

Performance evaluation of arithmetic coding data compression for internet of things applications

Nor Asilah Khairi, Asral Bahari Jambek, Rizalafande Che Ismail

School of Microelectronic Engineering, Universiti Malaysia Perlis, Malaysia

Article Info

Article history:

Received Sep 3, 2018

Revised Nov 10, 2018

Accepted Nov 24, 2018

Keywords:

Arithmetic Coding

Data compression

Internet of Things (IoT)

Real-world datasets

Wireless Sensor Network

(WSN)

ABSTRACT

Wireless Sensor Network (WSN) is known for its autonomous sensors, where it has been found to be useful during the development of Internet of Things (IoT) devices. However, WSN is also known for its limited energy supply and memory space, as it carries small-sized batteries and memory space. Hence, a data compression approach has been introduced in this paper with the purpose of solving this particular issue. This paper focused on the performance of the Arithmetic Coding algorithm. Temperature (Temp), Sea-Level Pressure (Pressure), stride interval (Stride), and heart rate (BPM) were chosen as the dataset in this project. Based on the results, the compression ratio of Temp, Pressure, Stride, and BPM were 0.428, 0.255, 0.217, and 0.159 respectively. From this analysis, BPM produced the best compression ratio. Undeniably, the Arithmetic Coding algorithm is one of the best methods to compress real-world datasets. Hence, by using this approach, it can reduce the usage of energy and memory space.

Copyright © 2019 Institute of Advanced Engineering and Science.

All rights reserved.

Corresponding Author:

Nor Asilah Khairi,

School of Microelectronic Engineering,

Universiti Malaysia Perlis,

Pauh Putra Campus, 02600 Arau, Perlis, Malaysia.

Email: asilah.khairi91@gmail.com

1. INTRODUCTION

Internet of Things (IoT) can be defined as the next generation of the internet evolution, where devices and objects connected via the internet allow information to be delivered through a digital signal [1]. By the year 2020, the number of devices connected via the IoT may be as high as 75 billion [2]. However, billions of devices connected through the internet produce a large amount of data to be stored. Vural et al. [3] stated that billions of IoT devices connected via the internet that also generate low-rate traffic of measurement, monitoring, and automation data are now a major challenge for network providers and the internet as a whole. According to them, although each IoT has a low rate of traffic, the entire sum of traffic on the core network is expected to be large, which can obviously disrupt regular data traffic. A number of researchers had also discussed the possibility of implementing the compression algorithm into IoT devices, where it can minimise both storage requirements and input/output processes [4].

The Wireless Sensor Network (WSN) is commonly used in the military, industrial, medical, and agricultural sectors due to its wireless capabilities. It is also known for its autonomous sensors that have the ability to monitor physical or environmental conditions. Therefore, WSN is useful during the development of IoT devices. However, sensor node devices are known to have a limited energy supply, as they carry small-sized batteries. Although it has limited energy to function, it requires a low operating power in order to save energy and extend the lifetime of the sensor node [4]. Besides that, Jambek, and Khairi [5] mentioned that the transmission module has the largest power consumption compared to other components of the sensor node. It is due to the large amount of energy required to operate the wireless transmitter to transmit data. Therefore, a

data compression method is highly recommended to overcome these problems. In this paper, the Arithmetic Coding algorithm is proposed to prove that it can produce a higher compression ratio and better performance.

1.1. Arithmetic Coding

Arithmetic Coding is known as a method for replacing every input symbol with a codeword. A stream of input symbols are replaced with a single floating point number as the output [6]. According to Jacob, Somvanshi, and Tornekar [7], the Arithmetic algorithm is a very effective mechanism in eliminating redundancy in the encoding of data. Based on the explanation of the Arithmetic Coding in paper [6], the main objective of the Arithmetic Coding is to allocate an interval to each potential symbol. The decimal number is later assigned to the interval, where the range of the interval is 0.0 to 1.0. After the process of reading the input of the symbol is completed, the interval is subdivided into smaller intervals in proportion to the input symbol's probability. This subinterval is divided into parts according to the probability of the symbols from the input. This step is repeated for every single input symbol. Finally, any floating point number from the final interval uniquely determines the input data.

1.2. Previous Arithmetic Coding Research References

Arithmetic Coding is recognised as one of the best data compression approach. The references for the Arithmetic algorithm are limited and difficult to obtain because of patent issues. Hence, only 4 suitable journals were selected for references.

Shanmugasundaram, and Lourdasamy [6] proposed a traditional Arithmetic Coding method. This paper is focused on surveying the different basic lossless data compression algorithms. Based on this paper, the experimental results and comparison of the lossless compression algorithms used the Statistical- and Dictionary-based approaches. However, the Statistical approaches were chosen as a reference due to the comparison between the Arithmetic approaches to other data compressions, whereas the Dictionary approaches only used the Lempel Ziv. In this project, the authors only used 12 different types of text files as the datasets. From the observation, this Arithmetic algorithm has been proven to be one of the best performers among other methods by achieving the range of 0.57 to 0.76 in terms of compression and 5.15 in average of bits per character (BPC).

Paper [7] studied the comparative analysis in terms of the compression efficiency. The authors dealt with lossless compression approaches such as the Huffman, Arithmetic, LZ-78, and Golomb Coding. English text files, Log files, Sorted word lists, and geometrically distributed data text files were used in the project as datasets, as well as the MATLAB software during the implementation process. According to the authors, the Arithmetic Coding was suitable for moderate and high frequency operation. However, the algorithm also produced high computational complexity and slow compression speed compared to the Huffman approach.

Based on paper [8], Porwal, Chaudhary, Joshi, and Jain focused on the lossless data compression methodologies and comparison of their performances. The authors used a traditional Arithmetic algorithm in their project, which was similar with projects [6] and [7]. However, they only used single-data compression during the comparative performance, such as the Huffman method. Texts, videos, audios, and images were used as datasets. According to the researchers, the Arithmetic algorithm produced a high compression ratio and used less memory space. However, the weakness of this algorithm is similar to previous project [7], which had slow compression and decompression processes.

Project [9] was quite different from the previous projects in term of its usage of a modified Arithmetic approach. The researcher focused on proposing a new algorithm called the j-bit encoding (JBE), where it has the ability to minimise the size of data. This modified algorithm is a combination of the Arithmetic (ARI) with the Run-Length Encoding (RLE), Burrows-Wheeler Transform (BWT), Move-To-Front (MTF), and JBE. During the comparative process, the researcher used four approaches: a combination of the ARI with RLE, the ARI with BWT, and MTF, the ARI with BWT, and RLE and the ARI with RLE, BWT, MTF, and RLE. Images, texts, binaries, and audios were used as datasets. Based on the observation, the proposed algorithm produced a high compression ratio in 5 datasets.

1.3. Comparison of the Performance of Previous Arithmetic Coding Research References

Several data compression projects in the past have used the Arithmetic approach, though not as frequently as other approaches such as the Huffman and Lempel-Ziv-Welch (LZW). However, some of these projects used a modified Arithmetic approach in order to achieve a better performance. Table 1 shows the summarisation of the comparison between the Arithmetic algorithms:

Table 1. Comparison of the Arithmetic Coding Algorithm in Previous Projects

Description	[6]	[7]	[8]	[9]
Year	2011	2012	2013	2012
Focus	Survey of different basic lossless data compression algorithms	Comparative analysis in terms of their compression efficiency and speed	Focus on lossless data compression methodologies and compares their performance	Proposed new algorithm that can minimise the size of data
Architecture/ Algorithm	Algorithm	Algorithm	Algorithm	Algorithm
Type of Data Compression	Arithmetic Coding	Arithmetic Coding	Arithmetic Coding	RLE + BWT + MTF + JBE + ARI
Comparison of Other Data Compression	Shannon-Fano Coding, Huffman, Adaptive Huffman, Run-Length Encoding	Huffman, LZ-8, Golomb Coding	Huffman	RLE + ARI, BWT + MTF + ARI, BWT + RLE + ARI, RLE + BWT + MTF + RLE + ARI
Dataset	Text data	English text files, log files, sorted word lists, and geometrically distributed data text file	Text, video, audio, and image	Image, text, binary, and audio
Advantage	Best performance in the compression ratio and low bits per character	Can perform well for texts that contain highly skewed probability symbols	High compression ratio, less memory space	High compression ratio in 5 types of files
Disadvantage	Not stated	High computational complexity, less speed	Slow compression and decompression	Not stated

Based on the data collected in table 1, project [6] was the most suitable project to be used as a reference. One of the similarities of project [6] with the other projects is in the implementation of the Arithmetic algorithm method in the project. Project [6] used the traditional Arithmetic approach, which was also similar with projects [7] and [8]. Each of these projects used the text file as the dataset. Authors in [6] used multiple types of text files in order to produce diverse results. Besides that, the advantage of project [6] is almost similar to other projects, in which it had better compression ratio.

Based on the observation, paper [6] was selected as a reference not only for its similarities, but also for its own uniqueness. The measurement method in paper [6] is different to other papers because it used the compression ratio and BPC measurement to measure the performance of data compression, whereas most of the projects used only the compression ratio to measure the performance of compression. Therefore, it produced a wide range of results. However, project [6] performed the experiment quite differently from the other projects. This paper divided the data compression into two categories: statistical- and dictionary-based compression techniques. The authors did not lump them together in one group due to their abilities. In spite of this, the results of this paper had more relevance, as it did not mix the results of statistical data compression with the dictionary-based data compression.

Paper [6] was chosen as the best reference paper because it is nearly similar with the proposed project. Although it was chosen as a reference paper, the Arithmetic Coding in [6] required a minor modification to meet the requirements. One of the examples was a modified version of the Arithmetic algorithm. The Arithmetic algorithm required some modification to support real-world datasets. Most of these previous projects did not use a numbering text file dataset, hence requiring certain alterations to allow the compiler to process the numbering data. This proposed project is focused on collecting real-world datasets. Therefore, the environmental and biomedical datasets were selected for its unique patterns.

2. RESEARCH METHOD

The process of the Arithmetic Coding algorithm is explained in this section using the flowchart approach. This algorithm can only support numbering due to the use of the double data type. The structure of the algorithm was built in the C programming language. In this project, it is considered as 1 byte for a single character, including a new line of characters. The output of the compression was printed in the text file format (.txt) by assuming the format of file that will be implemented in the WSN during the transferring of the data. Moreover, the algorithm of Arithmetic was modified to suit with the requirement.

The flowchart of the Arithmetic algorithm was divided into two parts: compression and decompression. Figure 1 illustrates the process of data compression for the Arithmetic Coding approach. In compression, the raw data from the .txt format file was inserted into array1. In the next step, array2 copied the data from array1 to preserve the originality of the array1 sequence. The lists of data inside array1 were then divided into 10 data and stored column to column. The data inside the columns were arranged in an ascending

order, since it was easier for the compiler to count the repetition of the data. The frequency, probability, and range of the repetitions of the data were later counted and stored into array3.

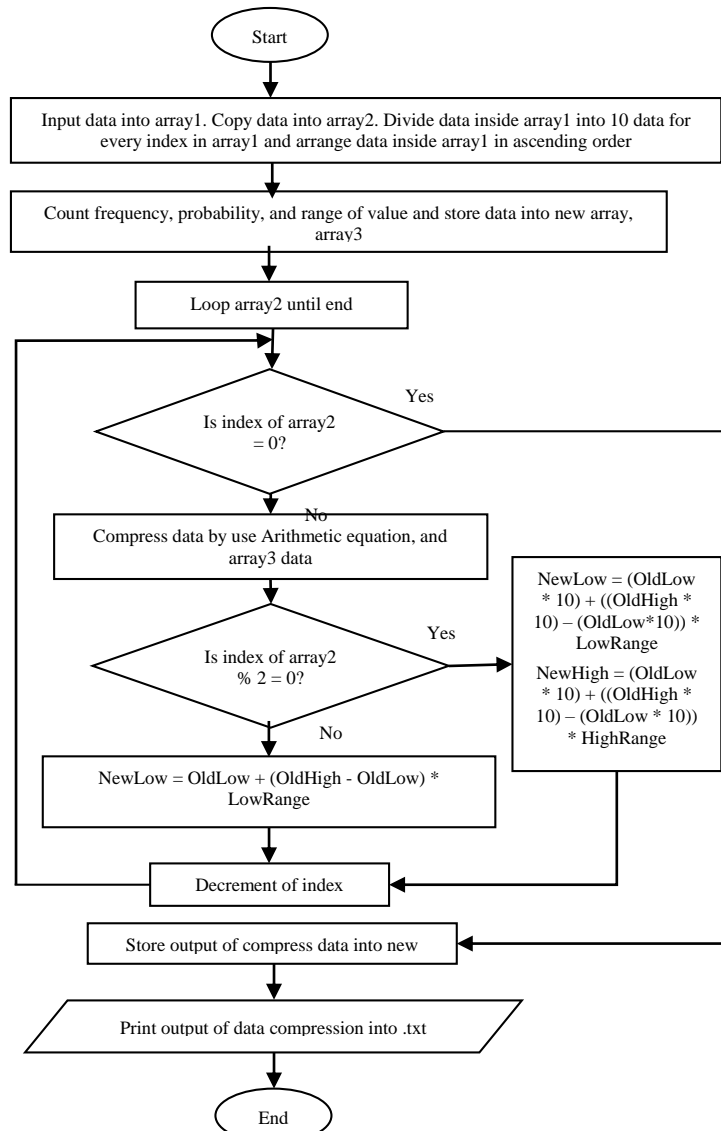


Figure 1. Flowchart of Arithmetic Coding compression algorithm

The probability and range of the repetitions were calculated based on the equations (1) and (2). Equation (1) was divided by 10 due to a single column requiring 10 data. On the other hand, equation (2) consists of two equations: LowRange and HighRange. Equation (2) was adapted from Salomon's book [10].

$$\text{Probability} = \text{Frequency} \div 10 \quad (1)$$

$$\text{LowRange} = \text{PreviousHighRange}; \text{HighRange} = \text{LowRange} + \text{Frequency} \quad (2)$$

For the next step, the compiler looped array2 until the index became zero value. First, the compiler checked the existence of value 0 in the index of array2. The compression process will occur if the value did not equal to 0. The compiler used the Arithmetic equations and the data inside array3 during the compression process. Before the compiler used the equation, the similarities of the values in array2 and array3 were required. The compiler could then execute the process by obtaining the data from array3 and determine the current index value of array2 by modulo with 2. If the value was equal to 0, it will use equation (3). Otherwise, it will use the modified equation (4). Equations (3) and (4) were adapted from book [10].

$$\begin{aligned} \text{NewLow} &= \text{OldLow} + (\text{OldHigh} - \text{OldLow}) * \text{LowRange} \\ \text{NewHigh} &= \text{OldLow} + (\text{OldHigh} - \text{OldLow}) * \text{HighRange} \end{aligned} \tag{3}$$

$$\begin{aligned} \text{NewLow} &= (\text{OldLow} * 10) + ((\text{OldHigh} * 10) - (\text{OldLow} * 10)) * \text{LowRange} \\ \text{NewHigh} &= (\text{OldLow} * 10) + ((\text{OldHigh} * 10) - (\text{OldLow} * 10)) * \text{HighRange} \end{aligned} \tag{4}$$

After the calculation process is completed, the index in array2 decremented and looped until it became 0. Then it stored the output of the compressed data into a new array. Finally, it printed the output of compressed data into the .txt format.

The process of decompressing was different and quite difficult due to its requirement to process a row of separate data before combining it into a single line of data. Figure 2 shows the process of data decompression using the Arithmetic Coding approach.

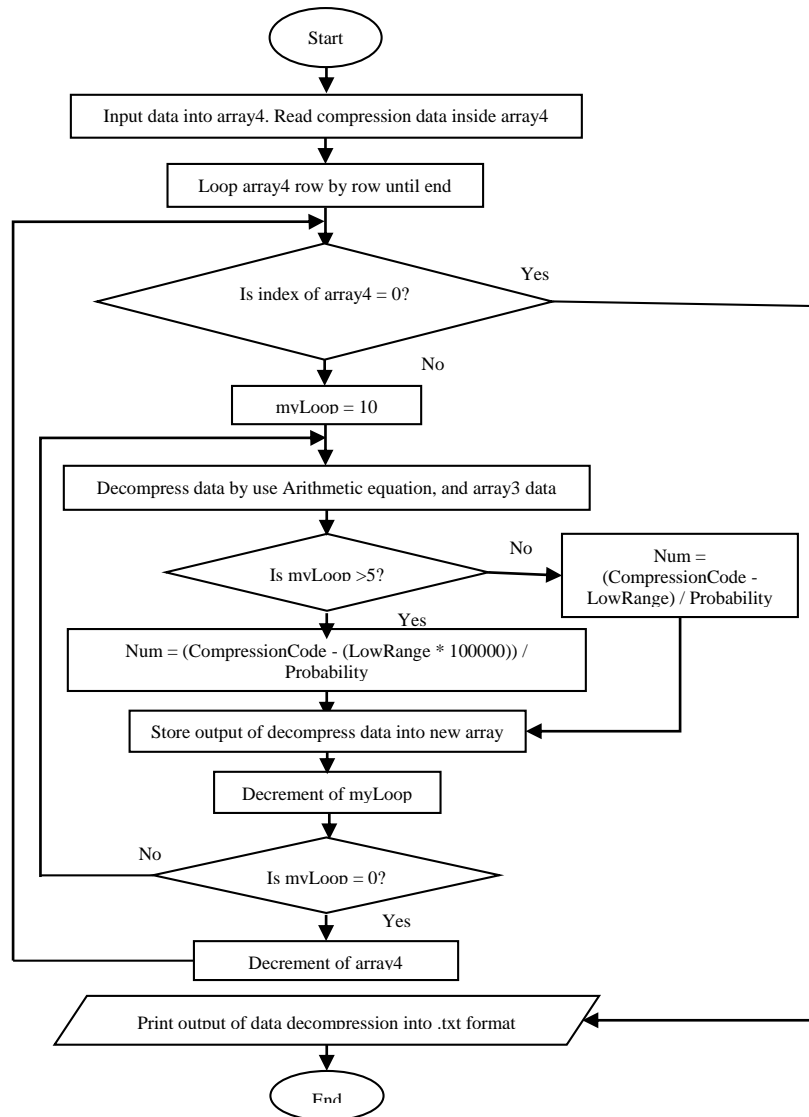


Figure 2. Flowchart of the Arithmetic Coding decompression algorithm

The first step of decompression was to input the compressed data into array4. Then the compiler read the data inside array4. For the next step, the compiler looped array4 row by row until the end. First, the compiler identified the existence of value 0 in index of array4. If it does not equal to 0, the compiler then set the integer of myLoop as 10, as the system was set by storing 10 data for each column. The decompression process occurred by using decompressed Arithmetic equations and array3 data. The compiler was required to identify

the values of myLoop. If the index of myLoop is bigger than 5, it thus requires the use of the modified equation (5). Otherwise, it would proceed to use equation (6). Equations (5)-(6) were adapted from book [10].

$$\text{Num} = (\text{CompressionCode} - (\text{LowRange} * 100000)) / \text{Probability} \quad (5)$$

$$\text{Num} = (\text{CompressionCode} - \text{LowRange}) / \text{Probability} \quad (6)$$

Before using this equation, the compiler was required to identify the similarities of the values in array4 with array3. Hence, the compiler would be able to execute the process by obtaining the data from array3. After the calculation was completed, it stored the output of the decompressed data into a new array. Then, the index in myLoop was decremented and looped until it became 0. If the index of myLoop was equal to 0, it would then proceed to the next step by decrementing array4. Then, it identified the index of array4. If it is equal to 0, it will then proceed by printing the output of the decompressed data into .txt format.

To make the results more realistic, real-world datasets from various application domains were used in this paper. The datasets were temperature measurement in Alor Setar (Temp) [11], sea-level pressure measurement in Alor Setar (Pressure) [11], stride interval of healthy human in fast-walking (Stride) [12], and heart rate of an elite athlete (BPM) [12]. The reason Temp, Pressure, Stride, and BPM data was used in this experiment is because they had a variety of patterns.

3. RESULTS AND ANALYSIS

This section discusses the compression results acquired from the four datasets. To estimating the performance of the compression algorithm, compression ratio calculation was used in this research. Figures 3 and figure 4 illustrate the performance of the four datasets in a bar graph form.

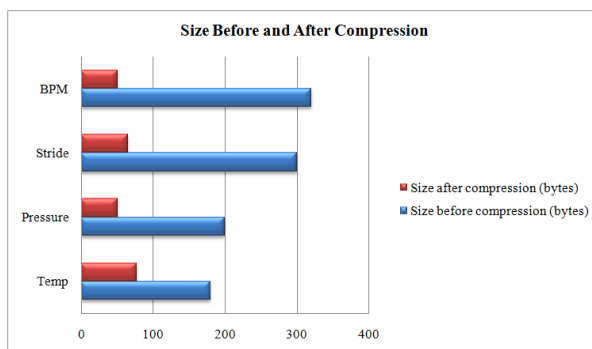


Figure 3. Size before and after compress for four datasets

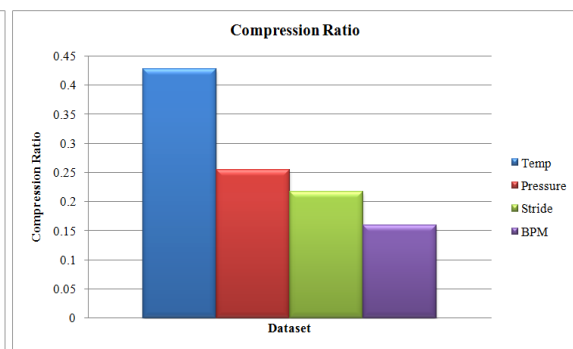


Figure 4. Compression ratio for four datasets

Based on the figures, all datasets decreased after being compressed. Based on these datasets, BPM produced the lowest compression ratio of about 0.159. This is followed by Stride at 0.217 and then Pressure at about 0.255. Temp dataset produced the highest compression ratio, which was about 0.428. Therefore, BPM achieved the best compression ratio among all the datasets. The result in figures 3 shows that the Arithmetic is one of the best approaches to compress data. Hence, the Arithmetic approach is a suitable for application on real-world datasets. According to the observation, the results of the compression ratio were affected by the length of numbers. Therefore, the algorithm produces better result in compression ratio when the lengths of numbers are longer. Based on the results, it can be deduced that Arithmetic is one of the best methods to compress real-world datasets by achieving better compression ratio. As a result, it has the ability to reduce the usage of memory and also consume less energy.

4. CONCLUSION

WSN is known for its autonomous sensors and usefulness in the IoT world. However, WSN is also known for its limited energy supply and memory space due to the small-sized battery and memory. Therefore, the Arithmetic algorithm was introduced in this project in hopes of solving just this problem. Different types of datasets were used in this project, such as Temp, Pressure, Stride, and BPM. Based on the results of the

experiment, the compression ratio of Temp, Pressure, Stride, and BPM were at 0.428, 0.255, 0.217, and 0.159 respectively. It showed that BPM produced better compression ratio than the other three datasets. It also showed that the Arithmetic algorithm is one of the best methods to compress real-world datasets. Therefore, by using this algorithm, it should be able to minimise the consumption of energy and memory space.

REFERENCES

[1] K. D. Chang, J. L. Chen, H. C. Chao and C. W. Liu, "The Potential Cloud Application Model for Internet of Things - Case Study of Shopping Malls," in *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, Kitakyushu , 2014, pp. 954 – 957.

[2] F. J. Riggins and S.F. Wamba, "Research Directions on the Adoption, Usage, and Impact of the Internet of Things through the Use of Big Data Analytics," in *2015 48th Hawaii International Conference on System Sciences (HICSS)*, Hawaii, 2015, pp. 1531 – 1540.

[3] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong and R. Tafazolli, "In-network caching of Internet-of-Things data," in *2014 IEEE International Conference on Communications (ICC)*, Sydney, 2014, pp. 3185 – 3190.

[4] S. Sheikh and H. Dakhore, "Data Compression Techniques for Wireless Sensor Network," (*IJCSIT*) *International Journal of Computer Science and Information Technologies*, vol. 6, no. 1, pp. 818-821, Nov 2015.

[5] A. Bahari Jambek and N. A. Khairi, "Performance Comparison of Huffman And Lempel-Ziv Welch Data Compression for Wireless Sensor Node Application," *American Journal of Applied Sciences*, vol. 11, no. 1, pp. 119-126, Jan 2014.

[6] S. Shanmugasundaram and R. Lourdasamy, "A Comparative Study Of Text Compression Algorithms," *International Journal of Wisdom Based Computing*, vol. 1, no. 3, pp. 68-76, Dec 2011.

[7] N. Jacob, P. Somvanshi and R. Tornekar, "Comparative Analysis of Lossless Text Compression Techniques," *International Journal of Computer Applications*, vol. 56, no. 3, pp. 17-21, Oct 2012.

[8] S. Porwal, Y. Chaudhary, J. Joshi and M. Jain, "Data Compression Methodologies for Lossless Data and Comparison between Algorithms," *International Journal of Engineering Science and Innovative Technology (IJESIT)*, vol. 2, no. 2, pp. 142-147, Mar 2013.




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

[10] D. Salomon, *Data Compression The Complete Reference*, 3rd ed. New York, NY: Springer-Verlag New York, Inc, 2004, pp. 110-113.

[11] Weather Underground, "<https://www.wunderground.com>".

[12] PhysioNet, "<https://physionet.org>".

BIOGRAPHIES OF AUTHORS

	<p>Nor Asilah Khairi is currently pursuing her Master of Science (M.Sc) in School of Microelectronic Engineering, Universiti Malaysia Perlis (UniMAP) since January 2016. She completed both her Diploma and Bachelor of Computer Science (Hons) at Universiti Teknologi MARA (UiTM) in 2012 and 2015. Her research study is Design of Efficient Data Compression Algorithm and Its Implementation in Portable Electronic Device for Internet of Things Applications.</p>
	<p>Asral Bahari Jambek is an Associate Professor at the School of Microelectronics Engineering, Universiti Malaysia Perlis (UniMAP), and was a Programme Chairperson for the Electronics Engineering Degree Programme, UniMAP. He has more than 15 years experience in integrated circuit and system design in both the industry and academic sectors, and has been involved at various levels of VLSI design such as transistor modelling, digital circuit design, analogue circuit design, logic synthesis and physical place and route, architecture design and algorithm development.</p>
	<p>Rizalafande Che Ismail is an Associate Professor of Electronic Engineering at the School of Microelectronic Engineering, Universiti Malaysia Perlis (UniMAP). He serves as Dean of School of Microelectronic Engineering UniMAP. He obtained his Ph.D from Newcastle University in Microelectronics System Design. His research activities include high speed computer arithmetic, development of high performance logarithmic based processor and FPGA design platform, high accuracy video graphic computation and biomedical applications.</p>