DOI: 10.24996/ijs.2023.64.3.35

ISSN: 0067-2904

A Developed Compression Scheme to Optimize Data Transmission in Wireless Sensor Networks

Zahraa Mazin[1]*, Saif Al-Alak[2]
[1] Information Networks, Information Technology, Babylon University, Babylon, Iraq
[2] Computer Sciences, College of Science for Women, Babylon University, Babylon

Received: 3/3/2022 Accepted: 16/8/2022 Published: 30/3/2023

Abstract
Improving performance is an important issue in Wireless Sensor Networks (WSN). WSN has many limitations including network performance. The research question is how to reduce the amount of data transmitted to improve network performance?
The work will include one of the dictionary compression methods which is Lempel Ziv Welch(LZW). One problem with the dictionary method is that the token size is fixed. The LZW dictionary method is not very useful with little data, because it loses many bytes when storing small-sized tokens.
From the results obtained, the best compression ratios were in the proposed algorithm. The proposed work suggests using a dynamic size token where the tokens are classified according to their size(one byte, two bytes, or three bytes). The main idea of the proposed work is based on increasing the frequency of data to increase the compression ratio. To increase the frequency of data, the work suggests keeping the amount of incremental reading data instead of keeping the whole real data. Because the climate reading data changes very slowly, the amount of change would be frequent.

Keywords: WSN, weather, LZW, Data Compression, Network Lifetime.

مخطط ضغط مطور لتحسين نقل البيانات في شبكة المستشعرات اللاسلكية

زهراء مازن[1]*, سيف العلاق[2]
[1]شبكات المعلومات, تكنولوجيا المعلومات, جامعة بابل, بابل, العراق
[2]علوم الحاسوب, كلية العلوم للبنات, جامعة بابل, بابل, العراق

الخلاصة
يعد تحسين الأداء مسألة مهمة في شبكة المستشعرات اللاسلكية. الشبكة لديها العديد من القيود بما في ذلك أداء الشبكة. سؤال البحث هو كيفية تقليل كمية البيانات المرسلة لتحسين أداء الشبكة؟
سيتضمن العمل إحدى طرق ضغط القاموس . تتمثل إحدى مشكلات طريقة القاموس في أن حجم الرمز المميز ثابت. طريقة قاموس ليست مفيدة جدًا مع القليل من البيانات ، لأنها تفقد العديد من وحدات البايت عند

*Email: zhraamazin422@gmail.com

تخزين الرموز صغيرة الحجم. من النتائج التي تم الحصول عليها ، كانت أفضل نسب الضغط في الخوارزمية المقترحة. يقترح العمل المقترح استعمال رمز حجم ديناميكي حيث يتم تصنيف الرموز وفقًا لحجمها إلى (بايت واحد ، اثنان بايت ، ثلاثة بايت). تعتمد الفكرة الرئيسية للعمل المقترح على زيادة البيانات المتكررة لزيادة نسبة الضغط. لزيادة البيانات المتكررة ، يقترح العمل الاحتفاظ بكمية بيانات القراءة المتزايدة بدلاً من الاحتفاظ بالبيانات الحقيقية بأكملها. لأن بيانات قراءة المناخ تتغير ببطء شديد ، بحيث يكون مقدار التغيير متكررًا.

1. Introduction

The growing use of WSNs in many fields (healthcare, smart homes, agriculture, industrial, and military) has increased researchers' interest in this type of network. The network in WSN consists of a group of nodes, where each node is sensing data from its surroundings and sending the messages to a sink or another neighbour. The processing capacity varies with each node Restricted battery resources, sensors have storage capacity which is limited, range of radio communication and reliability, etc. In WSN, energy is lost by receiving and sending data as well as processing it. While the energy consumption of data processing is much less than that of data transmission, reducing energy consumption has become extremely important. There are many technologies that lead to energy conservation by reducing the amount of data sent and received (data compression). [1]

The main challenge today in this field is to improvise the power and energy management of sensor networks. For this reason, the aim is to develop an algorithm for data compression. Dictionary-based compression algorithms are based on a dictionary instead of a statistical model. A dictionary is a set of possible words in a language that is stored in a table like structure and uses the indexes of entries to represent larger and repeating dictionary words. The LZW algorithm is one such algorithm. In this method, a dictionary is used to store and index string patterns. In the compression process, those index values are used instead of repeating string patterns. The LZW is a lossless algorithm. A dictionary is created dynamically in the compression process, and there is no need to transfer it with the encoded message for decompression. In the decompression process, the same dictionary is created dynamically. Therefore, this algorithm is an adaptive compression algorithm.

The contributions made by the research are that, first, the data has been pre-processed so that the first value is fixed and the difference between the first and second value will be the new value, so the representation of the new value will need fewer bits. There are five classes (A, B, C, D, and E), where each token is 1, 2, 3, 4, or 5 bytes in size, respectively.
The research is arranged as follows: the next part presents the relevant work; the third part is background on data compression; the fourth part is the proposed method; and the fifth is the methodology; then the results are examined and discussed.

2- Literature Survey

Many of the previous projects about optimizing network performance by compressing data will discuss some of them:
[1] proposed a set of technologies to provide energy in a network (WSN) of Data Reduction based on Compression Technique(DRCT). DRCT consists of two stages. The first is Symbolic Aggregate approximation(SAX)quantization which reduces dynamic range, then lossless LZW compression for lossy quantization output. It uses an object-oriented modular discrete event network simulator (OMNeT++)and data is collected from reality.
[2] explained and compared different compression methods to improve the performance of a wireless sensor network. [3] suggested and compared a set of lossless compression algorithms (Huffman Encoding, The Shannon Fano Algorithm, Arithmetic Encoding, and LZW). [4]

suggested and compared a group of popular compression algorithms (Run Length Encoding, Burrows- wheeler transform, Shannon-Fano coding, Huffman coding, Arithmetic coding, Lempel-Ziv Welch and Bit-Reduction algorithm).
[5] proposed a group of lossless compression algorithms (Bit Reduction algorithm, Huffman Coding, Run Length Encoding, Shannon-Fano coding, Arithmetic Coding, LZW, and Burrows-Wheeler Transform). [6] suggested a method of compression and encryption of the transmitted data for the purpose of reducing its size when the recipient is reassembled and decrypted. [7] suggested the segmentation method along with the proposed algorithm for the purpose (Huffman) of reducing the size of the transmitted data and thus saving energy.

3- Theoretical Background

3.1 WSNs Concept

A wireless sensor network is simply a network of wireless sensors that communicate with each other in different wireless way[8].WSN has created a wide field of research for long-term physical environmental monitoring where energy consumption is the main concern so the effort is focused on reducing the amount of data and thus reducing energy consumption[9]. The amount of sensing and processing of data, sending it to the base station, and receiving it are all things that must be taken into account with regard to energy consumption, Wireless Sensor Networks (WSN) are made up of a collection of devices known as motes. Each mote consists of a processor, internal memory, RAM or ROM, a wireless transceiver, a power source, and one or more sensors, and perhaps in some of them the Global Positioning System (GPS), because determining the location is important for receiving data, and the processor usually runs an operating system that supports network protocols[8].

3.2 LZW Compression

The widespread use of computing and the Internet led to an explosion in the amount of data transmitted, which led to the emergence of the need to compress data. The compression ratio is an important feature in data compression, as it reduces the file size, making it easier to store and transfer over the Internet. For data compression, there are two types of compression: "lossless" and "lossy". Lossless data compression is used when it is necessary to retrieve data exactly identical to the data before compression, such as text files, where the loss of one character leads to making the data misleading, and also video, image, and audio files must be lossless. While compression with loss is considered not to be necessary for the stored data to be perfect, each method has its uses. Lossless pressure is better in some cases and in other cases lossy pressure is better. In order to obtain a better result, the two methods are used, as well as the contents of the file in terms of repetition in the data, which greatly affects the compression ratio [9]. Data compression techniques can be categorized as follows: data quality, coding schemes, data type and application suitability [10].

In 1980, Terry Welch invented the LZW algorithm which is one of the most important and popular compression algorithms. It is one of the powerful compression techniques that provides high lossless compression efficiency. The LZW algorithm depends on the dictionary where a dynamic dictionary is created. For example, when using the standard code (ASCII), the dictionary will contain strings consisting of one number, so there will be 256 entries. The LZW algorithm will compare the incoming strings and search for each incoming character until it reaches a sub-string that is not in the dictionary where it is added to the dictionary with the next symbol provided. For example, if it has the sequence (abbacsd) in the beginning it will take until (0) is generated in the dictionary and then (ab) will be added and added as an index to dictionary el al[11].

The LZW algorithm is simple as the dictionary is not transmitted during the transmission process and both the sender and receiver will get the same dictionary dynamically. In the LZW algorithm, all the individual items must be initially added to the dictionary, which does not work with little data [1].

There are three main steps to compress text files using the LZW algorithm[12]:

- First, put all the individual letters.
- Compare the first letter w of the given strings with the dictionary.
- Compare the second letter k if present and the end of the string terminating, the opposite if wk does not exist is added to the dictionary with the provided index.

The limitations of LZW [1] :

- LZW algorithm is only suitable for text files.
- Initially, when creating the dictionary, all individual letters must be placed.
- A fixed token is used to store the index of character in dictionary

3.3 Raspberry Pi 4 model B (4GB)

It was developed by the Raspberry Pi Foundation (raspberrypi.org) and was first released in 2012 (see Figure 1). Several generations of it have been produced that can be classified into (The Raspberry Pi A, B ,and Zero). Each unit consists of a central processing unit (CPU), an on chip graphics processing unit (GPU), internal memory, a 5V DC power input, and a dedicated place to connect the camera. In addition to the general-purpose I/O ports, Ethernet and wireless (Bluetooth and WiFi) connectivity are also available. A Raspberry with these specifications is a small sized computer with lower costs.

The Raspberry Pi 4 model B**,** was released in June 2019 with a 1.5 GHz 64-bit quad core ARM 3.0 Cortex-A72 processor, on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet (throughput not limited), two USB 2.0 ports, two USB 3.0 ports, 2-8 GB of RAM, and dual-monitor support via a pair of micro HDMI (HDMI Type D) ports for up to 4K resolution [13].

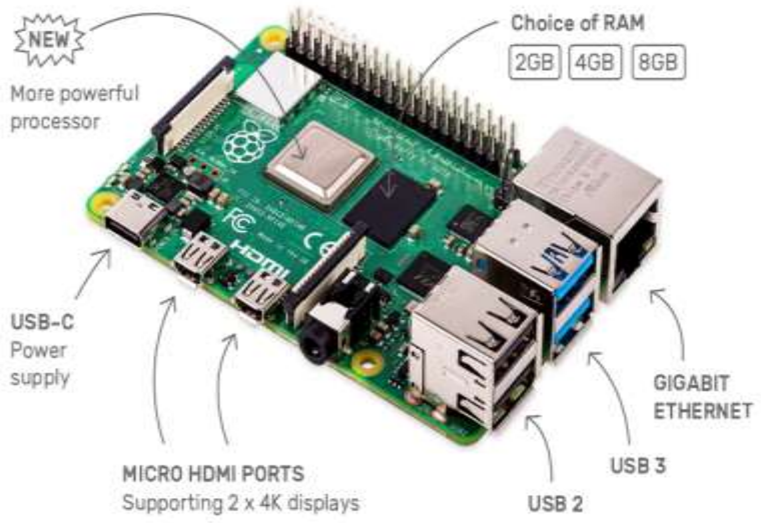


Figure 1: Raspberry pi 4 model B

The Raspberry Pi Foundation offers a Debian-based (32-bit) Linux distribution for download, as well as third-party Ubuntu, Windows 10 IoT Core, RISC OS, and LibreELEC. It also promotes Python and Scratch as the primary programming languages on the Raspberry, but it also supports many languages [14].

3.4 Zigbee protocol

IEEE 802.15.4/ZigBee is a wireless network standard that has the three characteristics of low power consumption (see Figure 2), low cost, and low data rate (250 kB/s) [15].

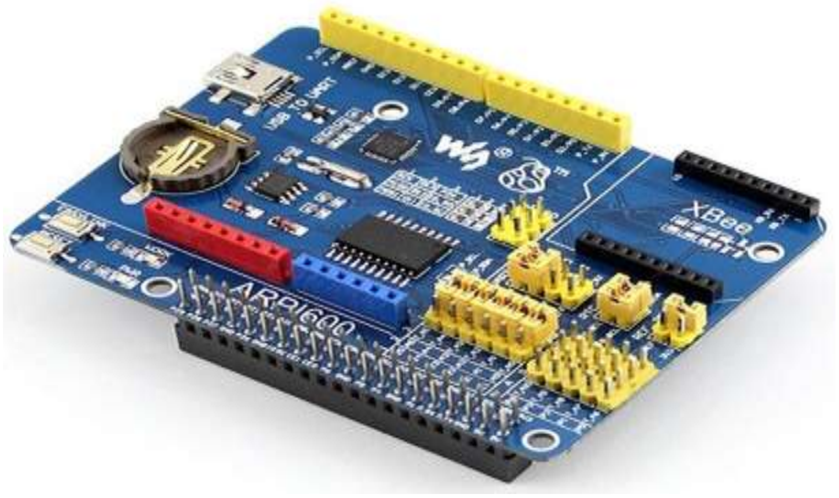


Figure 2: Raspberry pi Arduino Shield Xbee Expansion Hat

4- Proposed Method

Most of the sensed data in WSN is related to the weather, which includes heat, humidity, atmospheric pressure, etc. The data would be represented as an integer number. It means it needs one signed byte for binary representation. To address the problem of data representation, a proposed scheme does not deal with the real value; it computes the difference between the previous value and the new value. The difference between two values would be less than both, so when representing a new value, it would need fewer bits. An example to illustrate the idea is demonstrated in example 1.

Example 1:

When the degrees of somewhere at time (t1, t2, .., t7) are (25, 26,27, 27, 27, 28, 27) respectively, then the binary representation for degrees is as shown in the third column in Table1.

The difference between the new and old value starting from the original is shown in the fourth column, and binary representation is shown in the fifth column.

Table 1: Example1 data representation

Time	Data in Decimal	Binary (byte)	Difference	Binary (Signed byte)
t1	25	00011001	25	00011001
t2	26	00011010	+1	00000001
t3	27	00011011	+1	00000001
t4	27	00011011	0	00000000
t5	27	00011011	0	00000000
t6	28	00011100	+1	00000001
t7	27	00011011	-1	11111111

From example 1, it is possible to say that the difference between the original and new value could be represented by less than one byte.

To complete the addressing of the problem, the authors propose to use one of the dictionary compressions schemes like LZW (Lempel Ziv Welch) to compress the computed values. It is expected to get a high compression ratio because most of the computed values are of high similarity.

To address the problem of the fixed size token in LZW, the authors propose using a dynamic size token. The proposed work; classifies tokens into five classes (A, B, C, D, and E), where the size of each token is 1, 2, 3, 4, and 5 bytes, respectively. Each category is differentiated from the other based on the first byte. Furthermore, the left-most bits give the token category, as shown in Table 2. The Left Most Bit (LMB) of token belongs to class A is (1). The LMBs of a token belonging to class B are (0,1). The LMBs of a token that belonging to class C are (0,0,1). The LMBs of tokens that belong to class D are (0,0,0,1). The LMBs of a token belonging to class E are (0,0,0,0,1).

Table 2: LZW token categories

Token Class	Binary Representation	Token Size
A	1,-,-,-,-,-,-,-	1 byte
B	0,1,-,-,-,-,-,- -,-,-,-,-,-,-,-	2 bytes
C	0,0,1,-,-,-,-,- -,-,-,-,-,-,-,- -,-,-,-,-,-,-,-	3 bytes
D	0,0,0,1,-,-,-,- -,-,-,-,-,-,-,- -,-,-,-,-,-,-,- -,-,-,-,-,-,-,-	4 bytes
E	0,0,0,0,1,-,-,- -,-,-,-,-,-,-,- -,-,-,-,-,-,-,- -,-,-,-,-,-,-,- -,-,-,-,-,-,-,-	5 bytes

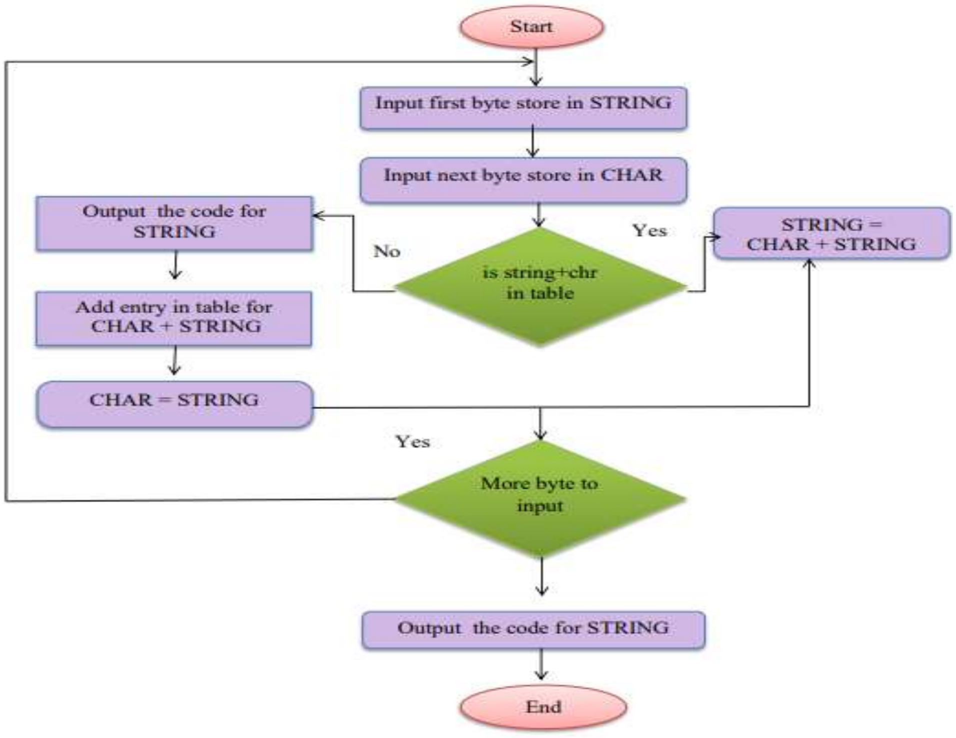
When the compression ratio is high it means a WSN system uses fewer messages for transferring more data than original state.

This algorithm has two inputs: STRING and CHAR (see Figure 3). Data is compressed by reading the first character(STRING) and the second character(CHAR). If STRING + CHAR is in the dictionary, the index is taken for compression purposes, but if it doesn't exist, then it will be added as New index in the dictionary. This index is compressed with a fixed token

Algorithm 1: Original LZW [1]

Input : string STRING ;character CHAR.
Output: encode STRING to output file .

```
1  STRING ← empty string
2  while (there is still data to be read)
3      CHAR ← read a character
4      if (dictionary contains STRING + CHAR)
5          STRING ← STRING + CHAR
6      else
7          encode s to output file
8          add STRING + CHAR to dictionary
9          STRING ← CHAR
10     end
11     end
12 end
13 return STRING
```



Start
Input first byte store in STRING
Input next byte store in CHAR
is string+chr in table
Yes
No
STRING = CHAR + STRING
Output the code for STRING
Add entry in table for CHAR + STRING
CHAR = STRING
Yes
More byte to input
Output the code for STRING
End

Figure 3: Flowchart of the Original Algorithm

Algorithm 2: Original Algorithm and pre-processing data

Input : string STRING, string x ,character CHAR.
Output: encode STRING to output file.

```
Begin
1   STRING ← first value
2   x ← next value
3   if(STRING ← x)
4       STRING ← 0
5     else
6         STRING ← x - STRING
7     end
8   end
9   while (there is still data to be read)
10      CHAR ← read a character
11      if (dictionary contains STRING+ CHAR)
12          STRING ← STRING + CHAR
13          else
14              encode STRING to output file
15              add STRING + CHAR to dictionary
16              STRING ← CHAR
17          end
18      end
19  end
20  return STRING
```

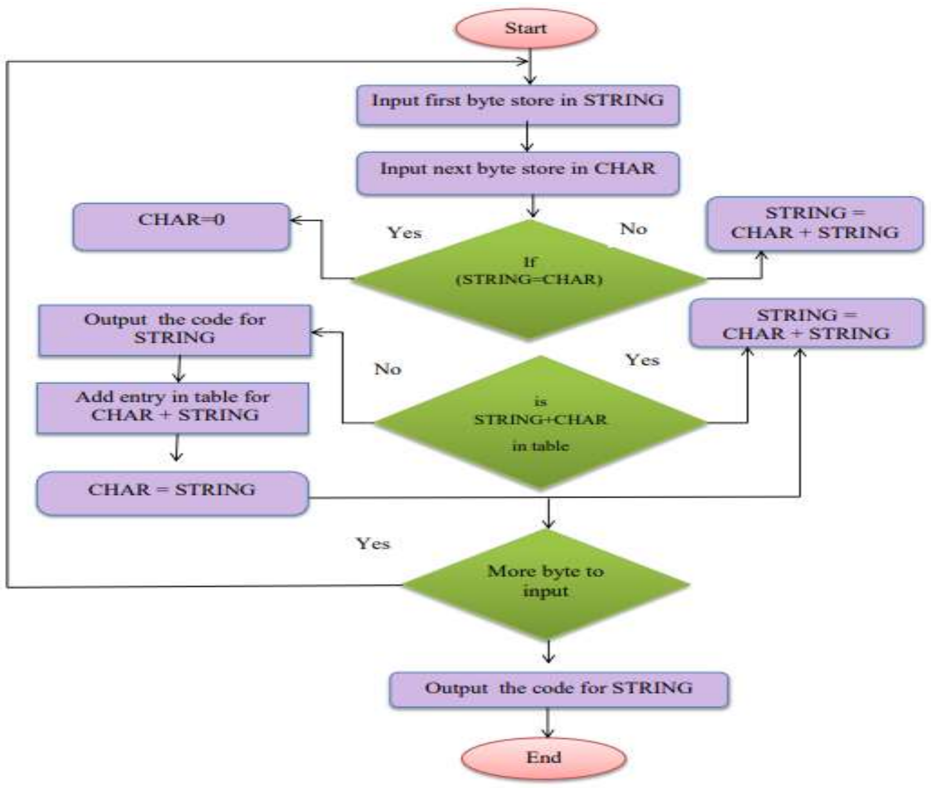

Start
Input first byte store in STRING
Input next byte store in CHAR
CHAR=0
Yes
No
If (STRING=CHAR)
STRING = CHAR + STRING
Output the code for STRING
No
Yes
is STRING+CHAR in table
STRING = CHAR + STRING
Add entry in table for CHAR + STRING
CHAR = STRING
Yes
More byte to input
Output the code for STRING
End

Figure 4: Fowcahrt of Original Algorithm and pre-processing data

In the beginning, it also had two entries which were STRING and CHAR. The data is compressed by reading the first character(STRING) and the second one (CHAR). If STRING + CHAR is present in the dictionary, the index is taken for the purpose of compressing it, but if it does not exist, the index is added as New in the dictionary. This index is compressed with a variable token according to the size of the index. It is classified into four classes: A, B, C, and D. The size of A is one byte, B is two bytes, C is three bytes, and D is four bytes.

Algorithm 3: Proposed Algorithms

Input : string STRING , character CHAR.
Output: encode STRING to output file.

```
1   STRING ← empty string
2   while (there is still data to be read)
3       CHAR ← read a character
4       if (dictionary contains STRING + CHAR)
5           STRING ← STRING + CHAR
6           else
7               encode s to output file
8               add STRING + CHAR to dictionary
9               STRING ← CHAR
10          end
11      end
12      classify STRING to class A,B,C,D,E,F
13  end
14 return STRING
```

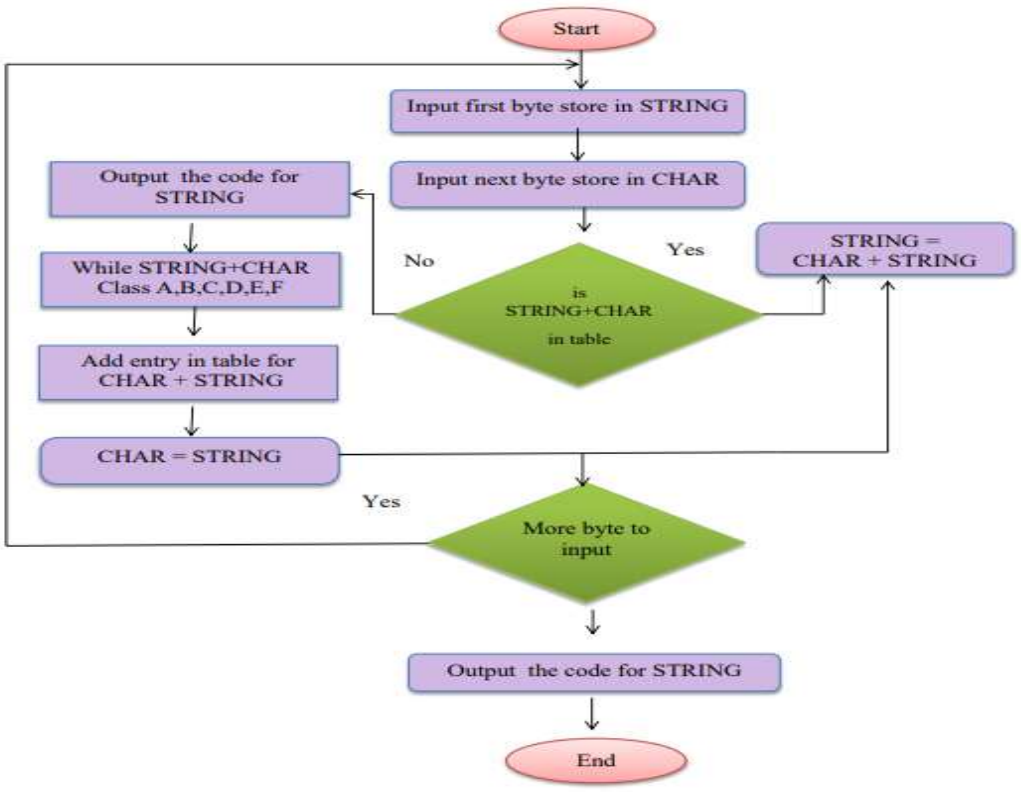

Start
Input first byte store in STRING
Input next byte store in CHAR
is STRING+CHAR in table
No
Yes
STRING = CHAR + STRING
Output the code for STRING
While STRING+CHAR Class A,B,C,D,E,F
Add entry in table for CHAR + STRING
CHAR = STRING
Yes
More byte to input
Output the code for STRING
End

Figure 5: Flowchart for the Proposed Algorithms

Algorithm 4 : Proposed Algorithms and pre-processing data

```
Input : string STRING , string x ,character CHAR.
Output: encode STRING to output file.
Begin
1    STRING ← first value
2    x ← next value
3    if(STRING ← x)
4        STRING ← 0
5        else
6            STRING ← x- STRING
7        end
8    end
9    while (there is still data to be read)
10       CHAR = read a character
11       if (dictionary contains STRING + CHAR)
12           STRING ← STRING + CHAR
13           else
14               encode STRING to output file
15               add STRING + CHAR to dictionary
16               STRING ← CHAR
17           end
18       end
19       classify STRING to class A,B,C,D,E,F
20   end
```

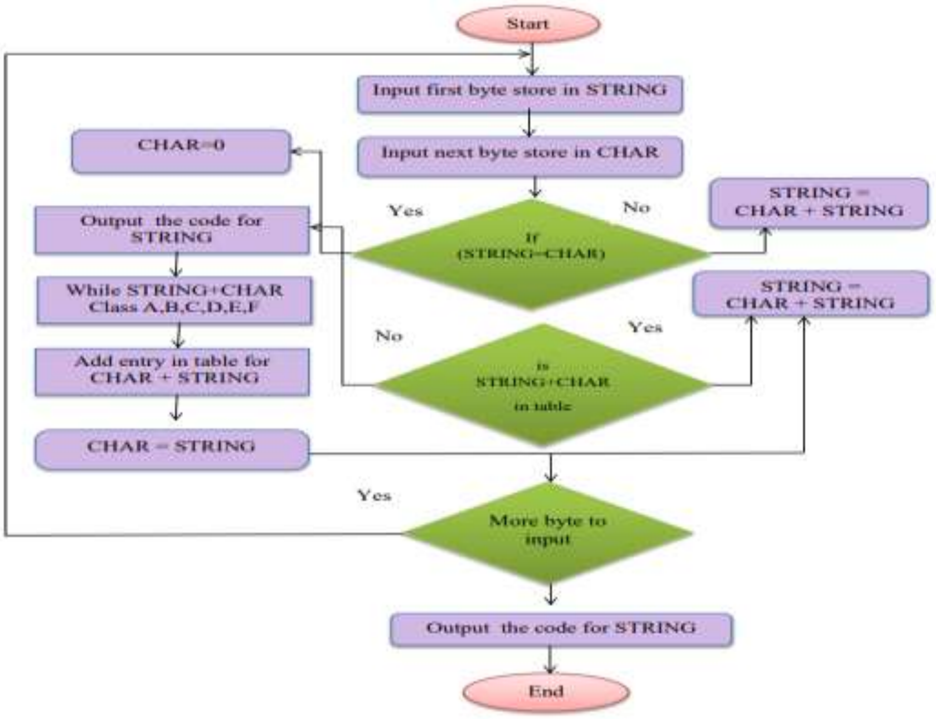

Start
Input first byte store in STRING
Input next byte store in CHAR
CHAR=0
Yes
No
If (STRING=CHAR)
STRING = CHAR + STRING
Output the code for STRING
While STRING+CHAR Class A,B,C,D,E,F
Add entry in table for CHAR + STRING
CHAR = STRING
No
Yes
is STRING+CHAR in table
STRING = CHAR + STRING
Yes
More byte to input
Output the code for STRING
End

Figure 6: Flowchart of Proposed Algorithms and pre-processing data

5- Methodology

5.1 Material

The proposed work is being implemented by collecting real data from the sensor network in Al-Hilla, Al-Moallem District, at 10 am on 10/3/2022. Work is carried out with the following **materials:**

1- Two Raspberry pi 4 models B were purchased with (4GB), two Xbee Shield, breadboard, temperature, humidity and rain sensors.
2- Ethernet cable, USB cable, mouse, keyboard, HDMI cable, mini HDMI cable.

5.2 Network Installation

❖ **Software**

1- Install and set up Raspberry Pi OS for two Raspberry pi 4 models B (4GB).
2- Install VNC viewer on the laptop for the purpose of accessing and controlling the raspberry
3- Install the NetBeans program for the purpose of programming the algorithms.

❖ **Hardware**

- **Sender**

1- Install Xbee Shield on Raspberry.
2- Connect the sensors to the raspberry transmitter, where a wire was connected to pin 5V and another to the GND pin. The humidity and temperature sensors were connected to pin P1, while the rain sensor was connected to pin P4.
3- Connect the Ethernet cable between the laptop and the Raspberry for the purpose of Internet connection.
4- Connecting a USB cable for the purpose of supplying power to the Raspberry.

- **Receiver (Sink)**

1- Install Xbee Shield on Raspberry.
2- Connecting a USB cable for the purpose of supplying power to the Raspberry.
3- Connect mouse and keyboard to Raspberry for the purpose of controlling it.
4- Connect mini HDMI from TV to raspberry.

5.3 Experiments

Three Experiments is implemented:

1- The data will be read every hour (3600 seconds). Each value is stored in one byte (128_-127) for each of the humidity, temperature, pressure and rain (4 bytes). It sent (3,600) bytes every hour. The first clock contains 24 readings (each reading has 150 values) so 24 * 150 = 3600, stored in a text file and sent to another raspberry.
2- The second clock contains 48 readings (each reading has 150 values) so 48 * 150 = 7,200, stored in a text file and sent to another raspberry.
3- The third one that contains 72 readings (each reading has 150 values) for this 72 * 150 = 10,800, stored in a text file and sent to another raspberry.

Five tests are conducted over the three hours:

1-First, the data is sent without any compression in order to compare the results after compression.
2-Compress and send the data using the traditional LZW algorithm. Notice the difference between the first and second sending.
3-The difference between the previous value and the new value was calculated using pro-processing code and stored in new text files, then compressed with the original LZW and sent to the sink.
4-In the fourth test, the original data is compressed and sent using the proposed algorithm.
5- The final test was performed on new files whose data had been modified using pre-processing code and compressed with the proposed algorithm.

5.4 Metrics

Three metrics are computed to evaluate the proposed work. To define the **compression ratio**, first it needs to know the "ratio". The ratio is a comparison of two numbers, so the compression ratio is the size of the data before and after compression [16].

Through the following mathematical eq.:

$$\text{Compression Ratio} = \frac{\text{size after compression}}{\text{size before compression}} \qquad (1\text{-}5)$$

Throughput is also calculated**.** It is the amount of data that the system can process during a certain period of time, so it is possible to measure the speed of data transfer between two systems [16].

$$\text{Throughput} = \frac{\text{The amount of data transmitted}}{\text{time}} \qquad (5\text{-}2)$$

Saving Percentage calculates the shrinkage of the source data as a percentage [16].

$$\text{Saving percentage} = \frac{\text{size before compression} - \text{size after compression}}{\text{size before compression}} \qquad (5\text{-}3)$$

6- Results and Discussion

In this section, the proposed algorithm is evaluated using real data collected from reality using weather sensors.

The proposed algorithm is compared to the original algorithm by the difference in file size.

Table 3 Compressed data size

No. hour	1 hours	2 hours	3 hours
Original Data size /Byte	8,048	13,392	**26,784**
Size (Byte) of Compressed Data (Original LZW)	1,229	1,974	**2,566**
Size (Byte) of Pre-processed and Compressed Data (Original LZW)	834	1,229	**1,749**
Size (Byte) of Compressed Data (Proposed LZW)	1,306	1,949	**2,494**
Size (Byte) of Pre-processed and Compressed Data (Proposed LZW)	814	1,229	**1,687**

By noting Table 3, it can be seen that the data size has decreased when compressing it with the proposed algorithm.

The following (Table 4) shows the compression ratio of the data in the above table.

Table 4: Compression Ratio

No. hour	1 hours	2 hours	3 hours
Original LZW	0.152	0.147	**0.096**
Preprocessing data with Original LZW	0.103	0.092	**0.043**
Proposed LZW	0.162	0.156	**0.065**
Preprocessing data with Proposed LZW	0.101	0.092	**0.063**
Compression Ratio			

The LZW algorithm does not work well for large data, since it has to maintain a huge dictionary for compression and decompression processors.

By observing the results in Table 4, it can be noticed that the amount of compression increases with the increase in file size.

Also, the time taken to send compressed and uncompressed data must be taken into account, as the time difference that occurs when sending compressed and uncompressed data is of great importance in calculating the success of the proposed algorithm.

Table 5 shows is the time taken to send each data item:

Table 5: Time of sending

No. hour	1 hour	2 hours	3 hours
Original Data	461 s	922 s	**1,383 s**
Compressed Data (Original LZW)	31 s	63 s	**83 s**
Pre-processed and Original LZW	20 s	31 s	**51 s**
Proposed LZW	41 s	62 s	**81 s**
Pre-processed and Proposed LZW	19 s	31 s	**49 s**

Table 5 shows that the time it takes for the compressed data to be sent is much less than the time required for sending the original files. For example, in the fourth column, the time it took for the original file to be sent (1,383) is much greater than the time it takes to send the compressed data with the proposed LZW algorithm (81). When preprocessing data and compressing it with the proposed algorithm, it notice the big difference in time (49) can be noticed.
Table 6 includes the throughput, which is the data size divided by the time it takes to send it.

Table 6: Throughput

No. hour	1 hour	2 hours	3 hours
Original Data	17.45 B/s	14.52 B/s	**19.36 B/s**
Compressed Data (Original LZW)	39.64 B/s	31.33B/s	**30.91 B/s**
Preprocessed and Compressed Data (Original LZW)	41.7 B/s	39.64 B/s	**34.29 B/s**
Compressed Data (Proposed LZW)	31.85 B/s	31.43 B/s	**30.79 B/s**
Preprocessed and Compressed Data (Proposed LZW)	42.84 B/s	39.64 B/s	**34.42 B/s**

Table 6 above shows the big difference in throughput that occurs when comparing the results. For example, in the first column, the throughput is low in the original uncompressed data (17.45 B/s) when compared with the compressed data with the original algorithm (39.64 B/s). The throughput increases as we go down. The throughput of the proposed algorithm is higher than the original algorithm (31.85 B/s) and when preprocessing and with the proposed algorithm, we also notice a big difference in throughput (42.84B/s).

Table 7: saving percentage

No. hour	1 hour	2 hours	3 hours
Compressed Data (Original LZW)	84.73%	85.26%	**90.42%**
Preprocessed and Compressed Data (Original LZW)	89.64%	90.82%	**93.47%**
Compressed Data (Proposed LZW)	83.77%	85.45%	**90.69%**
Preprocessed and Compressed Data (Proposed LZW)	89.89%	90.82%	**93.70%**

Through the clear results in Table 7, ti is noticeable that the last row, which is the proposed method, is considered the best results compared to the previous rows.

7- Conclusion & future work

In this research, weather data was collected from reality using sensors, which took the form of five stages. The first was to send data without compression and calculate the compression ratio and throughput for them; in the second, the data was compressed with the original algorithm, sent, and compared the results obtained; and in the third, a pre-processing of the data was conducted, compressing it with the original algorithm and also calculating the compression ratio and throughput. In the fourth, the proposed algorithm was compressed, which provides for dividing the data into different categories according to the number of bytes. In the fifth, compress the data that has been pre-processed with the proposed algorithm. Therefore, it is noticeable the difference that occurs when compressing with the proposed algorithm. The compression ratio and throughput increase significantly. We suggest Repeat the experiments with other conditions of location and degree as a future work.

8- References

[1] S. A. Abdulzahra, A. K. M. Al-Qurabat and A. K. Idrees, "Data Reduction Based on Compression Technique for Big Data in IoT,", in 2020 *International Conference on Emerging Smart Computing and Informatics* (ESCI), pp. 103-108, 2020.

[2] S. Pushpalatha, K.S. Shivaprakasha, Energy-efficient communication using data aggregation and data compression techniques in wireless sensor networks: A survey. In Advances in communication, signal processing, VLSI, and embedded systems (pp. 161-179). Springer, Singapore, 2020.

[3] S. R. Kodituwakku, U. S. Amarasinghe, "Comparison of lossless data compression algorithms for text data", *Indian journal of computer science and engineering*, vol. 1, no. 4, 416-425, 2010.

[4] Neha Sharma, Jasmeet Kaur, et al., "A Review on various Lossless Text Data Compression Techniques," *International Journal of Engineering Sciences*, vol 12, pp. 58-63, (December) 2014.

[5] Mamta Rani and Vikram Singh, "A Survey On Lossless Text Data Compression Techniques". *International Journal of Advanced Research in Computer Engineering & Technology* (IJARCET) vol. 5, Issue 6, June 20.

[6] Ganesh, S. Sankar and N Shubha G., "Energy Saving Techniques for Wireless Sensor Networks using Data Compression and Routing Protocols," *Imperial journal of interdisciplinary research*, 2016.

[7] S. Jancy, C. Jayakumar, "Packet level data compression techniques for wireless sensor networks," *Journal of Theoretical and Applied Information Technology*, vol.75. no.1, 2015.

[8] E.K. Gbashi, "Text Compression & Encryption Method Based on RNA and MTF," *Iraqi Journal of Science*, vol. 58, no. 2C, pp. 1149-58, 2022.

[9] Y. Liu, Q. Wu, et al., "An Improved Energy-Efficient Routing Protocol for Wireless Sensor Networks," *Sensors*, vol. 19, no. 20, pp. 4579, 2019 .

[10] Y. C. Wang, "*Data Compression Techniques in Wireless Sensor Networks," Pervasive Computing, Chapter 3, Nova Science Publishers*, 2012.

[11] W. Cui, "New LZW data compression algorithm and its FPGA implementation," In *Proc. 26th Picture Coding Symposium*, PCS 2007.

[12] Z. Yan-li, F. Xiao-ping, et al., "Improved LZW algorithm of lossless data compression for WSN," In 2010 *3rd International conference on computer science and information technology* vol. 4, pp. 523-527, IEEE, 2010.

[13] Upton, Eben, "Raspberry Pi 4 on sale now from $35," *Raspberry Pi Foundation*, June 2019.

[14] Raspberry Pi downloads". Retrieved 12 August 2016.

[15] L. Zheng, "ZigBee wireless sensor network in industrial applications," In 2006 SICE-ICASE *International Joint Conference* (pp. 1067-1070), IEEE, 2006.

[16] R. F. Abbas, "Images Compression Using Bezier curve with Ridgelet Transform," *Iraqi Journal of Science*, vol. 58, no. 3A, pp. 1290–1297, Dec. 2021.