Performance comparison of Huffman and Lempel-Ziv welch data compression for wireless sensor node application

Article *in* American Journal of Applied Sciences · January 2014

DOI: 10.3844/ajassp.2014.119.126

CITATIONS
19

READS
2,581

2 authors, including:

Asral Bahari Jambek
Universiti Malaysia Perlis
103 PUBLICATIONS **477** CITATIONS

SEE PROFILE

American Journal of Applied Sciences 11 (1): 119-126, 2014
ISSN: 1546-9239

doi:10.3844/ajassp.2014.119.126 Published Online 11 (1) 2014 (http://www.thescipub.com/ajas.toc)

PERFORMANCE COMPARISON OF HUFFMAN AND LEMPEL-ZIV WELCH DATA COMPRESSION FOR WIRELESS SENSOR NODE APPLICATION

Asral Bahari Jambek and Nor Alina Khairi

School of Microelectronic Engineering, Universiti Malaysia Perlis, Pauh Putra Campus, Perlis, Malaysia

Received 2013-09-23; Revised 2013-10-28; Accepted 2013-12-10

ABSTRACT

Wireless Sensor Networks (WSNs) are becoming important in today's technology in helping monitoring our surrounding environment. However, wireless sensor nodes are powered by limited energy supply. To extend the lifetime of the device, energy consumption must be reduced. Data transmission is known to consume the largest amount of energy in a sensor node. Thus, one method to reduce the energy used is by compressing the data before transmitting it. This study analyses the performance of the Huffman and Lempel-Ziv Welch (LZW) algorithms when compressing data that are commonly used in WSN. From the experimental results, the Huffman algorithm gives a better performance when compared to the LZW algorithm for this type of data. The Huffman algorithm is able to reduce the data size by 43% on average, which is four times faster than the LZW algorithm.

Keywords: Component, Formatting, Style, Styling, Insert

1. INTRODUCTION

The increasing usage of wireless communication devices has resulted in the rapid development of Wireless Sensor Networks (WSNs). The devices monitor and collect data before transmitting it to the base station. Due to its wireless capability, the system can be implemented in many applications, including military, industry, medical and agricultural.

One of the problems in implementing WSN is the energy consumed by the sensor node. Due to its small size, the sensor node has a limited energy supply and storage capacity. Thus, researchers need to find ways to reduce its power consumption so that the device's lifetime can be increased without the frequent need for the replacement of batteries.

Among the many components of the sensor node, the transmission module has the largest power consumption (Al-laham and El-Emary, 2007). This is because a huge amount of energy is needed to power up the wireless transmitter in order to transmit the data. Thus, one way to reduce the energy consumption is by compressing the data before transmission. By doing this, the amount of data needed to be transmitted to other nodes reduces, thus, reducing the power consumption due to the transmission. The higher that the data compression ratio is, the more power can be saved when transmitting the data.

The existing literature discusses the performance of the data compressed using different data types, such as text, images and others. In this work, we compare the performance of the data compression that is commonly used for WSNs.

In this study, two different data compression methods were analysed, namely the Huffman and Lempel-Ziv Welch (LZW) algorithms. The aim of the work is to identify the method that could results in the highest compression ratio and performance.

This study is organized as follows. Section II discusses the existing work on data compression techniques. In section III, the Huffman and LZW data compression algorithms are discussed. Section IV highlights the results obtain in this study. Lastly, section V concludes the paper.

Corresponding Author: Asral Bahari Jambek, School of Microelectronic Engineering, Universiti Malaysia Perlis, Pauh Putra Campus, Perlis, Malaysia

1.1. Literature Review

Shahbahrami *et al*. (2011), a survey of data compression techniques was discussed, including the Huffman and LZW data algorithms. The types of data evaluated in this study were .DOC, .TXT, .BMP, .TIF, .GIF and .JPG. From the paper it can be seen that for a text file (.DOC or .TXT), the compression ratio for both algorithms is almost the same. For an uncompressed image file (.BMP or .TIF), the LZW algorithm performs better than the Huffman algorithm. As for the .GIF and JPG image files, when compressed using the LZW algorithm, the compressed files were larger compared to before the compression was applied. This shows that the LZW data compression is not suitable for this image format since the original file is already in compressed form.

Paper (Strydis and Gaydadjiev, 2008) discusses the comparison between the Huffman and arithmetic data compression algorithms using image files. From the experimental results, as the size of the image file increases, the compression ratio also increases. The time taken for the Huffman algorithm to execute is shorter compared to the arithmetic algorithm. To compress a 128×128 image size, Huffman takes 0.14 sec while arithmetic coding requires 0.45 sec to complete the task.

Paper (Shanmugasundaram and Lourdusamy, 2011) analysed the most suitable type of data compression for biomedical applications. The paper analysed the compression ratio, execution time, energy consumption and program-code size. In this application, the implanted device typically consists of data-memory sizes ranging from 1KB to 10KB. Both sizes were investigated in this work. Based on the results, the Huffman algorithm gives a better compression ratio for 1kB data as compared to LZW, whereas both algorithms perform equally well for 10 kB. LZW has the advantages of a faster execution time and lower energy consumption for this application.

A survey was done in (Kodituwakku and Amarasinghe, 2010) to compare the performance between different types of data compression. Different file types and sizes were used in this research, consisting of various benchmark text files. From the paper, the LZW algorithm performs slightly better than the Huffman algorithm, with each of them consuming 4.9 and 5.7 bits per character, respectively.

Paper (Marcelloni and Vecchio, 2008) focuses on the compression of multiple sizes of text data. For the LZW, the compression ratio ranges between 30 and 60% and this ratio decreases as the file size increases. This is because larger text data will create longer LZW code. For Huffman coding, the compression ratio is obtained between 58 and 67%. The compression time for the LZW algorithm is larger than the Huffman algorithm because the scanning window or the LZW algorithm takes more time in order to fill up the dictionary inside the LZW. Although the compression time is longer, it takes a shorter time to decompress using the LZW algorithm than the Huffman algorithm. This is because the decoding process only needs to decode the data by matching the LZW code with the code inside the library.

While the existing method focuses more on text and image data, this study will focus especially on data that are commonly used in WSN, such as temperature, humidity and ECG. In the next section, the data compression that is used in this study will be elaborated.

2. MATERIALS AND METHODS

This section describes the work done for this study. First, it will discuss the Huffman algorithm, followed by a discussion of the LZW algorithm. In addition, the compression performance for a combined Huffman-LZW algorithm will also be discussed.

The Huffman encoder maps an alphabet or symbol to a binary code. The binary code is composed of sequences of binary bits of different sizes. The repeatedly appearing alphabet will be represented by smaller sized binary bits compared with the infrequently appearing one (Gonzalez and Woods, 2008). **Figure 1 and 2** shows the flow chart for the Huffman encoder and decoder, respectively.

Unlike Huffman coding, the LZW coding sets permanent-length code words to variable length series of source symbols (Kelly, 2007). LZW builds a 'dictionary' that contains words or parts of words of a datum. When the data needs to be decompressed, it needs to refer to the dictionary, which in turn represents the LZW code for that word (Shahbahrami *et al*., 2011). **Figure 3 and 4** shows the LZW encoder and decoder flow charts, respectively.

For double compression, the combination of Huffman followed by LZW (HLZ) and LZW followed by Huffman (LZH) were used. Double compression is investigated in this work to measure that performance when compressing different types of data.

In this work, there are four types of input data that are used, namely temperature, humidity, ECG and text. The temperature data were taken from the Average Daily Temperature Archive, University of Dayton (Dan, 2008).

The file contains daily temperatures from 1st January 1995 until 31 December 2012. **Figure 5** shows some samples of the temperature data in Fahrenheit (F).

For the humidity data, this was taken from the National Environmental Satellite, Data and Information Service (NIH, 2012). It is a monthly humidity record throughout the year 2002.

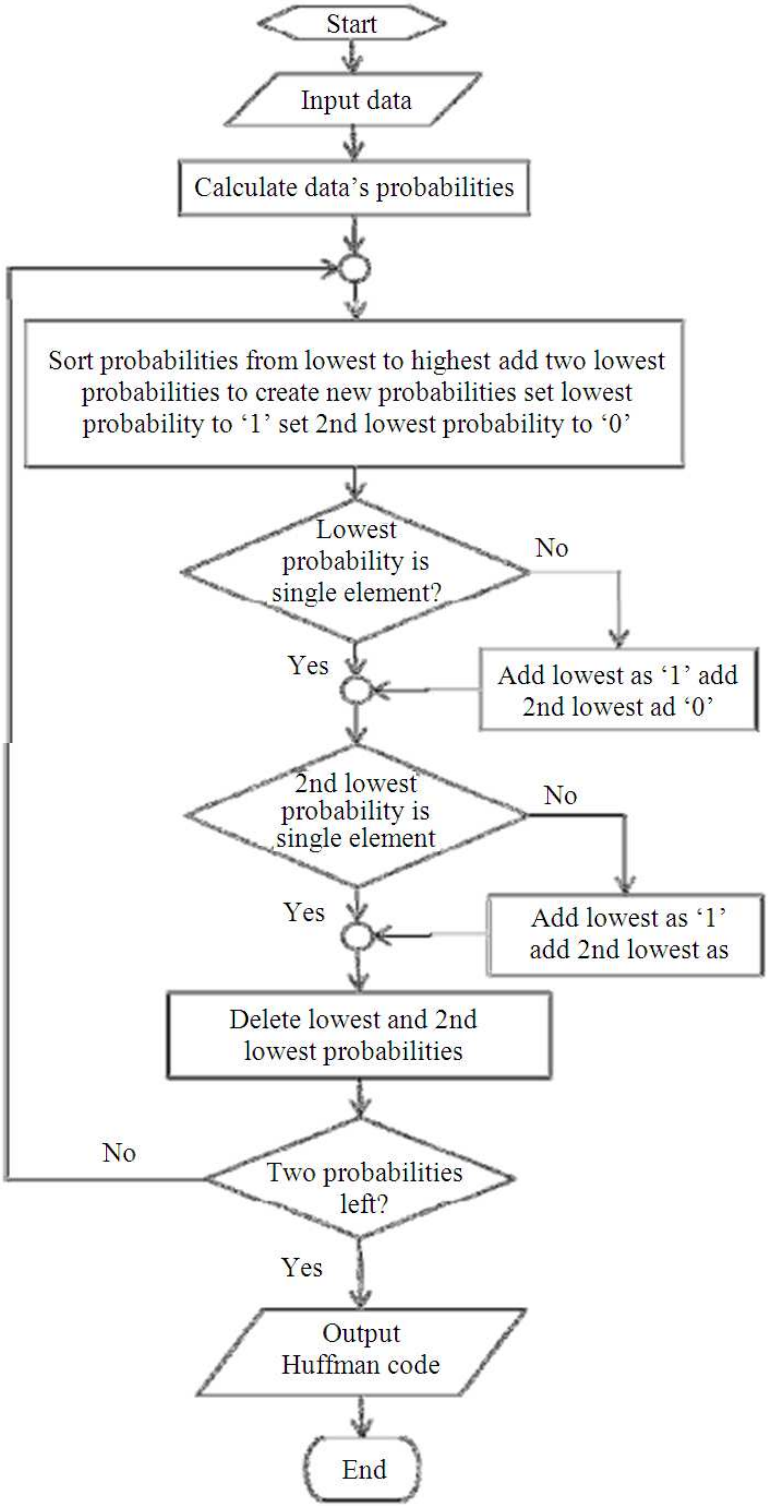


Fig. 1. Huffman encoder flow chart

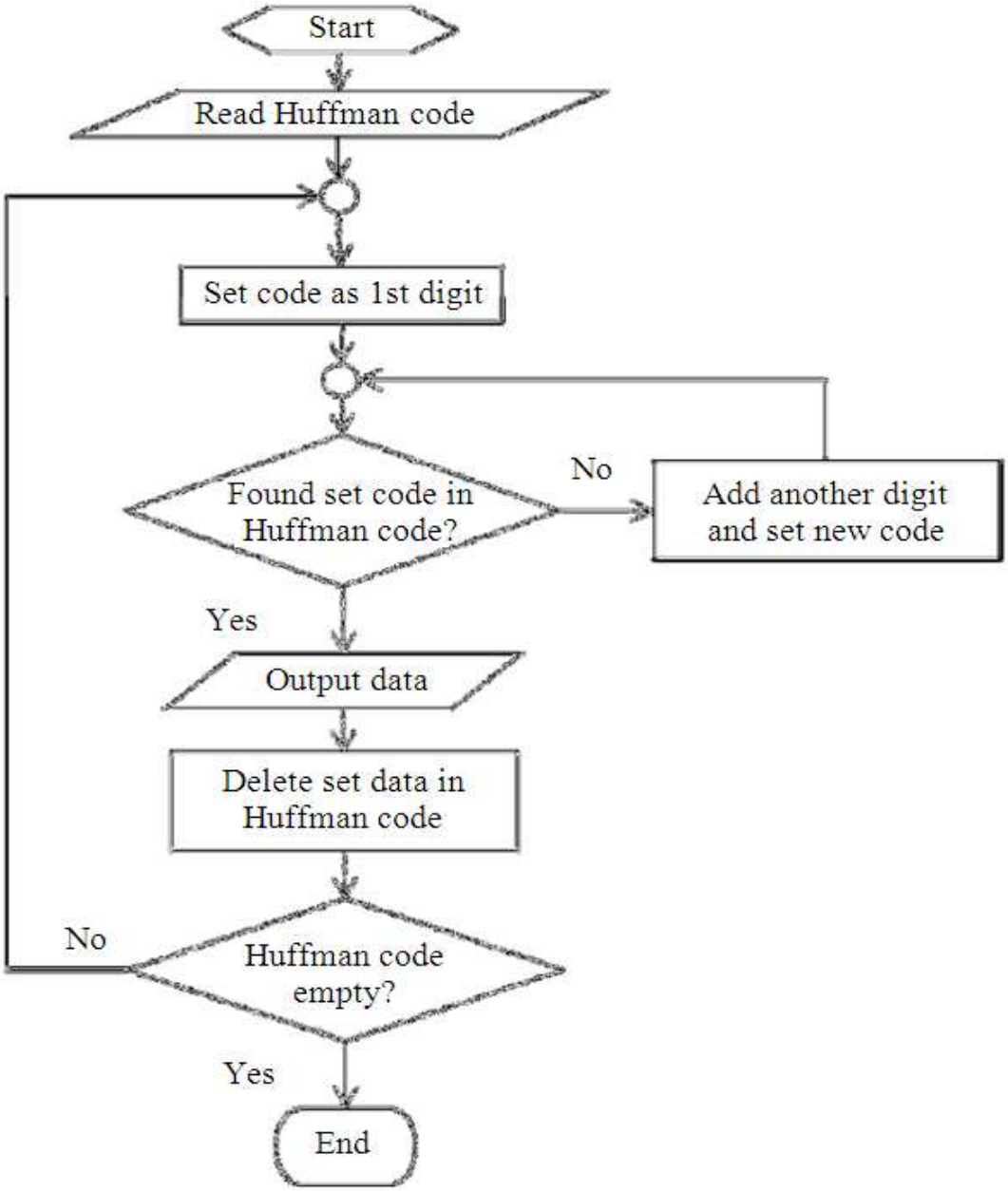


Fig. 2. Huffman decoder flow chart

Samples of the humidity are shown in **Fig. 6**. The numbers represent a percentage measure of the amount of moisture in the air compared to the maximum amount of moisture that the air can hoard at the same temperature and pressure.

PhysioBank is a website where the ECG data in this work were obtained (SMLLC, 2013). The data chosen concerned an apnoea patient, a disorder manifest by pauses in breathing or shallow breaths during sleep. The data in **Fig. 7** is relatively unique and has its own pattern. **Figure 7** shows the waveform for the ECG data used in this work, where the x axis is the time in 10^{-2} sec and the y axis is the amplitude in mV. Lastly, the text file sample was taken from the Mother Goose Club's website.

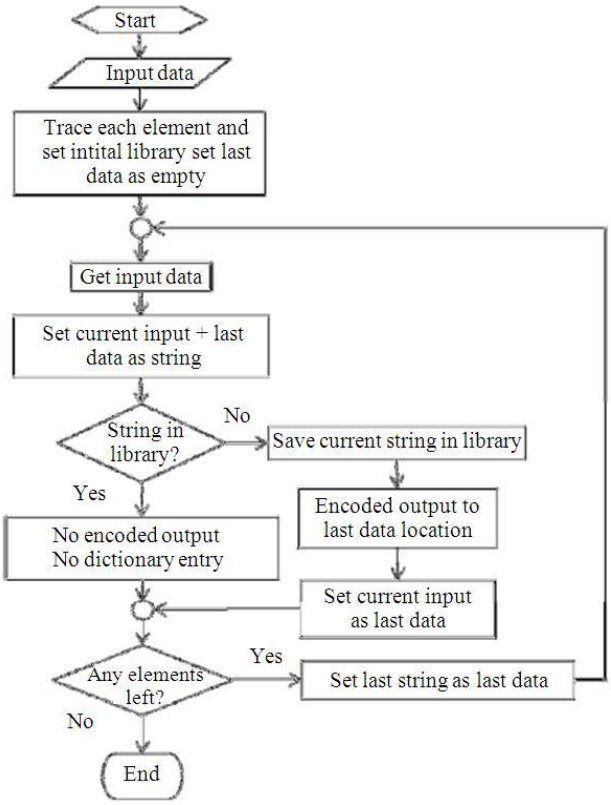


Fig. 3. LZW encoder flow chart

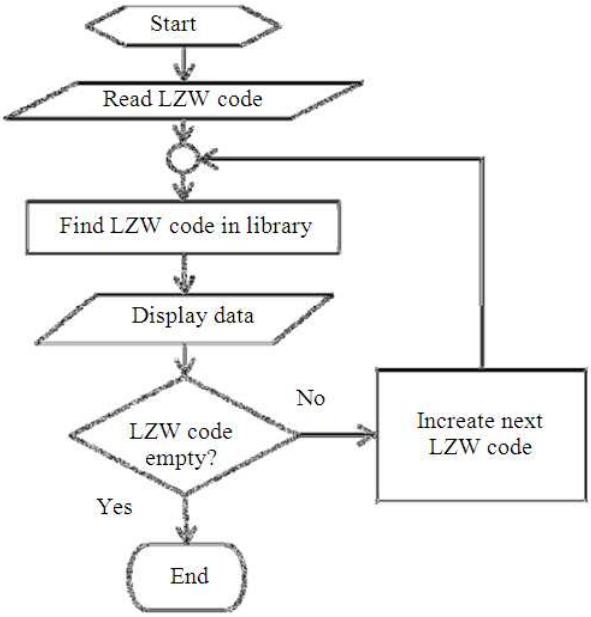


Fig. 4. LZW decoder flow chart

```
82.0, 82.6, 80.9, 81.1, 80.8, 80.2, 80.2, 79.5, 79.3,
78.7,...
```

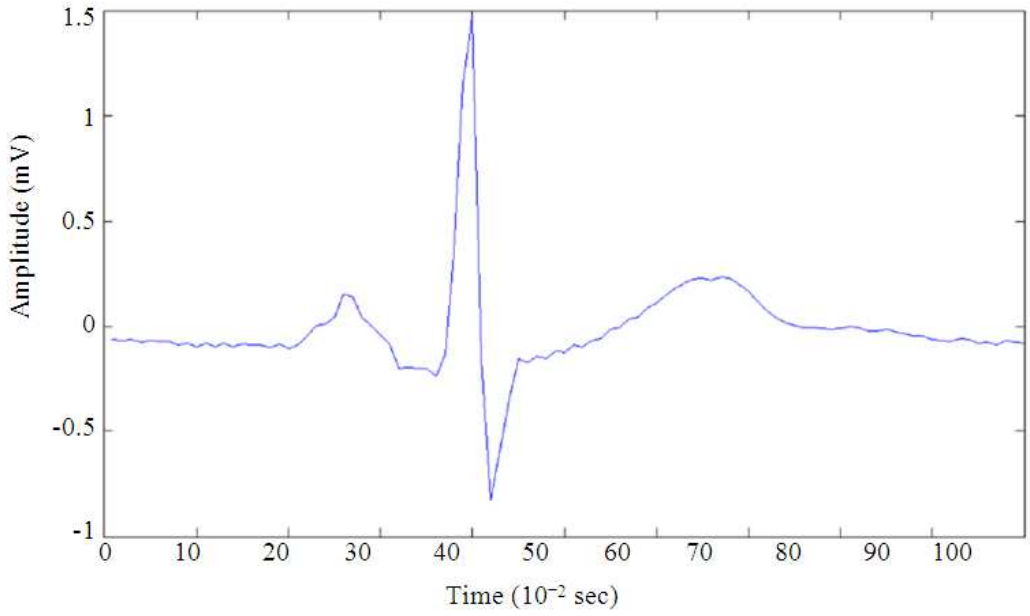
Fig. 5. Temperature data

```
80, 82, 83, 82, 75, 81, 69, 70, 78, 69,...
```

Fig. 6. Humidity data

```
-0.060, -0.065, -0.060, -0.075, -0.065, -0.070, -
0.070, -0.090, -0.080, -0.095,...
```

(a)



(b)

Fig. 7. (a) ECG data (b) Waveform for the ECG data

3. RESULTS AND DISCUSSION

This section discusses the compression results using data that are typical for WSNs, such as temperature, humidity, ECG and words. For each type of data, three different sizes are evaluated.

Table 1 shows the compression results for various data with different sizes compressed using Huffman,

LZW, HLZ and LZH algorithms. From **Table 1**, the Huffman algorithm performs good compression for temperature, humidity, ECG and text data. For temperature, the highest saving percentage is 47% for data size of 200 bits before compression. The percentage decreases as the data size increases. A similar pattern is observed for the humidity and ECG data. This pattern is because as the branches increases, the Huffman code for each of the branches also increases. Therefore, the longer the Huffman branches, the longer the Huffman code. Thus, the saving percentage decreases.

As compared to Huffman, the LZW performs poorly for temperature, humidity and ECG data. This is because the LZW algorithm compresses the data bit-by-bit, which is inefficient for this type of data since they are already arranged in a group of bits. Processing them bit-by-bit will result in an increase in output bits for the LZW.

Table 1. Huffman, LZW, HLZ AND LZH compression performance

	Size before compression	Size after compression (Bits)				Compression ratio				Saving (%)			
Data type	(Bits)	Huffman	LZW	HLZ	LZH	Huffman	LZW	HLZ	LZH	Huffman	LZW	HLZ	LZH
Temperature	200	106	200	296	106	0.53	1.00	1.48	0.53	47.00	0.00	-48.00	47.00
	400	247	400	544	247	0.62	1.00	1.36	0.62	38.25	0.00	-36.00	38.25
	600	398	592	776	396	0.66	0.99	1.30	0.66	34.00	1.33	-29.33	34.00
	800	550	784	992	546	0.69	0.98	1.24	0.68	31.25	2.00	-24.00	31.75
Humidity	200	102	200	272	102	0.51	1.00	1.36	0.51	49.00	0.00	-36.00	49.00
	400	240	400	536	240	0.60	1.00	1.34	0.60	40.00	0.00	-34.00	40.00
	600	363	584	720	363	0.61	0.97	1.20	0.61	39.50	2.67	-20.17	39.50
	800	485	752	896	488	0.61	0.94	1.12	0.61	39.38	6.00	-12.00	39.00
ECG	200	92	184	264	88	0.46	0.92	1.32	0.44	54.00	8.00	-32.00	56.00
	400	243	384	536	237	0.61	0.96	1.34	0.59	39.25	4.00	-34.00	40.75
	600	411	584	800	404	0.69	0.97	1.33	0.67	31.50	2.67	-33.33	32.67
	800	555	776	1000	549	0.69	0.97	1.25	0.69	30.63	3.00	-25.00	31.38
Text	800	367	504	728	328	0.46	0.63	0.91	0.41	54.13	37.00	9.00	59.00
	1200	567	696	1000	491	0.47	0.58	0.83	0.41	52.75	42.00	16.67	59.08
	1600	753	840	1264	626	0.47	0.53	0.79	0.39	52.94	47.50	21.00	60.88
	2000	936	960	1480	743	0.47	0.48	0.74	0.37	53.20	52.00	26.00	62.85
	Average									42.92	13.01	-18.20	45.07

Table 2. Huffman, LZW, HLZ AND LZH compressions time

		Time taken (sec)							
	Size before Compression	Huffman		LZW		HLZ		LZH	
Data type	(Bits)	Encoder	Decoder	Encoder	Decoder	Encoder	Decoder	Encoder	Decoder
Temperature	200	0.143	0.073	0.360	0.027	0.733	0.143	0.492	0.053
	400	0.790	0.183	0.848	0.119	2.166	0.916	1.040	0.209
	600	0.481	0.669	1.298	0.098	3.568	4.587	1.684	0.353
	800	0.313	1.225	2.102	0.120	3.950	6.957	2.009	0.445
Humidity	200	0.207	0.065	0.509	0.029	0.790	0.163	0.543	0.059
	400	0.341	0.569	1.506	0.059	4.098	0.518	0.783	0.231
	600	0.230	0.279	1.863	0.096	4.036	3.838	1.473	0.229
	800	0.648	0.505	1.805	0.292	2.923	6.339	3.181	0.558
ECG	200	0.187	0.072	0.748	0.043	1.237	0.163	1.156	0.058
	400	0.586	0.300	1.151	0.068	3.814	0.531	1.429	0.137
	600	0.650	0.403	3.284	0.084	4.923	4.317	2.362	0.311
	800	0.506	0.943	2.581	0.191	3.132	7.605	4.171	0.582
Text	200	0.178	0.053	0.697	0.055	0.702	0.147	0.823	0.054
	400	0.222	0.135	1.730	0.075	3.294	0.341	1.462	0.098
	600	0.447	0.106	1.984	0.107	4.316	0.629	2.424	0.175
	800	0.446	0.136	2.046	0.171	3.837	3.263	1.926	0.372
	Average	0.398	0.357	1.532	0.102	2.970	2.529	1.685	0.245

LZW performs well for text data sizes of 800 bits, with a saving percentage of 37% being observed. The saving is observed for LZW as the data size increases. This is due to the increase in the repetition of words that match with the words inside the library. For double compression, the LZH performs better compared to the HLZ. HLZ gives lower compression results for all data types because after the Huffman algorithm, the data has been arranged into a certain pattern that is not optimized for the LZW library. However, the LZH algorithm gives better compression since the output from LZW contains a highly repetitive value. This repeated value is suitable for Huffman compressions.

Table 2 shows the result of the time taken to compress and decompress various data using the Huffman, LZW, HLZ and LZH algorithms. For the ingle data compressions, the average time taken to compress all four types of data for the Huffman is less than for the LZW. The Huffman algorithm only takes 0.398 sec, while LZW algorithm takes 1.532 sec. This is due to the Huffman algorithm being less complex than the LZW algorithm, which means it takes less time to compress the data.

For the decompression part, the average time taken for the LZW is less than for the Huffman for all four types of data. The LZW decoder takes 0.102 sec, while the Huffman decoder takes 0.357 sec. This is because the LZW decoder only needs to scan the LZW code through the library, whereas the Huffman decoder reads the input bit-by-bit, which is slower.

4. CONCLUSION

This study analyses the compression performance of the Huffman algorithm and the LZW algorithm using various input data commonly measured by a wireless sensor node, namely temperature, humidity, ECG and text data. For the given tested data, the Huffman algorithm shows better performance when compared to the LZW in terms of compression ratio and computation time. From the experiments, the Huffman algorithm is able to achieve an average of a 43% data reduction. For double compression, the LZH could provide up to 9% improvement in terms of data reduction, but at the cost of an increase in the computation time. In the future, this work will further study various techniques on WSN data representation to further increase the Huffman algorithm efficiency.

5. REFERENCES

Al-Laham, M. and I.M.M. El-Emary, 2007. Comparative study between various algorithms of data compression techniques. Int. J. Comput. Sci. Netw. Sec., 7: 281-291.

Dan, D., 2008. National environmental satellite, data and information service.

Gonzalez, R.C. and R.E. Woods, 2008. Digital Image Processing. 1st Edn., Prentice Hall, Upper Saddle River, ISBN-10: 013168728X, pp: 954.

Kelly, K., 2007. Average daily temperature archive. The University of Dayton.

Kodituwakku, S.R. and U.S. Amarasinghe, 2010. Comparison of lossless data compression algorithms for text data. Indian J. Comput. Sci. Eng., 1: 416-425.

Marcelloni, F. and M. Vecchio, 2008. A simple algorithm for data compression in wireless sensor networks. IEEE Commun. Lett., 12: 411-413. DOI: 10.1109/LCOMM.2008.080300

NIH, 2012. National Institute of Health.

Shahbahrami, A., R. Bahrampour, M.S. Rostami and M.A. Mobarhan, 2011. Evaluation of huffman and arithmetic algorithms for multimedia compression standards. Int. J. Comput. Sci., Eng. Appli., 1: 34-47.

Shanmugasundaram, S. and R. Lourdusamy, 2011. A comparative study of text compression algorithms. Int. J. Wisdom Based Computing, 1: 68-76.

SMLLC, 2013. Mother Goose Club. Sockeye Media LLC.

Strydis, C. and G.N. Gaydadjiev, 2008. Profiling of lossless-compression algorithms for a novel biomedical-implant architecture. Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Oct. 19-24, ACM Press, New York, USA., pp: 109-114. DOI: 10.1145/1450135.1450160