2025 1st International Conference on Emerging Trends in Information Systems and Informatics (ICETISI)

Lightweight Data Compression Algorithm for Climate Monitoring in Internet of Things (IoT) Application

Nor Asilah Khairi
Faculty of Electronic Engineering Technology,
Universiti Malaysia Perlis, Arau, Perlis, Malaysia
asilah.khairi91@gmail.com

Asral Bahari Jambek
Faculty of Electronic Engineering & Technology
Centre of Excellence for Micro System Technology (MiCTEC)
Universiti Malaysia Perlis, Arau, Perlis, MALAYSIA.
asral@unmap.edu.my

***Abstract*— The Internet of Things (IoT) has become widespread in our society. It is expected that 48.6 billion IoT devices will be deployed in the field by 2034. However, this large deployment will increase network traffic and data storage requirements to store the monitored data. To solve this problem, data compression is required at the sensor node to reduce the measured data before it is transmitted to the base station. Since most IoT devices have limited computational power, memory, and battery life, lightweight data compression is needed for wireless sensor nodes. In this work, we propose a modified K-RLE algorithm that reduces data redundancy while maintaining the original data after decompression. The proposed algorithm can achieve up to 39% data reduction when tested with a climate dataset. The proposed algorithm achieves 19.8% improvement in compression efficiency compared to the existing RLE algorithm, while requiring only 45% of the computational time of the Huffman algorithm.**

Keywords*— *Data Compression, K-RLE, Internet of Things, Low complexity Algorithm

I. INTRODUCTION

The Internet of Things (IoT) has become a global trend where various objects are connected via the internet and can communicate with other devices ubiquitously. These devices are very useful for remote monitoring and periodic assessment of various machines [1]. Researchers predicted the total number of IoT devices worldwide is expected to increase from 19.8 billion in 2025 to more than 40.6 billion IoT devices by 2034 [2].

One of the important IoT applications is climate monitoring, which helps in weather forecasting. Critical climate data such as severe storms, extreme winter, and hurricanes require careful monitoring to prevent information loss. Any loss of information can potentially lead to incorrect weather predictions and thus harm to life and property. Thus, climate data needs to be accurate [3,4].

However, climate data sizes are increasing rapidly [5]. This can increase network traffic and lead to insufficient data storage.

IoT devices are expected to contribute to the creation of over 181 zettabytes (ZB) of data globally by the end of 2025, with approximately 2.5 quintillion bytes generated daily [6,7,8].

One way to improve network efficiency is by implementing data compression at the sensor node [9,10]. Data compression helps to minimize data size, leading to more efficient data transfer over the network [11]. However, lossless data compression techniques can consume a long execution time, whereas lossy data compression techniques can cause unacceptable data loss [12].

Furthermore, some of the existing data compression methods have been designed specifically for certain applications, such as compressing text or images. While these methods perform well in their specific domain, the data compression algorithm for IoT still poses a great challenge. This is because the measured data on the sensor node side can vary from one application to another. Furthermore, the sensor nodes are typically driven by miniaturised devices with limited memory capacity and processing power.

In this work, a suitable data compression algorithm for IoT devices targeted at monitoring climate is studied. The main aim of this work is to develop a lightweight data compression algorithm that is suitable for sensor nodes with limited memory capacity and processing power. Based on our observation, RLE and K-RLE are good candidates to achieve this target. While K-RLE does give better data compression as compared to RLE, K-RLE results in data loss. To overcome this problem, this paper proposes an improvement to the K-RLE algorithm that can prevent data loss but still gives better data compression than RLE.

This paper is organised as follows: the Literature Review Section discusses the existing data compression algorithms, while the Methodology Section details the proposed algorithm. The datasets and experimental settings used in this work are elaborated in this section as well. The Result and Discussion Section discusses the performance of the proposed data compression algorithms against the selected dataset and existing algorithms. Finally, the Conclusion Section summarises this paper.

II. LITERATURE REVIEW

Data compression can be divided into two categories: lossy and lossless. Any compressed data using the lossy approach cannot be recovered or reconstructed exactly [13], such as MPEG, JPEG, MP3, K-RLE, etc. On the other hand, the lossless compression method is capable of performing

Authors' Final Manuscript.
Published paper link: https://doi.org/10.1109/ICETISI67983.2025.11405970

compression without any information loss. Lossless data compression algorithms are divided into two groups: the statistical compression technique and the dictionary-based technique. For example, Shannon-Fano coding, Huffman coding, adaptive Huffman coding, RLE, and arithmetic coding are examples of statistical compression techniques, whereas the Lempel-Ziv family is known as a dictionary technique [14].

Implementing data compression on the wireless sensor node allows for better network performance [9, 10]. However, data compression intended to be implemented inside wireless sensor nodes must be lightweight, as these devices have limited computational power, storage, bandwidth, and battery life [15]. Furthermore, since most data compression algorithms are originally created for space-saving, many data compression algorithms are not energy-saving. Hence, data compression with low memory usage and reduced computational effort is needed for wireless sensor nodes [16, 17].

A study by Long (2012) shows that most of the compression algorithms implemented inside wireless sensor networks use a modification of traditional compression algorithms such as S-LZW, improved LZW, static Huffman, K-RLE, and RLE (Long, 2012). On the other hand, Discrete Cosine Transform (DCT), Wavelet Transform, Principal Component Analysis (PCA), and Linear Approximation approaches have superior compression ratios and accuracy, but they come with high computational complexity [18]. This can lead to high algorithm execution time and increased battery usage.

Several existing studies have been conducted on climate data compression using lossless and lossy techniques. A study on applying a simple lossless data compression method for climate datasets based on differential and predictive coding is discussed in Mummadisetty (2015). Lossy climate compression has been the subject of recent studies. [19] studied the impact of lossy data compression on climate simulation data. [20] extended the study by implementing multiple lossy data compression techniques. [12] studied the effects of three compression algorithms on climate data, namely GRIB2 encoding, GRIB2 using JPEG 2000, and LZMA, and the simplified Application Acceleration (APAX) encoder.

The data loss from compressed climate data should be as minimal as possible to ensure that the decompressed data is the same as the original data [19, 20], as extremely small differences can result in different data interpretations [13].

A. *Run-Length Encoding (RLE)*

RLE is known for its simple data compression algorithm, which reduces data redundancy [21]. This method removes repeating characters in the input data and replaces them with symbols and run lengths [22]. For example, if a data item '*d*' occurs '*n*' consecutive times in the input, the algorithm substitutes the '*n*' occurrences with '*nd*' [16]. Figure 1 shows an example of an original text before and after compression using RLE. From this sequence, the RLE algorithm converted the *X* sequence into *X*31*. As a result, it reduces the sequence to approximately 28 characters [23].

B. *K-Run-Length Encoding (K-RLE)*

K-RLE is a modification of RLE and is known for its lossy technique. This algorithm works as follows: let *K* be a number. The algorithm replaces the '*n*' occurrences with the single pair, '*nd*', when a data item '*d*', or data between '*d*+*K*' and '*d*−*K*', occurs '*n*' consecutive times in the input data [24]. Figure 2 shows an example of K-RLE data compression.

In K-RLE, the parameter *K* represents the precision of the compressed data, where $K \geqslant 0$. However, K-RLE becomes RLE when the value of K is equal to 0 [25]. The selected value of '*K*' is very important for the K-RLE algorithm since it affects the compressed data's compression ratio and percentage loss [26]. A higher value of *K* leads not only to better compression performance but also to higher data loss after decompression.

Original Input : XXXXXXXXXXXXXXXXXXXX that's all, folks!
Compressed Data: X *20 that's all, folks!

Figure 1: Data compression by RLE (Al-laham2007).

Original Input	: 24 24 23 25 26 23
Compressed data (*K=2*)	: 24 *4 26 23.

Figure 2: Data compression by K-RLE.

III. METHODOLOGY

RLE and K-RLE are the main focus of this study because these algorithms are less complex and suitable for implementation inside IoT devices. A complex compression algorithm would cause longer computational time and higher energy usage for the sensor node. While the K-RLE algorithm has a better compression ratio, it is a lossy technique in which the original value cannot be restored after decompression. Since critical climate datasets require high accuracy, this work proposes a new algorithm known as Modified K-RLE to avoid data loss.

The proposed method modifies the existing K-RLE algorithm to avoid data loss. In the original K-RLE, any value in the range of *K* is combined and compressed together. However, in the proposed algorithm, any value in the range of *K* is replaced by special markers named '*a*', '*b*', and '*c*'. These markers help avoid data loss from happening, as in K-RLE.

Figure 3 shows the pseudocode of the Modified K-RLE algorithm. The highlighted area is the modified part that shows how the markers are being used in the proposed algorithm. The proposed algorithm works as follows. First, the proposed algorithm reads the input data and stores the first value inside a temporary integer variable as reference data, denoted as *REF*, before proceeding with the next input data. Then, the algorithm calculates the difference between the current value and the *REF* value. Let's call the difference *DIFF*. This method compresses the data if the *DIFF* value falls between -1 and 1. If the *DIFF* value is 1, the algorithm stores the character '*a*'. If the *DIFF* value is -1, then it stores the character '*b*'. Otherwise, it stores the character '*c*' if the *DIFF* value is equal to 0. If the *DIFF* is not in the range -1 to 1, the algorithm skips that step and updates the *REF* variable with the current value. The number of repetitions is counted whenever the algorithm detects the *DIFF* falls within the

range of -1 to 1 in the data series. The algorithm continues to loop until it reaches the last data.

```
Modified K-RLE algorithm
Read data inside a text file
while not end of file
    if index == 0 then
        Store current reading value inside REF variable
    else if ((current value – REF) ≥ -1) AND ((current value – REF) ≤ 1)
then
        Count repetitive value
        if ((current value – REF) == 1) then
            Store character 'a' inside array
        else if ((current value – REF) == -1) then
            Store character 'b' inside array
        else
            Store character 'c' inside array
        end if
    else
        Store repetitive value inside array
    end if
    Store current reading value inside REF
    index = index + 1
end while
```

Figure 3: Pseudocode of Modified K-RLE. REF is a temporary integer variable.

```
Original Stream   : 19 20 20 20 18 18 17 16 23 23 23 22 22 24 24 24 24
RLE               : 19 20 *3 18 18 17 16 23 *3 22 22 24 *4
K-RLE (K=1)       : 19 *6 17 *2 23 *9
Modified K-RLE    : 19 b3 a2 17 a 23 c2 b2 a4
```

Figure 4: Comparison between the RLE, K-RLE and Modified K-RLE

To show how the proposed algorithm works, Figure 4 shows the comparison of the compressed data between the proposed method, RLE, and K-RLE. In the proposed method, the value of 19 from the first value in the original stream is marked as the *REF* value in the proposed algorithm. The following data with a *DIFF* value that falls in the range of -1 to 1 is compressed by the algorithm. Thus, when the *REF* is 19, the algorithm compresses the following data that falls between 18 and 20. This proposed method labels the compressed value by using reserved alphabet characters, 'a', 'b', and 'c', as discussed before. The number of times the *DIFF* value repeats is also appended to the reserved character.

Since this work only uses three characters ('*a*', '*b*', and '*c*'), this method supports $K = 1$. To support a larger value of K would require introducing an additional marker symbol to represent the *DIFF* value that falls within that larger K. In this work, the value of K is limited to 1 since, based on our observation, using a greater value of K does not significantly improve compression performance.

A. Benchmark Data

Three climate datasets are used in this study: temperature (TMP), humidity (HMD), and sea-level pressure (SLP). The datasets were taken from [27], and are accessible to the public at the time of writing this paper. Each dataset contains 365 data points from 1st January 2017 until 31st December 2017. In these datasets, TMP and HMD contain two-digit numbers, whereas SLP contains four-digit numbers. These three climate datasets are chosen since the data contains patterns reflecting real-world measurements and are suitable to test the performance of the proposed algorithm. For each dataset, three different cases are used, which represent best, typical, and worst-case scenarios. Thus, a total of nine datasets is used to measure the performance of the proposed algorithm.

TABLE I: SUMMARY OF BENCHMARK DATASET USED IN THIS WORK

Abbreviation	Category	Data pattern	Location
TMP-A	Best-case	- uniform pattern - fewer spikes - a high number of repetitive values	Sultan Abdul Aziz Shah Air Port, Selangor, Malaysia
HMD-A			
SLP-A			
TMP-B,	Typical-case	- a moderate number of spike - a moderate number of repetitive values	Los Angeles Downtown, California, US
HMD-B			
SLP-B			
TMP-C	Worst-case	- a higher number of spikes - a less number of repetitive values	Palm Spring International, California, US
HMD-C			
SLP-C			

TABLE II: SUMMARY OF THE NINE DATASETS

Dataset	Range (Max. Point – Min. Point)	Repetitive Values
TMP-A	7	42
TMP-B	22	28
TMP-C	37	12
HMD-A	22	44
HMD-B	75	19
HMD-C	92	1
SLP-A	8	32
SLP-B	25	10
SLP-C	33	6

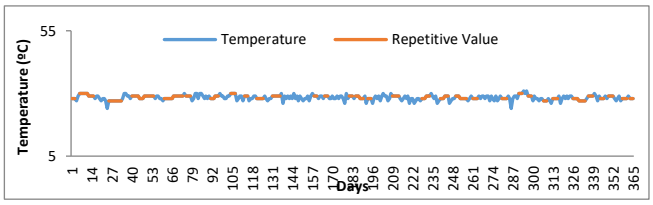

Figure 5: TMP-A: Temperature vs Number of days in Sultan Abdul Aziz Shah, Malaysia [27.

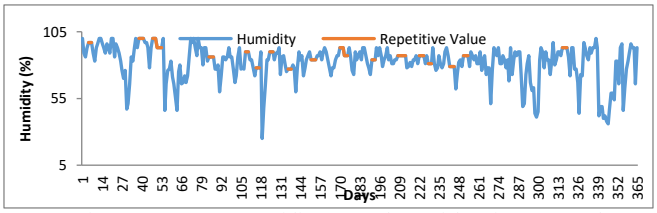

Figure 6: HMD-B: Humidity vs numbers of days in Los Angeles Downtown, US [27].

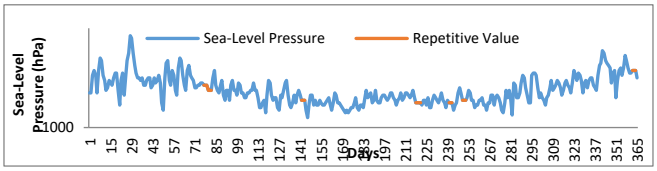

Figure 7: SLP-C: Sea Level Pressure vs Days in Palm Spring International, US [27].

$$Compression\ Ratio = \frac{Compressed\ Data\ Size}{Original\ Data\ Size} \quad (1)$$

Table I and II summarizes the characteristics of each dataset. The best-case dataset represents a dataset that has a uniform pattern, fewer spikes, and a high number of repetitive data. For a typical-case dataset, the data has a moderate number of spikes and repetitive data, whereas the worst-case contains data with a high number of spikes and fewer repetitive data. Figures 5 to 7 show the graphs for the three selected datasets.

IV. Result and Discussion

This research uses two parameters to measure the performance of the proposed compression algorithms. These parameters are compression ratio and execution run time. The performance of the proposed algorithms is compared to other lossless methods, namely RLE and Huffman, and a lossy method, namely K-RLE. The compression algorithms are written in C language using the Linux platform, running on a desktop computer with an Intel Core 2 Duo processor and 4 GB RAM.

The compression ratio is defined as the ratio between the compressed data size and the original data size. Equation 1 shows the formula to calculate the compression ratio [28]. The compressed and original data sizes are in units of bits. Data compression achieves better performance when the compression ratio value is lower.

The proposed Modified K-RLE algorithm has been tested with all nine datasets. Table III shows the compression ratio for all four compression algorithms. Based on the table, the best compression ratio achieved by the Modified K-RLE algorithm is 0.611, and the worst compression ratio is 0.942. The average compression ratio for the nine datasets is 0.825. The proposed method shows a good compression ratio for the best-case dataset since this dataset contains a high number of repetitions. With this kind of data, data reduction of up to 39% can be achieved.

Based on Table III, the proposed algorithm shows a better compression ratio compared to RLE in most of the tested datasets. The proposed algorithm can achieve 19.8% compression improvement over RLE. The improvement achieved by the Modified K-RLE is due to the use of markers to represent the repetitive data. The use of markers allows more data to be compressed more efficiently than RLE.

For K-RLE, while its average compression ratio is better than the proposed Modified K-RLE, the output is lossy. Thus, this poses significant disadvantages compared to the Modified K-RLE, especially in applications where data accuracy is crucial. Furthermore, it can be seen that the compression ratio result for K-RLE for HMD-A, HMD-B, and HMD-C is almost similar to the Modified K-RLE, with around a 2% compression ratio difference. This shows that the Modified K-RLE is equally good for this kind of data compared to K-RLE, but with no data loss.

When compared to Huffman data compression, the proposed algorithm shows a lower average performance. This is expected, as the Huffman algorithm uses statistical information to compress the data. However, for the Huffman algorithm, pre-analysis must be performed on the input data to create a proper lookup table to compress the data. This adds an additional task for the sensor node to perform this analysis. Furthermore, in some types of data, Huffman cannot perform optimally, such as TMP-C and HMD-C. Huffman performs worse than the proposed method in these types of datasets. One reason is that these datasets represent a worst-case scenario where the data has few repetitions and the values are random. Thus, Huffman cannot compress the data optimally.

Figure 8 shows the normalized run time against various amounts of the dataset for all evaluated algorithms. As predicted, Modified K-RLE has more time compared to RLE and K-RLE. The introduction of markers in the algorithm contributes to this. From the figure, Huffman consumes more run time compared to Modified K-RLE since it requires a more complex operation to encode and decode the compressed data. Based on the figure, the Modified K-RLE algorithm consumes around 45% of the computational time of Huffman.

TABLE III: Comparison results of compression ratio for Modified K-RLE and RLE

Dataset	RLE	Huffman	K-RLE	Modified K-RLE
TMP-A	0.762	0.282	0.326	0.611
TMP-B	0.896	0.586	0.616	0.795
TMP-C	0.964	0.962	0.803	0.912
HMD-A	0.696	0.283	0.677	0.688
HMD-B	0.918	0.734	0.915	0.918
HMD-C	0.997	1.630	0.989	0.995
SLP-A	0.844	0.168	0.340	0.669
SLP-B	0.959	0.281	0.721	0.896
SLP-C	0.978	0.345	0.874	0.942
Average	**0.890**	**0.586**	**0.696**	**0.825**

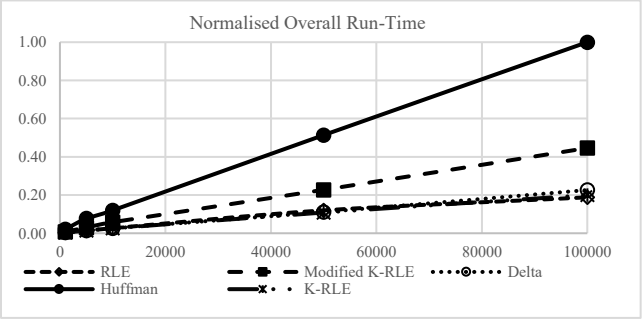

Figure 8: Normalise overall run time for various compression and decompression algorithms.

V. Conclusion

This paper discusses the proposed data compression algorithm for the Internet of Things (IoT) for climate monitoring applications. Since wireless sensor nodes typically consist of limited computational power, memory, and battery life, the data compression must be lightweight and lossless. In this work, a new algorithm called Modified K-RLE has been proposed to remove data redundancy in the wireless sensor node measured data. Based on the experimental results, the proposed algorithm can achieve up to 39% compression ratio when tested using the selected benchmark data. The proposed method outperforms the existing RLE method by 19.8%. In terms of computational power, the proposed method consumes only 45% of the computational time compared to the Huffman algorithm.

Based on the experimental results, the proposed algorithm is suitable for IoT devices, especially in removing data redundancy in the measured data before it is transmitted to the base station. To improve the compression ratio, further work

can be done by combining this method with other data compression methods, such as those that use a statistical approach or dictionary technique.

REFERENCES

[1] Vlachou, Vasileios I., Theoklitos S. Karakatsanis, Dimitrios E. Efstathiou, Eftychios I. Vlachou, Stavros D. Vologiannidis, Vasiliki E. Balaska, and Antonios C. Gasteratos. 2025. "Condition Monitoring and Fault Prediction in PMSM Drives Using Machine Learning for Elevator Applications" Machines 13, no. 7: 549. https://doi.org/10.3390/machines13070549

[2] Lionel Sujay Vailshery, "Number of Internet of Things (IoT) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2034", 2025. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[3] Xie Hongtao , Chang Mengyuan , Wang G. Geoff , Tang Yu , Jin Songheng, "Integrating climate-smart practices in forestry: insights from Europe and America", Frontiers in Plant Science, Vol (16), 2025. DOI=10.3389/fpls.2025.1583294

[4] Fathi, M., Haghi, K.M., Jameii, S.M. et al, Big Data Analytics in Weather Forecasting: A Systematic Review, Arch Computat Methods Eng, (2021)

[5] Ikegwu, A.C., Nweke, H.F., Mkpojiogu, E. et al. Recently emerging trends in big data analytic methods for modeling and combating climate change effects. Energy Inform 7, 6 (2024). https://doi.org/10.1186/s42162-024-00307-5

[6] Elias Del-Pozo-Puñal, Felix Garcia-Carballeira, Diego Camarmas-Alonso, Alejandro Calderon-Mateos," Hierarchical and distributed data storage for computing continuum", Future Generation Computer Systems, Volume 174, 2026, https://doi.org/10.1016/j.future.2025.107931.

[7] Naveen Kumar, "Big Data Statistics 2025 (Growth & Market Data)", 2025. [Online]. Available:https://www.demandsage.com/big-data-statistics/

[8] Big Data Market Report, 2025. [Online]. Available: https://www.marketdataforecast.com/market-reports/big-data-market

[9] Gashti, M.Z., Gasimov, Y.S., Farjamnia, G., New Study the Required Conditions for using in Compression WSNs During the Data Collection, International Journal of Mechatronics, Electrical and Computer Technology (IJMEC), 8(27), 3725-3735, (2018)

[10] Kimura, N., Latifi, S., A Survey on Data Compression in Wireless Sensor Networks, International Conference on Information Technology: Coding and Computing (ITCC'05), Volume II, 8–13 Vol. 2, (2005)

[11] Kaur, P., Kaur, S., Kaur, A., Big Data Compression in Mobile and Pervasive Computing, International Journal of Wireless and Microwave Technologies, 6(3), 1–8, (2016)

[12] Hubbe, N., Wegener, A., Kunkel, J.M., Ling, Y., Ludwig T., Evaluating ¨Lossy Compression on Climate Data, International Supercomputing Conference, 343–356, (2013)

[13] Sayood, K., Introduction to Data Compression 3rd Edition, Elsevier (3rd ed.), (2006)

[14] Shanmugasundaram, S., Lourdusamy, R. L., A Comparative Study Of Text Compression Algorithms, International Journal of Wisdom Based Computing, 1(3), 68–76, (2011)

[15] Kolo, J.G., Shanmugam, S.A., Lim, D.W.G., Ang, L.M., Seng, K.P., An Adaptive Lossless Data Compression Scheme for Wireless Sensor Networks, Journal of Sensors, 2012, 1–20, (2012)

[16] Long, S., Xiang, P., Lossless Data Compression for Wireless Sensor Networks Based on Modified Bit-Level RLE, 2012 8th International Conference on Wireless Communications, Networking and Mobile Computing, 1–4, (2012)

[17] Marcelloni, F., Vecchio, M., A Simple Algorithm for Data Compression in Wireless Sensor Networks, IEEE Communications Letters, 12(6), 411–413, (2008)

[18] Del Testa, D., Rossi, M.,Lightweight Lossy Compression of Biometric Patterns via Denoising Autoencoders, IEEE Signal Processing Letters, 22(12), 2304–2308, (2015)

[19] Baker, A. H., Hammerling, D.M., Mickelson, S.A., Xu, H., Stolpe, M.B., Naveau, P., Lindstrom, P., Evaluating Lossy Data Compression on Climate Simulation Data Within a Large Ensemble, Geoscientific Model Development, 9(12), 4381–4403, (2016)

[20] Baker, A.H., Xu, H., Hammerling, D.M., Li, S., Clyne, J.P., Toward a Multi-method Approach: Lossy Data Compression for Climate Simulation Data, Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, 30–42, (2017)

[21] Tiwari, B., Kumar, A. (2012). Aggregated Deflate-RLE Compression Technique for Body Sensor Network. 2012 CSI 6th International Conference on Software Engineering, CONSEG 2012.

[22] Dudhagara, C.R., Patel, H.B. (2017). Performance Analysis of Data Compression using Lossless Run Length Encoding. Oriental Journal of Computer Science and Technology, 10(3), 703–707.

[23] Al-laham M., El Emary, I.M.M., Comparative Study Between Various Algorithms of Data Compression Techniques, International Journal of Computer Science and Network Security (IJCSNS), 7(4), 281–291 (2007)

[24] Capo-Chichi, E.P., Guyennet, H., Friedt, J.M., K-RLE: A New Data Compression Algorithm for Wireless Sensor Network, 2009 Third International Conference on Sensor Technologies and Applications, 502–507, (2009)

[25] Bhadane, D.S., Kanawade, S.Y., Comparative study of RLE & K-RLE compression and decompression in WSN, 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), 1–5, (2016)

[26] Capo-Chichi, M.E.P., Friedt, J.M., Guyennet, H. (2010). Using data compression for delay constrained applications in Wireless Sensor Networks. Proceedings - 4th International Conference on Sensor Technologies and Applications, SENSORCOMM 2010, 101–107.

[27] Weather Underground. Retrieved January 3, 2017, from https://www.wunderground.com/

[28] Salomon, D. (2007). Data Compression The Complete Reference (3rd ed.). Springer-Verlag New York, Inc.