IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.13, No.4, October 2019, pp. 345~356
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: https://doi.org/10.22146/ijccs.47270

The Development of IoT Compression Technique To Cloud

Kartika Sari*[1], Mardhani Riasetiawan[2]
[1]Master Program of Computer Science, FMIPA UGM, Yogyakarta, Indonesia
[2]Department of Computer Science and Electronics, FMIPA UGM, Yogyakarta, Indonesia
e-mail: *[1]**kartika.sari.16@mail.ugm.ac.id,** [2]mardhani@ugm.ac.id

Abstrak

Masalah utama pengiriman data adalah bagaimana mengurangi panjang pengiriman paket data, sehingga dapat mengurangi waktu pengiriman data. Salah satu metode yang dapat digunakan untuk mengurangi ukuran data adalah dengan mengompresi ukuran data. Kompresi data adalah teknik untuk mengompresi data untuk mendapatkan data dengan ukuran yang lebih kecil dari ukuran aslinya sehingga dapat mempersingkat waktu pertukaran data.

Penelitian ini bertujuan untuk mengembangkan teknik kompresi data dengan memodifikasi dan menggabungkan teknik pengkodean dan pemodelan berdasarkan algoritma RAKE. Percobaan pengujian penelitian ini menggunakan 4 metode yang berbeda dalam 5 periode waktu yang berbeda untuk menentukan nilai kompresi, parameter efisiensi dekompresi, dan parameter waktu transmisi data.

Hasil dari penelitian ini adalah teknik pengkodean data yang menggunakan desimal ke data konverter biner dan teknik pemodelan dengan menghitung residu dari nilai sensor akan menghasilkan data dalam ukuran kecil dan mendapatkan nilai efisiensi kompresi 45%. Untuk teknik pengkodean menggunakan ASCII dan teknik pemodelan dengan operasi XOR akan menghasilkan data ukuran yang lebih besar dan nilai efisiensi kompresi 71%. Dalam pengujian dekompresi data, nilai efisiensi dekompresi 100%, tidak ada kehilangan data.

Kata kunci*— Cloud, Internet of Things, Kompresi Data, Transmisi Data*

Abstract

The main problem of data transmission is how to reduce the length of data packet delivery, so it can reduce the time of sending data. One method that can be used to reduce the data size is by compressing the data size. Data compression is a technique for compressing data to get the data with smaller size than the original size so that it can shorten the data exchange time

This study aims to develop the data compression techniques by modifying and combining the coding and modelling techniques based on the RAKE algorithm. This study testing experiments use 4 different methods in 5 different time-periods to determine the value of the compression, decompression efficiency parameters, and the data transmission time parameters.

The result of this study is the data coding technique that using decimal to binary converter data and the modeling technique by calculating the residue from the sensor value will produce data in small sizes and get a compression efficiency value of 45%. For coding techniques using ASCII and modeling techniques with XOR operations will produce bigger size data and the compression efficiency value of 71%. In testing data decompression, the decompression efficiency value of 100%, there is no data loss.

Keywords*—Cloud, Internet of Things, Data Compression, Data Transmission.*

Received July 3rd,2019; Revised July 17th, 2019; Accepted July 26th, 2019

1. INTRODUCTION

The main problem of data transmission is how to reduce the length of data packet delivery, so it can reduce the time of sending data. One method that can be used to reduce the data size is by compressing the data size[1]. The more data stored and in a long period, the higher the size of the data. The solution that can be done is to reduce the size (compression) of the data [4].

The algorithm that can be used for IoT data compression is a lossless RAKE compression algorithm, this is because lossless algorithms are more widespread in some IoT scenarios [5]. Although several other lossless compression algorithms are available. For example, the Lempel-Ziv algorithm, most are not suitable when only limited storage and computing resources are available [6]. Also, the implementation of this RAKE algorithm only requires basic calculation operations, limited overhead (only a few extra bits are needed to encode all required parameters for decompression), centralized storage (not dependent on previous codewords) and this algorithm can easily be extended to a sequence of integers [4].

G-Connect Project is one of the use of Cloud-based systems and Internet of Thing (IoT), while G-Connect project one of the community service projects at Computer Science Department of Universitas Gadjah Mada with the goal is to provide independent information access into a small area with limited infrastructure support and also areas with high levels of disaster vulnerability. This project applied the disaster information remote monitoring and monitoring technology by implementing IoT and cloud technology to help disaster-prone areas. In 2016, the research phase and prototype development to limited trials were carried out through an independent research scheme. The results of the technology that supports this connectivity are in the form of a mobile access device consisting of several components including hot air balloon components, solar panels as power supply components, power banks as power storage components, SBC mini routers as access control components and wireless as a spreader component access [7].

The problem faced by the G-connect project at this time is the signal limitations and the absence of a method to reduce the size of sensor data that will be sent to the cloud. The more data, the higher the size of the data sent, so that information access will not be fast because it needs more time to transmit data. For this reason, a method is required in order to reduce the size of the data. Based on the explanation above, in this study development will be carried out in coding techniques and modeling a method of compression and decompression of data using the RAKE algorithm. This study aims to compress and reduce the time of sending data obtained from IoT G-Connect devices to the cloud

2. METHODS

An analysis of the minimum device requirements is recommended for building the system. In this study used several supporting equipment consisting of hardware and software.

2.1 System in General

The focus of this research is on two subsystems which are then integrated into a whole system, namely the IoT gateway subsystem (Raspberry Pi) and the cloud platform subsystem (Digital Ocean Server). Figure 2 is a general diagram of the system to be built:

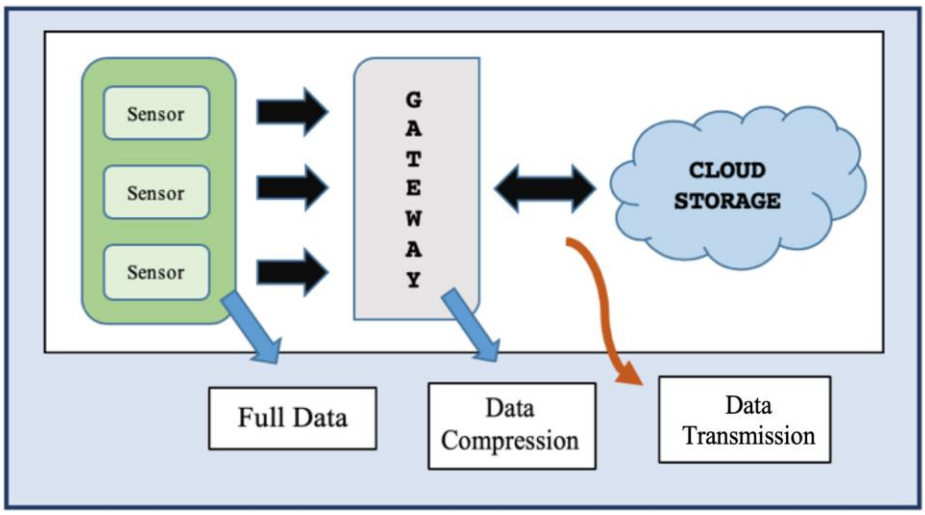

Figure 1 General System

In this system, there are several sensor nodes to collect data from the real environment. Sensors are connected to the local gateway on the client-side, reporting values obtained from all sensors continuously and or according to the pre-arranged time, and then the data is sent to the cloud platform. The gateway on the system is responsible for managing IoT devices, receiving sensor data and then compressing sensor data before data is sent to the cloud. While the cloud platform is responsible for storing device data using a service interface provided by the gateway so that users can store device data sent by the gateway and decompress data using the RAKE algorithm.

This research is part of the G-Connect project which comes with the main concept of providing access to information that is independent of a region and implements the IoT devices and Cloud technology to help disaster-prone areas. G-Connect device record data from the environment using SHT11 (Temperature and humidity), GY-521 MPU 6050 (x, y, z movement) and Anemometer (wind speed. Apart from sensors, G-Connect also uses Arduino and Raspberry Pi. The use of Arduino on G-Connect functions as a hardware interface that deals directly with sensor devices, while the Raspberry Pi functions to request sensor data to Arduino which will then be sent to the cloud. Data is recorded at any time and sent to the cloud server regularly. Then the collected data is presented in the form of time series information and can be analyzed data on temperature, humidity, wind speed and symptoms of symptomatic land movement in landslides. Information on the cloud server can be accessed to provide an early warning on the scope of the region. Figure 1 shows the G-Connect project which is divided into seven main parts: Communication device between Arduino and Raspberry Pi, The operating system and scheduling on Raspberry Pi, Scheduling data sensor to cloud, The data compression and data transmission on Raspberry Pi to cloud, extracting shipping data, correct data and validating data on the cloud.

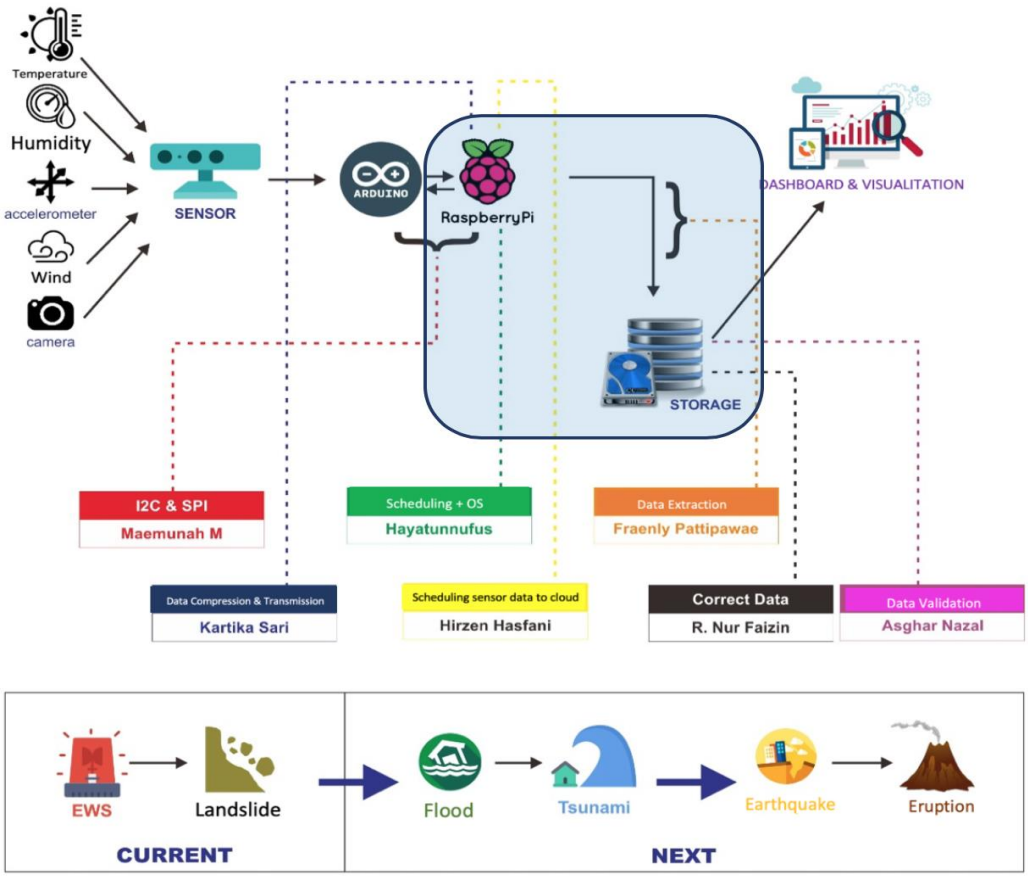

Figure 2 G-Connect Project Schemes

2.2 System Requirement

Table 1 is the general requirements for software that used in the development of the RAKE algorithm for compression and transmission of IoT G-Connect sensor data to the cloud.

Table 1
General requirements for Software and Hardware

Software		Hardware
OS at running Client	Mac OS X	Raspberry Pi
OS at server	Ubuntu Server	Arduino
OS at Raspberry Pi	Raspbian	SHT11
Language Programming on Raspberry Pi	Phyton	MPU6050
Arduino Compiler	Arduino IDE 1.8.2	Anemometer
Cloud Server	Digital Ocean Server	Wifi
OS at Smartphone	Android	
Hotspot App (getting the IP)	Hotspot manager	

2.3 The Compression and the Decompression System Stages in General

The stages of the data compression delivery chart based on the RAKE algorithm can be explained as follows:

a. The first step is to take the timestamp and a number/package of sensor data received by the time of data collection.
b. Then the data coding process is carried out, namely by converting sensor data values.
c. Check if the data taken is the value / first set of bits. If yes, then a set of bits is sent and saved to the cloud as the default which functions as an identity if data will be retrieved and decompressed.
d. If the value is not the first set of bits (set to 1 + n, and so on), then the default bit values that are on the cloud will be retrieved.
e. The next step is checking whether there is still sensor data after that that must be taken. If not, it will proceed to the compression process resulting from data modeling. If there is still sensor data, it will proceed to the timestamp value retrieval process and sensor data collection.
f. Then the compression of binary numbers results from the modeling process using the RAKE algorithm.

After the value of the sensor data is successfully compressed, the next step is the process of sending compressed data to the cloud. Then the data decompression process will be carried out in the cloud, so that the data can be returned intact and can be used/displayed. The process of decompressing data includes the process of data normalization and the process of converting data to the original data form. The data decompression chart based on the RAKE algorithm explanation is as follows:

a. The process of retrieving data from compression on cloud platform.
b. Cloud Platform Decompresses the compressed data.
c. Data Normalitation
d. Data conversion, by changing the binary value to the text by using ASCII, or converting the binary value of the sensor to decimal.

2.4 System Implementation And Testing

2.4.1 Data Coding and Data Modeling Implementation

In this study using three methods in the process of coding data and the process of modeling data as follows:

1) ASCII + XOR
   Encoding: ASCII
   Modeling: XOR Operation + RAKE
2) (Binary Decimal + XOR)
   Encoding: Decimal to Binary Converter
   Modeling: XOR Operation + RAKE
3) (Binary Decimal + RAKE-residue)
   Encoding: Decimal to binary converter.
   Modeling: Residue + RAKE

2.4.2 Data Testing

The configuration flow before conducting the test, in general, will be shown in Figure 3

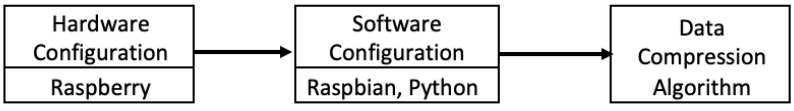

Figure 3 General Testing Design Flow

Figure 3 describes the flow of the test system design in general, starting from the hardware configuration, namely the Raspberry Pi as a client to compress data and send data to the cloud server. Generally, a testing system must have an input, process, and output. The testing phase begins with inputting sensor data into the system as material for analyzing performance. The process consists of the process of data compression and decompression and the process of sending data. The output is the analysis of data obtained from the compression and transmission of data from the IoT gateway to the cloud platform.

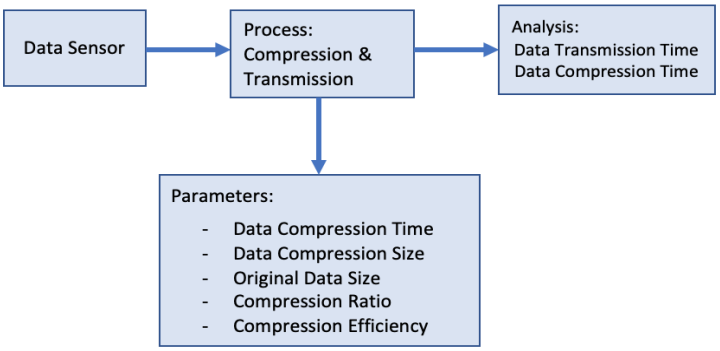

Figure 4 The Test Block Diagram

Figure 4 describes the test block diagram. First of all, sensor data input is done as entering the load into the system. After that the process of compression and transmission of sensor data is carried out, then monitoring of each parameter will be used as material for analysis of test results, namely data delivery time (without compression and compression), data size before and after compression, compression ratio and compression efficiency. After the testing system is prepared, the next process is to carry out testing and measurement of the system that has been implemented and then analyze the system, whether the system created is in accordance with what was planned. The conditions tested are as follows:

4) Performance test of data compression algorithms in test files.
- Compression Ratio (CR) is defined as the ratio between the number of bits before and after compression. Calculation of the compression ratio is shown in equation (1)[15].

$$CR = \frac{Number\ of\ bits\ BEFORE\ compression}{Number\ of\ bits\ AFTER\ compression} \quad (1)$$

- Compression Efficiency (CE%) defined as:

$$CE\ \% = 100 \times \left(1 - \frac{1}{CR}\right) \quad (2)$$

The compression efficiency is presented in percentage to describes a measure of the success of data compression.

5) Performance test algorithm for decompressing data in text files.
- Decompression ratio is defined as the ratio between the number of bits before and after decompression. Calculation of the decompression ratio is shown in equation (3).

$$DR = \frac{Original\ file\ size}{Number\ of\ bit\ AFTER\ decompression} \quad (3)$$

- Decompression Efficiency (DE%) is shown in equation (4).

$$DR\% = 100 \text{ x } \left(1 - \frac{1}{DR}\right) \quad (4)$$

The decompression efficiency is presented in percentage that describes a measure of the success of data decompression.

3. RESULT AND DISCUSSION

3.1 Compression Efficiency Parameter

The compression efficiency parameter is obtained from the calculation of the compression ratio by considering variable size data before compression and variable size data after compression according to a predetermined period. The results of testing data compression using several coding methods and modeling are attached to the attachment chapter. In this study, sensor data was first collected using five different periods for each testing experiment. This was done to obtain various data size values. The time period used is 10 minutes, 15 minutes, 20 minutes, 25 minutes, and 30 minutes with each period carried out as many as 70 attempts, then data compression is done using 4 different coding methods and data modeling methods.

3.1.1. ASCII + XOR Method Data Compression Testing

In the coding method by using text to binary conversion with ASCII, the size of the data length is large. This is because coding in this method is done per character (1 character is translated to 8 bits). So, the more data sensor is taken, the more characters produced. But the efficiency of compression using this coding method is above 70%. Figure 5 is the Graph of compression efficiency on the data length of testing data compression using ASCII coding and modeling using the XOR operation and the RAKE algorithm, while T1, T2, T3, T4 and T5 mean the number of experimen.

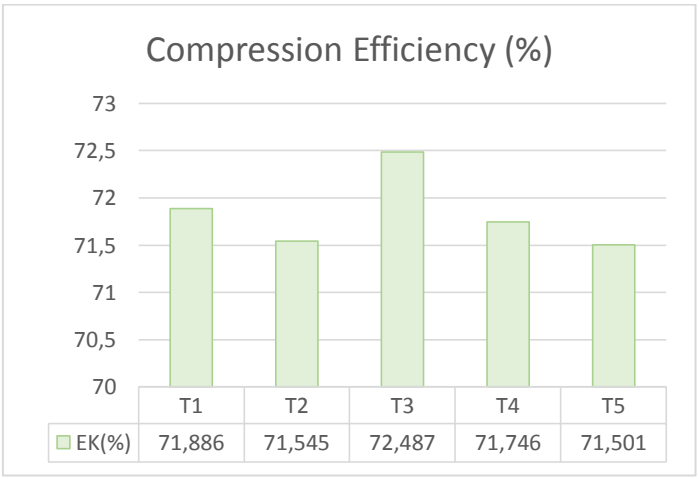

Figure 5 Compression efficiency graph of the data length

Based on the graph of compression efficiency on the data length, which is taken as many as 5 test times, for 10 minutes, 15 minutes, 20 minutes, 25 minutes, and 30 minutes, it can be seen that the compression efficiency value uses ASCII coding methods, and modeling methods using XOR gets an average yield of around 71.501% to 72.487%. This is because the

coding used is in the form of converting text characters to binary forms using ASCII so that changes in sensor values produce data changes that are not so large. So, when the XOR operations are carried out on the data, the possibility of '0' binaries appearing more, so compression efficiency is higher.

3.1.2. Binary-Decimal Method + XOR Data Testing

Fig 6 shows the graph of compression efficiency using coding by converting decimal to binary data and modeling using XOR operations and RAKE algorithms.In the data coding method using decimal to binary conversion, the data size is smaller than the data encoding using ASCII, this is because in binary coding to decimal for the sensor value in this study only requires 17 bits per sensor value, namely 1 bit as a negative representation, and the other 16 bits for decimal to binary representation. But the efficiency of compression using this coding method is low.

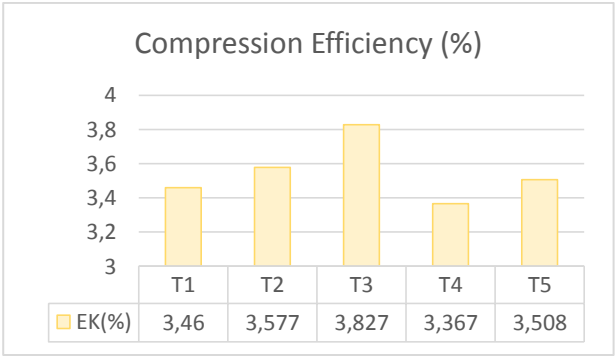

Figure 6 Compression efficiency graph of the data length

Figure 6 describes the compression efficiency value uses a binary-decimal data conversion coding method and the modeling method using XOR gets an average yield of around 3.367% to 3.827%. The value of compression efficiency using this method gets a lower compression efficiency value than the method with ASCII coding. This is because, in this method, data that is converted in binary form is only a binary conversion of a decimal value from the default sensor value with a sensor value at a specific time, ignoring the character. The binary value of the conversion of a decimal number has a significant change. For example, the binary humidity sensor value of 5422 is [1010100101110] XORed with 5423 which has a binary value [1010100101111], it will produce many bits 1. The more differences in data, the more likely the appearance of binary '1' in XOR operation. While this RAKE algorithm will be more effective if there are 0 binary numbers with large numbers, conversely, the less data is taken, the difference in data will be smaller.

3.1.3. Decimal-Binary + Residue Data Testing

Fig. 7 shows the Graph of compression efficiency using coding by converting decimal to binary data and modeling using residual values and RAKE algorithms. In the coding method using decimal to binary conversion, the data size is smaller than the data encoding using ASCII, this is because in binary coding to decimal for the sensor value in this study only requires 17 bits per sensor value, namely 1 bit as a negative representation, and the other 16 bits for decimal to binary representation.

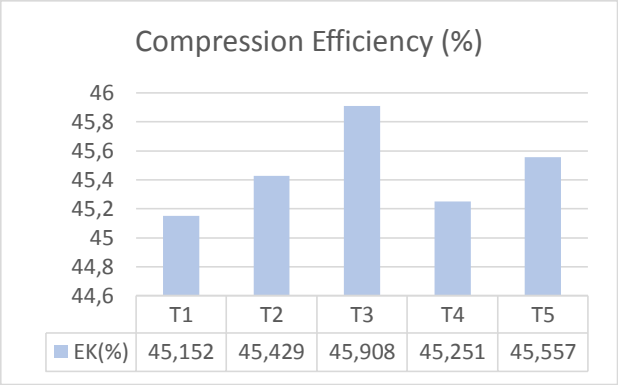

Figure 7 Compression efficiency graph of the data length

Based on the graph in Fig. 7 it can be seen that the compression efficiency value by using decimal to binary conversion coding method and using compression sensor residue values for modeling method gets an average yield of about 45.152% to 45.908%. This is because the process of directly reducing the value of the sensor is done directly to the default value, then the difference will be converted in binary form. The residual value does not change, so that when it is converted to binary form, the possibility of getting a binary value of '0' is greater.

3.1.4. ASCII + Residue Data Testing

In this section, data compression testing is carried out using the original method which will then be compared. In this method, the coding technique used is to use character conversion into binary using ASCII, as well as modeling techniques using compression residual values. In the coding method by using text to binary conversion with ASCII, the size of the data length is large. This is because coding in this method is done per character (1 character is translated to 8 bits).

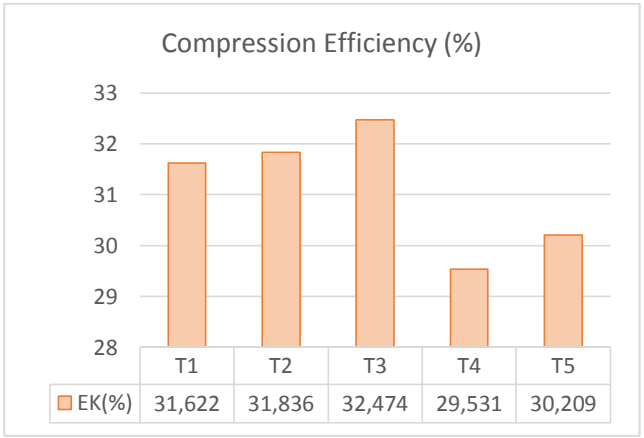

Figure 8 Compression Efficiency graph of the data length

3.2 Resume

The compression efficiency using the ASCII coding method and modeling with the XOR process gets the highest presentation value of 71.833%. After did the XOR operation to the data, the possibility of getting a bit with a value of '0' will be greater. The second-largest percentage of compression efficiency is to use coding using decimal to binary conversion and residual modeling, which is 45.459%. This is because changes to the sensor value will be

immediately reduced, then the decimal value of the residue will be converted to binary form. So the possibility of producing '0' binaries is also greater.

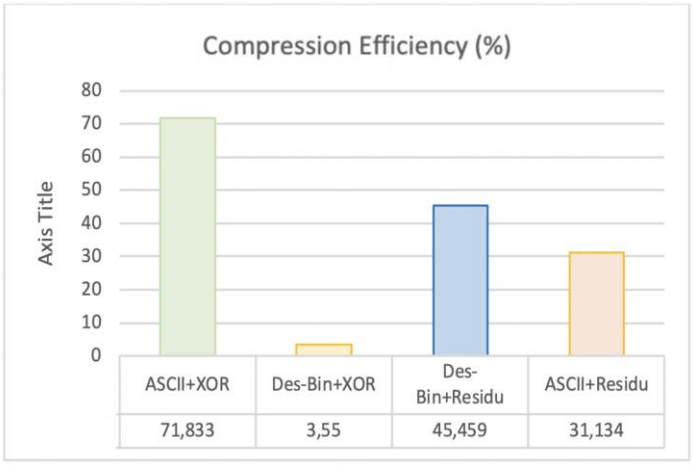

Figure 9 Resume

For the compression method using coding with decimal to binary conversion and modeling using the XOR operation, the average compression efficiency value is only 3.55%. This is because when there is a change in data, the conversion of decimal to binary from the sensor value obtained has the possibility of a large change in value. So that when the XOR operation is performed, it will generate a lot of '1' binary. The more binary '1' in the value of data to be compressed using the RAKE algorithm, the higher the size of the compressed data.

3.3 Decompression Efficiency Parameters

In the data decompression test, 4 test scenarios are carried out using five different periods. Table 3 is a summary of the results of testing of sensor data decompression using the RAKE algorithm. From the results of testing the decompression of the data, it is known that 100% of the data successfully returned as a whole.

Table 2 Decompression Efficiency Parameters

Number of Test	Decompression Efficiency (%)			
	ASCII +XOR	XOR-Desimal	Residue-Desimal	Residue-ASCII
P1	100	100	100	100
P2	100	100	100	100
P3	100	100	100	100
P4	100	100	100	100
P5	100	100	100	100

In the data decompression test, 70 trials were conducted for 5 periods with the aim that the results of the research conclusions can be generalized for all decompressed sensor data. From the results of the decompression test, the value of the data length before and after decompression remains the same and does not change. Then when the conversion of the compressed data is carried out, the data value is obtained before compression and the data value after the SAME / WHOLE compression for all compression methods that have been done. So that it can be concluded that 100% of the data has returned to its original state as a whole.

4. CONCLUSION

Based on the results of experimental data compression testing using ASCII+XOR, XOR-Decimal, Residue-Decimal, and Residue-ASCII methods, it can be seen that the combination of data coding techniques using ASCII and data modeling using XOR operations will produce the highest compression efficiency value, which is about 71%. *While the* Data compression using coding techniques to convert binary data to decimals combined with modeling techniques using residual values produces the smallest compression data.

The results of the test analysis using five different periods and data sizes indicate that sending data using coding techniques to convert decimal to binary data will produce small data, while coding techniques that use text to binary conversion with ASCII will produce data with large size. The compression efficiency will increase if the binary '1' in the bitmap set is less than 15% of the overall set of bitmaps, the more binary is '0' in the bitmap set, the more data is compressed.

REFERENCES

[1] Z. Lin & Zhang, L., 2016, Data Synchronization Algorithm for loT Gateway and Platform, 2016 2nd *IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, 2016, pp. 114-119

[2] D. Salomon & Motta, G., 2010, Handbook of data compression.

[3] K. Sayood, 2012, Lossless Image Compression, In, Introduction to Data Compression (Fourth Edition).

[4] G. Campobello, Segreto, A., Zanafi, S. & Serrano, S., 2017, RAKE : a Simple and Efficient Lossless Compression Algorithm for the Internet of Things, 7, 2650–2654.

[5] M. Vecchio, Giaffreda, R. & Marcelloni, F., 2014, Adaptive lossless entropy compressors for tiny iot devices, IEEE Transactions on Wireless Communications, 13, 2, 1088–1100.

[6] A. Pinho, J., 2002, An online preprocessing technique for improving the lossless compression of images with sparse histograms, IEEE Signal Processing Letters.

[7] Cloud & Grid Technology Research Group, 2017, G-Connect Project, https://cloud.wg.ugm.ac.id/newgamabox, accessed on September 2nd 2017.

[8] A. Desai, Nagegowda, K.S. & Ninikrishna, T., 2016, A framework for integrating IoT and SDN using proposed OF-enabled management device, Proceedings of IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2016, 1–4.

[9] M. Aazam, Khan, I., Alsaffar, A.A. & Huh, E.N., 2014, Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved, Proceedings of 2014 11th International Bhurban Conference on Applied Sciences and Technology, IBCAST 2014, 414–419.

[10] N. Malhotra & Chaudhary, A., 2014, Implementation of Database Synchronization Technique between Client and Server. *International Journal of Engineering Science and Innovative Technology* Volume 3, Issue 4, July 2014.

[11] A. Kumar, Nanjangud, C., Narendra & Umesh, B., 2016, Uploading And Replicating Internet Of Things (IoT) Data On Distributed Cloud storage", *2016 IEEE 9th International Conference On Cloud Computing, Vol. 00, No. , Pp. 670-677*, 2016, Doi:10.1109/CLOUD.2016.0094

[12] M. Sharma, 2010, Compression Using Huffman Coding, IJCSNS International Journal of Computer Science and Network Security.

[13] F. Hadiatna, Hindersah, H., Yolanda, D. & Triawan, M.A., 2017, Design and implementation of data logger using lossless data compression method for the Internet of Things, Proceedings of the 2016 6th International Conference on System Engineering and Technology, ICSET 2016, 105–108.