Journal of Information Systems Engineering and Management
2025, 10(10s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Proposed Method for Data Size Reduction in WSN: Optimizing Humidity, Temperature, and Pressure Sensing

Saif Al-Alak
Saif.mahmood@uobabylon.edu.iq
Dept. of Computer Science – College of Science for Women- University of Babylon- Babylon- Iraq

ARTICLE INFO

Received: 08 Nov 2024

Revised: 26 Dec 2024

Accepted: 10 Jan 2025

ABSTRACT

The utilizing of Wireless Sensor Network (WSN) for collecting data of humidity, temperature, and pressure is important to avoid natural disasters. However, the using of WSN has many challenges which are represented by node energy and power consumption as well node throughput. This study proposes a scheme to improve node life time and node throughput. The proposed scheme is coding the sensed data in a way that making an efficient compression by Huffman coding. The using of the proposed scheme with Huffman algorithm optimizes humidity, temperature, and pressure transmitted data over WSN, which produces efficient energy and power consumption as well node throughput. The testing result shows that the proposed scheme enhances network life time and network performance.

Keywords: Energy Saving, Humidity, Node Throughput, Power Consumption, Pressure, Temperature, WSN

1-Introduction

The weather elements including humidity, temperature, and pressure are important to be monitored by a system that is linked to Wireless Sensor Network (WSN). [1] The monitoring system needs to get real time information about the weather to avoid natural disasters. However, the WSN suffered from several challenges which are including inefficient node energy and power consumption as well node throughput. In WSN, the sensing and transmitting of high volume data leads to consume node energy and performance.

To address the above challenges of WSN, data compression is utilized to reduce the size of the transmitted data. The survey about data compression shows that the previous works focus on improving the network performance and life time through minimizing the size of transmitted data; however the life time and performance of WSN is still inefficient.

In [2] the author proposed a scheme called 'Mixed Byte' to treat the data of weather before compressing by Lempel-Ziv-Welch (LZW) algorithm. The scheme is based on mixing the value of successor byte with previous one to produce new byte. The produces bytes are compressed by LZW algorithm. Also, authors in [3] proposed a method that treated a data by scheme called 'Data Value Minimization' (DVM) before compressing the data by LZW algorithm. The idea of scheme is based on save the first reading of data then it keeps the incremental value of the next input data. In [4] the authors proposed a scheme to reduce the size of transmitted data, where it focuses on data aggregation, compression, and prediction. It is based on the state of the sensor to compress data for improving network performance, and life time. The authors in [5] proposed a scheme called 'S-LEC' which focuses on smooth and dynamic inputs from sensors. It focuses on getting remaining temporal information in the remaining sequences. The authors in [6] proposed a scheme which is dividing inputs to blocks and treats each block separately from others. It performs multiple coding options to compress the inputs. The authors in [7] proposed a scheme that makes enhancement on Huffman for weather data reduction. It focuses on multi scale wavelet transform for data compression in the level of channel. The authors in [8], introduced enhancement to the technique of coding for indices in the dictionary using the LZW method to improve the size reduction of sensed data, where the index was represented by its precise size. This work introduces a novel scheme to address the issue of minimizing the size of the transmitted data. Moreover, it utilizes the nature of sensed data which is repeated over a period of time.

2. Theoretical Background

2.1 Definition of Wireless Sensor Networks (WSN)

A WSN consists of interconnected sensor nodes that communicate wirelessly with a central sink. WSNs are utilized to collect information about various positions and send it to the sink. Either the sensor senses the information and sends it to the sink, or to another sensor that forwards the information to the sink. The WSNs are necessary for data collection in various life applications including smart cities and weather monitoring. The getting of information about weather (such as humidity, temperature, and pressure) in real time is necessary to make a right decision to save individuals, and other properties. [9]

2.2 Data Collection in WSN

Collecting and sending data to the destination over WSN are important challenges facing the designers of the monitoring system. Moreover, the transmission of frequent and large size data over WSN consumes the network energy and power as well performance that leads to early network failure. Also most of the collected data is repeated over periods. A key solution to these challenges lies in minimizing the size of the transmitted data through techniques such as data compression, aggregation, and filtering. [10][11]

2.3. Huffman Data Compression Techniques for WSN

The data collected in a WSN can be compressed to minimize its size using various lossless compression techniques. Two widely used methods for data compression are Huffman coding and LZW. Both of these techniques are lossless, meaning they can accurately reconstruct the original data after decompression. The idea of Huffman is based on coding the input characters with different size codes and it assigns high frequent input characters to shortest code and vice versa.

Huffman technique includes some important steps, which are frequency count, building tree, code generating, and data encoding. As shown in Figure (1), the Huffman algorithm starts by mapping all 256 characters to integers and set to zero. Then it finds the count of each character in the input string (X). Next, it is going to create a priority queue (Q) which includes cells (nodes) that has each character and its frequency. The cells should be sorted by their frequency, where highest priority cell has least frequent.

To create a Huffman Tree, it checks a priority (Q) while there are more cells (nodes), then it deletes two cells of the lowest frequency from (Q) and create new cell with deleted cells as its children at left and right. The frequency for the new created cell is computed by adding the frequency of two deleted cells. The new created cell is added to the priority queue (Q). The remaining cell is the root of the tree. A code is generated by traversing the tree (Q) for each character and encoding a string (X) is done by replacing the character (ch) with its code to get the output. The data decompressing needs to keep the tree. [12]

```
Input (X: String, n: length, d: distinct character)
Output: compressed binary and tree for decompress
1.  Begin
2.      For ch ← 1 to 255 do
3.          count[ch] ← 0
4.      Endfor
5.      For each ch in X do
6.          count [ch] ← count[ch]+1
7.      Endfor
8.  Q ← [ ]
9.  For ch ← 1 to 255 do
10.     if count[ch] > 0 then
11.     Begin
12.         Node.char = ch
13.         Node.freq = count[ch]
14.         Node.left = None
15.         Node.right = None
16.         Q ← Node
17.     End
18. Endfor
19. For i ← 1 to n-1 do
20. Begin
21.     L = Q.DeleteMin( )
22.     R = Q.DeleteMin( )
23.     Node.char = L.first + R.first
24.     Node.freq = 0
25.     Node.left = L
26.     Node.right = R
27.     Q ← Node
28. Endfor
29. For each ch in X do
30.     Output = Output+ binary [c]
31. Endfor
32. Return Output and root node
33. End
```

Figure (1): Huffman Algorithm

3-Proposed work

In this work, the proposed algorithm focuses on minimizing the repetition of identical data sensed by the sensor. Specifically, when a sensor's input remains constant over a period, the sensor node avoids repeatedly storing the same value. Instead, the algorithm increments a frequency counter for each subsequent instance of the same value, effectively capturing repeated inputs without redundancy. As illustrated in Figure (2), the input consists of raw sensing data from a specific sensor, while the output represents the encoded data, optimized for storage and transmission efficiency.

```
Input: Sensor Data
Output: codes
1.  Begin
2.      Freq ← 1
3.      Data_1 ← getDataSensor
4.      While True do
5.      BeginWhile
6.          Data_2 ← getDataSensor
7.          if (Data_1 = Data_2 ) Freq ← Freq +1
8.          else
9.          Begin
10.             output(Data_1)
11.             output(":")
12.             output(Freq)
13.             Freq ← 0
14.             Data_1 ← Data_2
15.         EndElse
16.     EndWhile
17.     output(Data_1)
18.     output(":")
19.     output(Freq)
20. End
```

Figure (2): Proposed Algorithm

Moreover, the sensor reads the first input and saves it as **Data_1**, setting its frequency value to one. The next input read is saved as **Data_2**, which is then compared with the previous input **Data_1**. If they are identical, the frequency of **Data_1** is incremented by one. Otherwise, the algorithm outputs **Data_1** and its frequency value, separated by a colon ":", and assigns the value of **Data_2** to **Data_1**.

The above steps are applied for a period of time. The Huffman algorithm is applied on the received code from the proposed scheme to minimize the size of the transmitted data.

4-Research Methodology

The proposed algorithm aims to minimize the size of the transmitted data over WSN to improve node power usage, energy consumption, and throughput. To evaluate the impact of using the proposed scheme on the size of data, a number of compression metrics are computed. The compression metrics include compression ratio and saving rate.

The utilizing of the proposed scheme in WSN influences network performance and life time, so that other network performance such as node throughput, energy savings, and power savings are calculated.

In more details, a compression ratio is computed as the rate of transmitted bytes before and after compression. A saving rate is computed as one minus a compression ratio multiply by 100%. Node throughput is computed as a rate of transmitted bytes over transmission time.

The testing is achieved on sensor node of type 'XbeeS2C' [13]. The experiment is implemented with three sensors to measure humidity, temperature, and pressure. Data sensing occurs every minute, with readings collected for a total duration of 503,910 minutes. The sensor node transmits the collected data to the sink every 60 minutes. The distance between the sensor node and the sink is 30 meters. Each data packet has a size of 30 bytes, and the transmission current (Tx Current) is 33 mA.

5-Result

This section includes two subsections: first subsection discusses the results related to data compression, and second discusses the results related to network performance and energy. The proposed work is implemented on variant types of data. It compresses the data to reduce its size. The tested data includes sensed data of temperature, pressure, and humidity by three sensors in sensor node. For temperature data, the size of sensed data is reduced about 44 times when using the proposed work for data size reduction as illustrated in Figure (3). In compare to previous works, the proposed work is better than DVM and Mixed-Byte. Furthermore, DVM reduces the temperature data size 2 times and Mixed-Byte about 4 times.

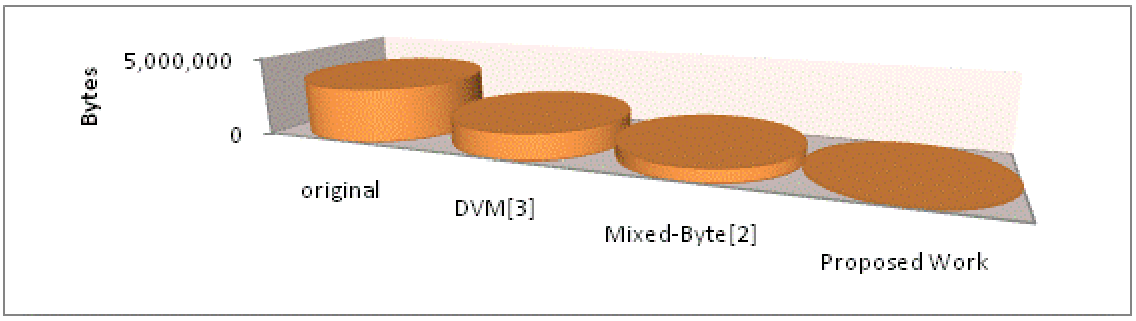

Figure (3): Temperature Data Size

For pressure data, the size of sensed data is reduced about 47 times when using the proposed work for data size reduction as illustrated in Figure (4). In compare to previous works, the proposed work is better than DVM and Mixed-Byte. Furthermore, DVM reduces the pressure data size about 3 times and Mixed-Byte about 5 times.

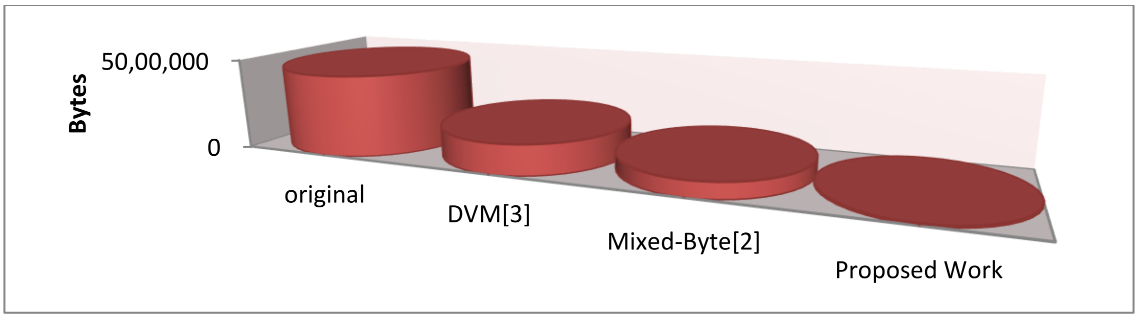

Figure (4): Pressure Data Size

For humidity data, the size of sensed data is reduced about 48 times when using the proposed work for data size reduction as illustrated in Figure (5). In compare to previous works, the proposed work is better than DVM and Mixed-Byte. Furthermore, DVM reduces the humidity data size about 2 times and Mixed-Byte about 4 times.

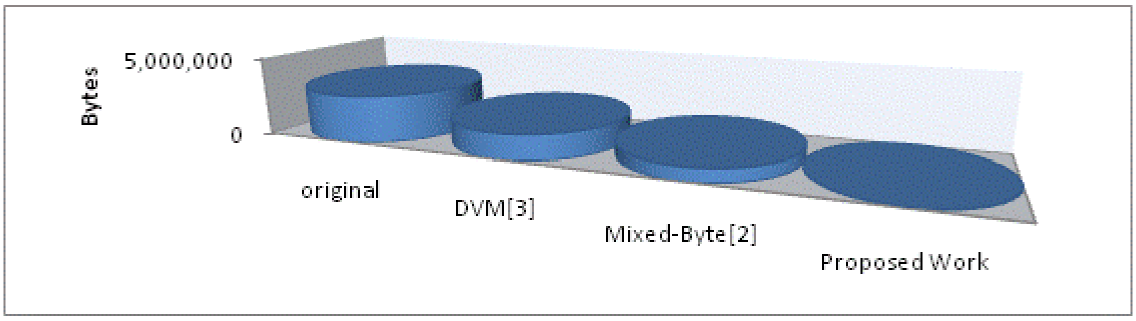

Figure (5): Humidity Data Size

The data size after compression by (proposed work-Huffman) is reduced better than previous works (DVM-LZW and Mixed-byte-LZW). For temperature data, the size of the sensed data becomes about 2600 times smaller than the original size when using the (proposed work–Huffman) for data compression as illustrated in Figure (6). Compare to previous works, (proposed work-Huffman) is better than DVM-LZW and Mixed-Byte-LZW. Furthermore, DVM-LZW compresses the temperature data size to be 37 times smaller than its original size and Mixed-Byte compresses it about 48 times.

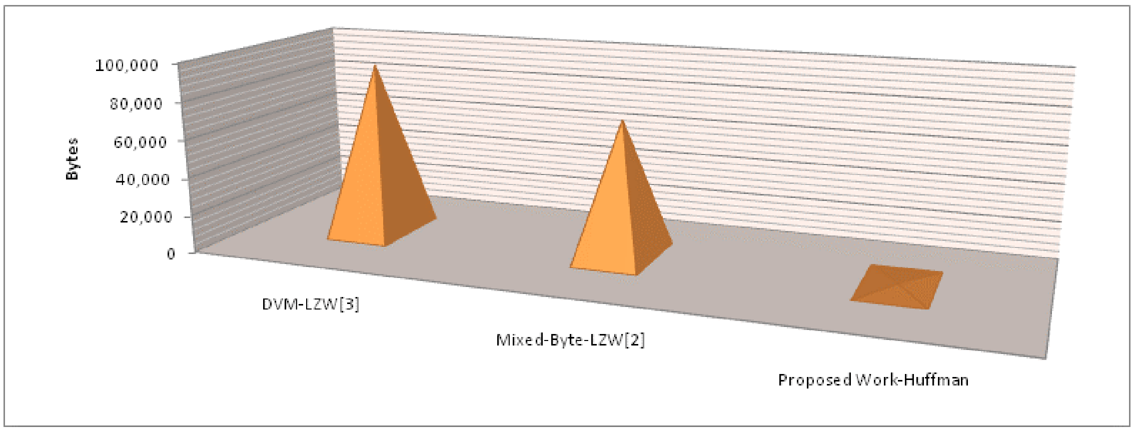

Figure (6): Compressed Temperature Data Size

For pressure data, the size of the sensed data becomes about 3500 times smaller than the original size when using the (proposed work–Huffman) for data compression as illustrated in Figure (7). Compare to previous works, (proposed work-Huffman) is better than DVM-LZW and Mixed-Byte-LZW. Furthermore, DVM-LZW compresses the pressure data size to be 61 times smaller than its original size and Mixed-Byte-LZW compresses it about 76 times.

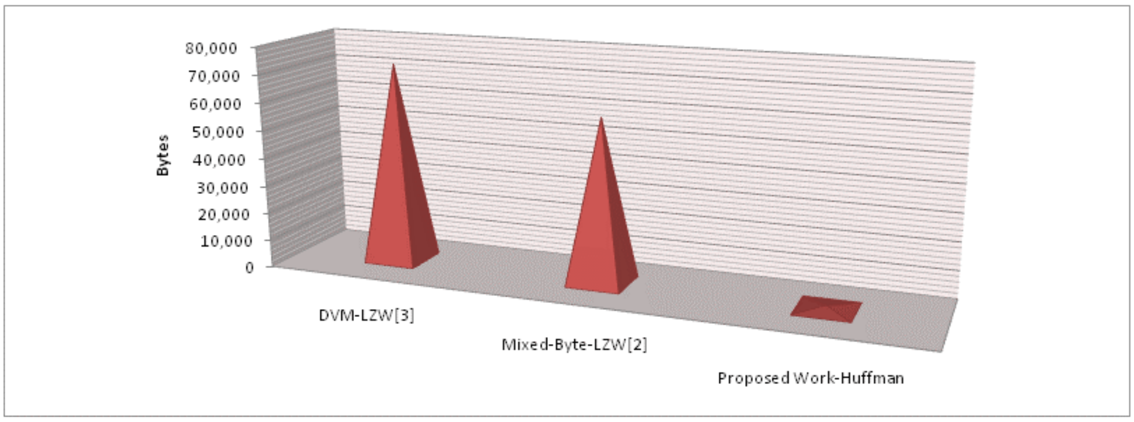

Figure (7): Compressed Pressure Data Size

For humidity data, the size of the sensed data becomes about 2239 times smaller than the original size when using the (proposed work–Huffman) for data compression as illustrated in Figure (8). Compare to previous works, (proposed work-Huffman) is better than DVM-LZW and Mixed-Byte-LZW. Furthermore, DVM-LZW compresses the humidity data size to be 92 times smaller than its original size and Mixed-Byte-LZW compresses it about 98 times.

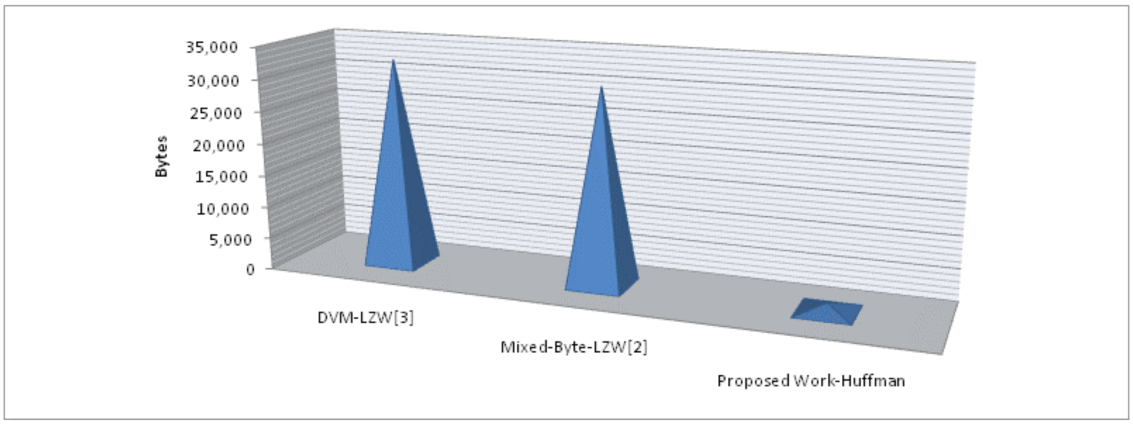

Figure (8): Compressed Humidity Data Size

The computed results refers to that the proposed work-Huffman has better compression ration than previous works DVM-LZW and Mixed-Byte-LZW as illustrated in Figure (9). Moreover, the compression ratio of proposed work-Huffman is 0.000385, 0.000278, and 0.000447 for data of temperature, pressure, and humidity respectively. The compression ratio of Mixed-Byte-LZW is 0.020845607, 0.013154666, and 0.010230208 for data of temperature, pressure, and humidity respectively. The compression ratio of DVM-LZW is 0.026640677, 0.016206147, and 0.010922175 for data of temperature, pressure, and humidity respectively.

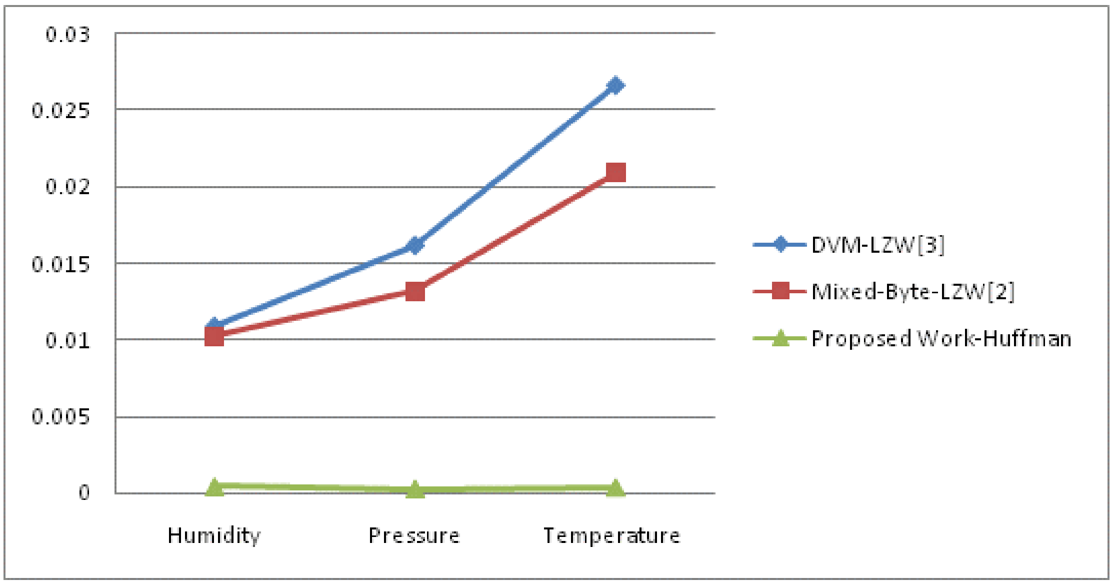

Figure (9): Compression Ratio

The computed results refers to that the proposed work-Huffman has better Saving Rate than previous works DVM-LZW and Mixed-Byte-LZW as illustrated in Figure (10). Moreover, the Saving Rate of proposed work-Huffman is 100% for data of temperature, pressure, and humidity. The compression ratio of Mixed-Byte-LZW is 98%, 99%, and 99% for data of temperature, pressure, and humidity respectively. The Saving Rate of DVM-LZW is 97%, 98%, and 99% for data of temperature, pressure, and humidity respectively.

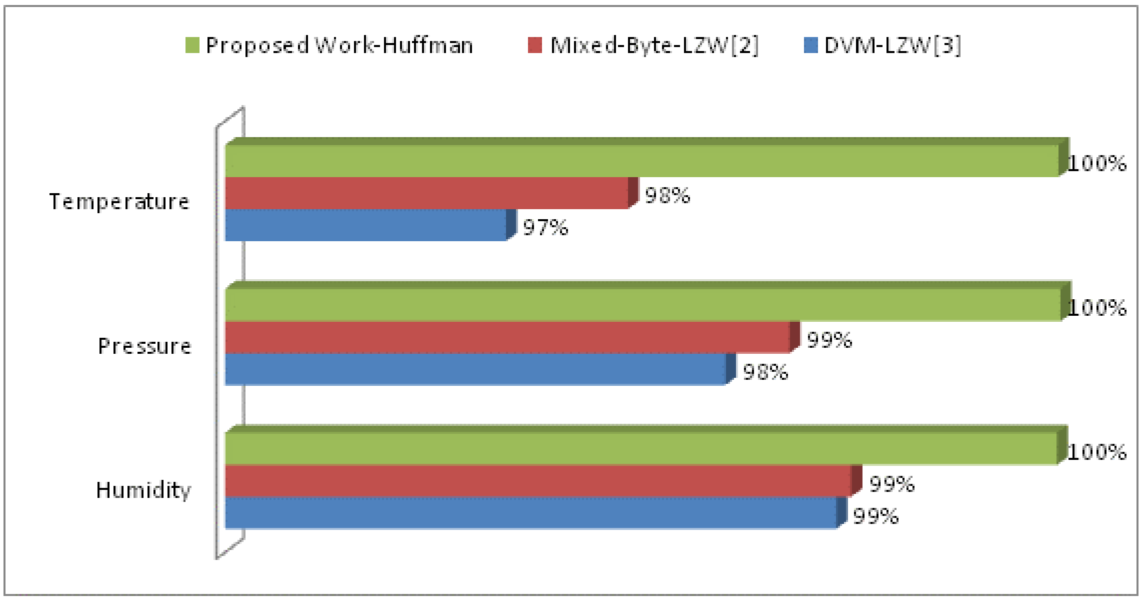

Figure (10): Saving Rate

The using of data reduction and compression techniques influences the node throughput, as well as consumed energy and power for sending and receiving data. The accomplished result shows that the using of proposed work-Huffman increases node throughput higher than DVM-LZW and Mixed-Byte-LZW. As shown in Figure (11), the node throughput of proposed work-Huffman is 15428, 21354, and 13288 for temperature, pressure, and humidity respectively. The node throughput of Mixed-Byte-LZW is 284, 451, and 580 for temperature, pressure, and humidity respectively. The node throughput of DVM-LZW is 222, 366, and 543 for temperature, pressure, and humidity respectively.

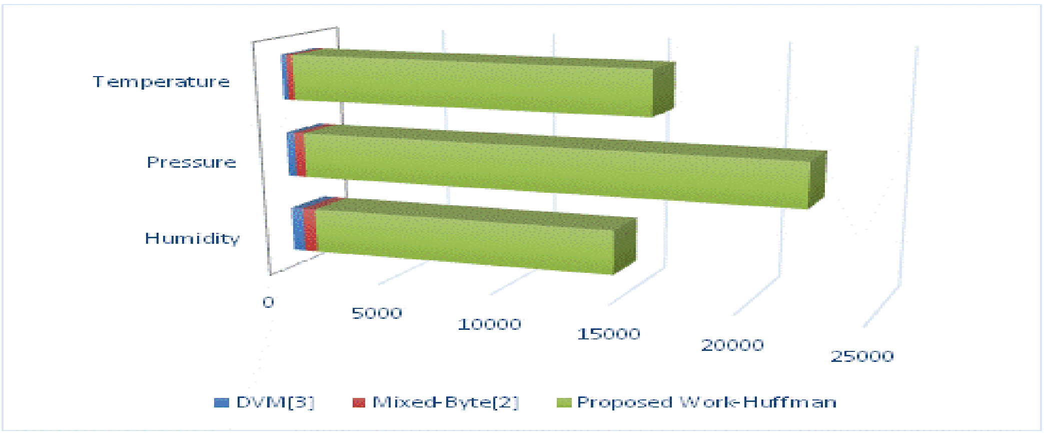

Figure (11): Node Throughput

The result shows that the using of proposed work-Huffman reduces the consumed energy and power for transferring data better than DVM-LZW and Mixed-Byte-LZW. As shown in Figure (12) and Figure (13), the saving energy and saving power of proposed work-Huffman is (44.6, 57.5, and 38.36), (12.5, 16.19, and 10.79) for compressing data of temperature, pressure, and humidity. The saving energy and saving power of Mixed-Byte-LZW is (43.7, 56.8, and 37.98), (12.3, 15.98, and 10.68) for compressing data of temperature, pressure, and humidity respectively. The saving energy and saving power of DVM-LZW is (43.4, 56.6 and 37.96), (12.2, 15.93, and 10.67) for compressing data of temperature, pressure, and humidity respectively.

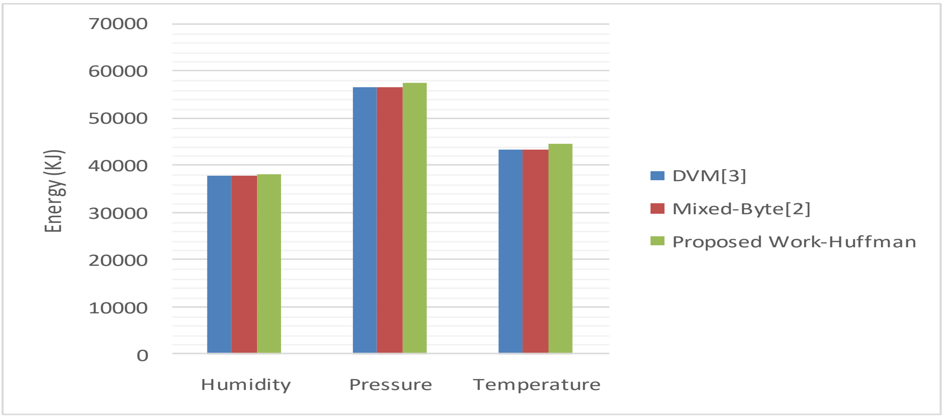

Figure (12): Node Energy Saving

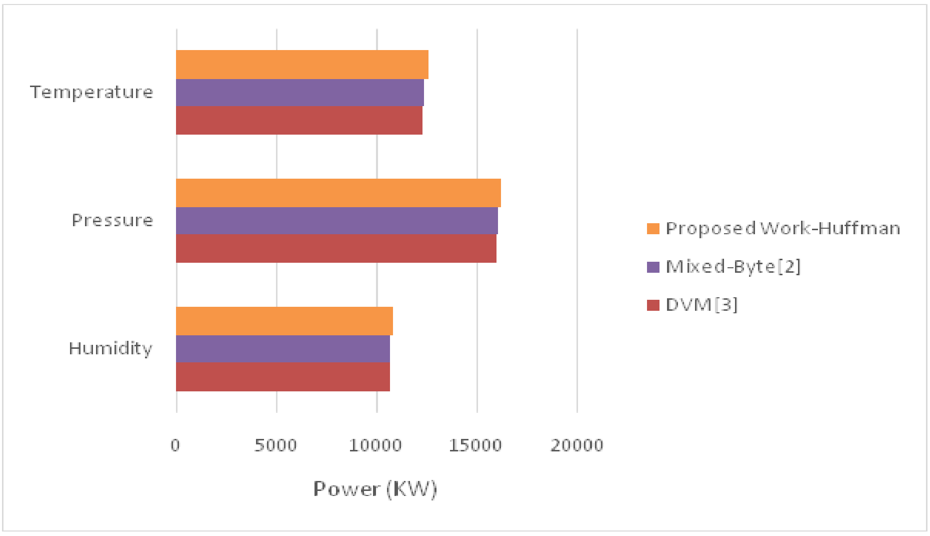

Figure (13): Node Power Saving

6-Conclusions

The proposed approach effectively reduces the size of transmitted humidity, temperature, and pressure data over WSN, resulting in enhanced node throughput, improved energy efficiency, and lower power consumption. By leveraging Huffman compression, it achieves a higher compression ratio by encoding the frequent content of the message more efficiently. This method takes advantage of the inherent characteristics of the data, utilizing the frequency patterns of humidity, temperature, and pressure readings. Integrating the proposed scheme with Huffman compression significantly enhances WSN performance and extends its overall lifespan.

References

[1] Hasan, S. S. and Y. Y. Mohammed (2020). "Environmental Monitoring Systems Using Wireless Sensor Networks: an Overview." Journal of Engineering and Sustainable Development (JEASD)(Conference proceedings 2020).

[2] Al-Alak, S. (2024). "Performance Improvement for Weather Wireless Sensor Network by Mixed-Byte Compression Scheme." Iraqi Journal of Science 65(12).

[3] M. Zahra and A. Saif, (2023). "A Developed Compression Scheme to Optimize Data Transmission in Wireless Sensor Networks," *Iraqi Journal of Science,* vol. 64, p. 14, 2023.

[4] S. Sawalha and G. Al-Naymat, (2021). "Internet of things data compression based on successive data grouping," *Turkish J. Electr. Eng. Comput. Sci.,* vol. 29, pp. 32-45.

[5] Y. Liang and Y. Li, (2014). "An Efficient and Robust Data Compression Algorithm in Wireless Sensor Networks," *IEEE Communications Letters,* vol. 18, pp. 439-442.

[6] J. G. Kolo, S. A. Shanmugam, D. W. G. Lim, L.-M. Ang, and K. P. Seng, (2012). "An Adaptive Lossless Data Compression Scheme for Wireless Sensor Networks," *J. Sensors,* vol. 2012, pp. 539638:1-539638:20.

[7] X. Xue, C. Wang, H. W. Ma, Q. H. Mao, X.-g. Cao, X. Zhang*, et al.*,(2022). "Self-Derived Wavelet Compression and Self Matching Reconstruction Algorithm for Environmental Data in Complex Space of Coal Mine Roadway," *Energies.*

[8] S. M. K. Al-alak, I. Alwan, and A. A. Hussein, (2017). "Adapted LZW Protocol for ECG Data Compression," *Journal Of University Of Babylon for Pure and Applied Sciences,* vol. 25, pp. 1618-1626.

[9] Hamza, A. H. and S. M. K. Al-Alak (2019). "Study of environmentally sustainable security in wireless sensor networks." Periodicals of Engineering and Natural Sciences (PEN) 7(4): 1722-1732.

[10] Mohammed, S., et al. (2018). "ECC and AES based hybrid security protocol for wireless sensor networks." Journal of Engineering and Applied Sciences 13(24): 10356-10363.

[11] Raju, M. R., et al. (2025). "Age and energy aware data collection scheme for urban flood monitoring in UAV-assisted Wireless Sensor Networks." Ad Hoc Networks 168: 103704.

[12] Moffat, A. (2019). "Huffman Coding." ACM Comput. Surv. 52(4): Article 85.

[13] K. F. Haque, A. Abdelgawad, and K. Yelamarthi, (2022)."Comprehensive Performance Analysis of Zigbee Communication: An Experimental Approach with XBee S2C Module," *Sensors,* vol. 22, p. 3245.