dengan penggunaan memori yang harus diatur secara efisien untuk menghindari kelebihan beban.

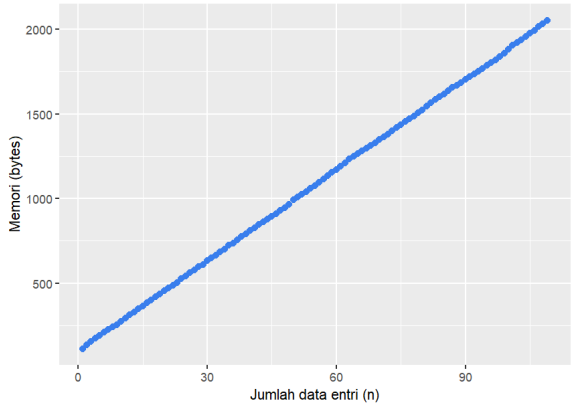
Setelah data berhasil di-decode, langkah selanjutnya adalah dekompresi. Proses ini membutuhkan alokasi memori tambahan dan waktu pemrosesan yang signifikan, terutama jika data yang diterima berukuran besar atau telah dikompresi secara kompleks. Grafik memori menunjukkan pola penggunaan memori selama tahapan dekompresi, yang biasanya lebih tinggi dibandingkan tahap decoding karena kompleksitas algoritma dekompresi. Penggunaan waktu selama proses ini juga diukur untuk memastikan efisiensi operasional ESP32. Analisis ini penting untuk mengoptimalkan performa ESP32 dalam menerima dan memproses data dari server, memastikan bahwa kedua proses tersebut dapat dilakukan dengan kecepatan dan efisiensi yang memadai.
Grafik ini menampilkan penggunaan memori oleh ESP32 selama proses decoding data yang diterima oleh server. Sumbu horizontal pada grafik menunjukkan waktu atau tahapan proses decoding, sedangkan sumbu vertikal menunjukkan jumlah memori yang digunakan dalam bytes. Data dalam grafik memperlihatkan fluktuasi penggunaan memori selama berbagai tahap proses decoding, mulai dari inisialisasi hingga penyelesaian decoding.



Gambar 5 Memori yang dibutuhkan dalam proses dekompresi

Grafik ini menampilkan penggunaan memori oleh ESP32 selama proses dekompresi data setelah proses decoding. Sumbu horizontal pada grafik menunjukkan waktu atau tahapan proses dekompresi,

sedangkan sumbu vertikal menunjukkan jumlah memori yang digunakan dalam bytes. Data dalam grafik memperlihatkan fluktuasi penggunaan memori selama berbagai tahap proses dekompresi, mulai dari inisialisasi hingga penyelesaian dekompresi.



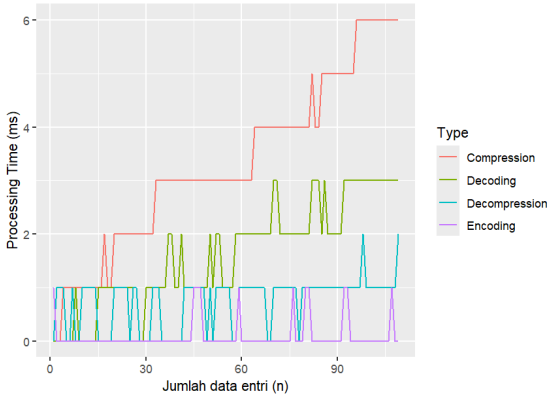
Gambar 6 Penggunaan memori dalam proses decode

Tabel ini menampilkan waktu yang dibutuhkan oleh ESP32 untuk memproses data yang diterima dari server, meliputi proses decoding dan dekompresi. Kolom pertama menunjukkan jumlah objek JSON yang diproses. Kolom kedua mencantumkan waktu yang diperlukan untuk mendecode data dalam milidetik. Kolom ketiga menampilkan waktu yang dibutuhkan untuk mendekompresi data yang telah di-decode dalam milidetik. Kolom keempat menunjukkan waktu total yang dibutuhkan untuk menyelesaikan kedua proses tersebut. Data dalam tabel ini memberikan gambaran yang jelas mengenai durasi waktu yang diperlukan oleh ESP32 untuk mengubah data yang diterima dari server ke dalam bentuk yang dapat digunakan oleh aplikasi.

JSON OBJECT	DECODING TIME	DECOMPRESSION TIME	TOTAL TIME
1	0	0	0
2	0	1	1
3	0	1	1
4	0	1	1
5	0	0	0
6	0	0	0
7	0	1	1
8	1	0	1
9	0	0	0
10	0	1	1
11	0	1	1
12	0	1	1
13	0	1	1
14	0	1	1
15	1	0	1
16	1	0	1
17	1	0	1
18	1	0	1
19	1	0	1
20	1	1	2

Table 5 Total waktu yang dibutuhkan dalam proses decode dan decompress

Grafik ini menampilkan waktu yang digunakan untuk empat proses utama: kompresi, encode, decode, dan dekompresi. Sumbu horizontal pada grafik menunjukkan masing-masing proses, sementara sumbu vertikal menunjukkan waktu yang dibutuhkan dalam milidetik. Data dalam grafik ini menggambarkan durasi waktu yang diperlukan untuk setiap proses, mulai dari kompresi dan encoding data hingga decoding dan dekompresi data yang diterima.



Gambar 7 Penggunaan waktu dalam tiap proses kompresi, encode, decode dan dekompresi

5. KESIMPULAN

Berdasarkan hasil yang telah dilakukan dalam penelitian ini, dapat menjawab rumusan masalah sebagai berikut:

1. Ukuran data yang dikirim 21% lebih kecil dari data aslinya, ukuran data dalam proses kompresi ini bervariasi semakin kecil data maka rasio kompresi akan semakin turun.
2. Dibutuhkan memori tambahan untuk

melakukan proses kompresi encode decode dan dekompresi sebesar ukuran data yang dikompres.

3. Dibutuhkan penambahan waktu untuk proses kompresi encode decode dan dekompresi hingga 4ms dalam sistem ESP32 dan akan semakin besar waktu yang dibutuhkan ketika jumlah data yang dikirim semakin banyak.

6. DAFTAR PUSTAKA

Rahmat, H., Damiri, D. J., & Susanto, A. (2012). Implementasi Kompresi Data pada Jaringan Komputer Menggunakan Algoritma Zlib. *Jurnal Algoritma Sekolah Tinggi Teknologi Garut*, 9(1), 119-124. Retrieved from http://jurnal.sttgarut.ac.id [18].

Pinandito, A., Kharisma, A. P., & Jonemaro, E. M. A. (2023). Architectural Design of Representational State Transfer Application Programming Interface with Application-Level Base64-Encoding and Zlib Data Compression. *Journal of Information Technology and Computer Science*, 8(3), 286-298. Retrieved from https://www.jitecs.ub.ac.id [19].

Kharisma, A. P., Jonemaro, E. M. A., & Arwani, I. (2021). Paratransit Trip Data Collection System with Smartphone GPS and REST Web Service in Malang, Indonesia. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 9(1), 245-255. Retrieved from http://section.iaesonline.com/index.php/IJEEI/index.

Josefsson, S. (2003). RFC4648: The Base16, Base32, and Base64 Data Encodings. IETF. [Online]. Available: https://tools.ietf.org/html/rfc4648

Gailly, J.-l., & Adler, M. (1996). zlib: A Massively Spiffy Yet Delicately Unobtrusive Compression Library. [Online]. Available: https://zlib.net/

Sayood, K. (2012). Introduction to Data Compression (4th ed.). Morgan Kaufmann.

Deutsch, P. (1996). RFC1951: DEFLATE Compressed Data Format Specification version 1.3. IETF. [Online]. Available: https://tools.ietf.org/html/rfc1951

Nelson, M., & Gailly, J.-l. (1996). The Data Compression Book (2nd ed.). M&T Books.