Symmetric Segmented Delta Encoding for Wireless Sensor Data Compression

Shu-Ping Liang
Department of Computer Science and Engineering, Yuan Ze University, Taoyuan, Taiwan 320, R.O.C, s1011433@mail.yzu.edu.tw

Yi-Yu Liu
Department of Computer Science and Engineering, Yuan Ze University, Taoyuan, Taiwan 320, R.O.C, yyliu@saturn.yzu.edu.tw

Wireless sensor networks (WSNs) are utilized for various applications such as environmental monitoring, urban surveillance, home security, etc. Ideally, the massively deployed sensor nodes should be inexpensive, power-efficient, and reliable to maximize the functional lifetime. However, the data-transmission energy consumption has become one of the most challenging issues. To minimize the data-transmission energy cost in wireless applications, we propose a lossless Symmetric Segmented Delta encoding (SSD encoding) algorithm, which exploits high similarity of environmental sensing data in a short period of time. According to the experimental results, our encoding algorithm achieves better sensor data compression ratios and at the same time requires less hardware resource for wireless sensor data.

I. Introduction

With the rapid advancement of micro-electro-mechanical systems (MEMS), wireless sensor networks (WSNs) could be widely deployed in a cost-effective manner for various applications. A large number of sensor nodes play a major role in sensing, processing and communication. Among them, environmental monitoring, urban surveillance, and home security have drawn much attention, since those WSN applications are related to human lives. In most WSN scenarios, sensor nodes are deployed in locations that are not conveniently accessible. Therefore, these nodes are usually powered by batteries. Ideally, the massively deployed sensor nodes should be inexpensive, power-efficient, and reliable to maximize the functional lifetime, facilitate data collection, and minimize the cost for maintenance. However, the data-transmission energy consumption has become one of the most challenging issues. Consequently, many researchers focus on energy reductions for WSN applications.

The main cause of power consumption is wireless communication operation. The energy consumed for transmitting a single bit is comparable to that for processing thousands of commands [1]. Therefore, reducing the amount of sensor data by data compression is a useful solution to reduce data-transmission energy. However, the existed powerful data compression algorithms may not be very appropriate for sensor nodes owing to the hardware resource constraint. A balanced data compression algorithm for WSN application should take hardware cost, compression ratio, and overall energy consumption into account. In this paper, we first analyze the environmental sensor data characteristics and propose a lossless Symmetric Segmented Delta encoding (SSD encoding) algorithm by modifying the state-of-the-art Delta encoding algorithm with symmetric and segmented properties for WSN applications. The SSD encoding can be efficiently used to compress environmental sensing data in the sensor nodes. With the proposed SSD encoding technique, we are capable of not only enhancing the compression ratio but also conserving hardware resource.

This paper is organized as followss. The preliminary of power consumption in WSNs, state-of-the-art data compression algorithms, and environmental temperature data characteristics are in Section II. The proposed Symmetric Segmented Delta encoding (SSD encoding) is described in Section III. The experiment results and hardware resource estimation are in Section IV. We conclude this paper and points out important issues for future research in Section V.

II. Preliminary

In this section, we give preliminary background of power consumption in WSNs and data compression algorithms. Then, we focus on environmental monitoring applications in WSNs – the SensorScope project develops a large-scale distributed environmental measurement system in Switzerland. According to the environmental temperature datasets of this project, we analyze the temperature data characteristics.

A. Power Consumption in Wireless Sensor Networks

A wireless sensor network is composed of one or several remote sinks and a large number of sensor nodes [2]. Sensor nodes have to wirelessly transmit collected data to the sinks periodically. Each sensor node contains power unit, sensing unit, processor, storage, and transceiver. To massively deploy the sensor nodes, sensor nodes are expected to have small form factor, reasonable manufacture cost, and long lifetime. Hence, the design and implementation of WSNs are constrained by hardware resource in terms of battery, transmission bandwidth, computation capability, and memory size.

Since battery replacement and sensor re-deployment are prohibitively expensive in common practice, energy saving is one of the most important aspect in sensor node design. The

energy consumption in sensor nodes can be divided into three parts: data collecting, data processing, and data transmission. Among those three operations, data transmission consumes approximately 80% overall energy [3]. In order to save transmission energy, we should minimize the volume of transmitted data by compression. With data compression technique, we can improve the lifetime of sensor nodes by reducing the transmission data volume, the required storage size, and the hardware cost simultaneously.

However, data compression before data transmission consumes more energy as compared to the original raw data transmission. Hence, the energy consumption of data compression must be smaller than the energy saving from data transmission. That is, assume that there are n bits original data string; there are m bits data string after data compression, where $n > m$; if c is the energy consumed by compressing/decompressing and w is the energy consumed by data transmission per bit, the compression algorithm is feasible when $c < w \times (n - m)$.

B. Data Compression Algorithms

Data compression algorithms can be classified into two categories, lossless and lossy, depending on whether the content can be perfectly restored or not during compression. Both lossless and lossy compressions can be found in current WSNs. In this work, we target the lossless compression for users who want to have the raw data intact. Thus, we discuss the existed works on lossless data compression methods, such as LZW, Huffman encoding, and Delta encoding [4].

Lempel-Ziv-Welch (LZW) algorithm sets same length codewords to variable length series of source sequences. LZW dynamically builds the dictionary that maps symbol sequences to an *N*-bit index. The dictionary has 2^N entries and the codeword is an index into the dictionary to retrieve the corresponding original sequences. LZW reads a new symbol and concatenates it to get a new subsequence that is added in the dictionary based on previous sequences. When LZW revisits a subsequence, this subsequence will be encoded using an index. Usually, the user defines the maximum number of dictionary entries, so that the process doesn't run away with memory. Based on LZW, prof. Sadler develops S-LZW [5] for resource-constrained wireless sensor nodes. This work finds that sensing data may be repetitive at short intervals, so it also proposes a S-LZW-MC with a mini-cache into the dictionary.

Huffman encoding first calculates the occurrence frequency of each symbol and sorts the symbols. A binary Huffman tree is constructed by iteratively merging two symbols with lowest frequencies. Once the Huffman tree is constructed, a unique path from root to leaf is encoded as the binary codeword for a symbol. Symbols with high occurrence frequency will near the root of the tree and be represented by short codewords. Prof. Jambek [6] analyses the compression performance of the Huffman and LZW algorithms. From the experimental results, the Huffman algorithm is better than LZW for the common data using in WSNs.

Delta encoding is fairly straightforward and the codeword size has two types. The Greek letter delta (Δ) implies difference or change in mathematics or science. The first value in the input stream is always written out the same as the original value. After that, every value is replaced with an *S* bits signed binary value representing the difference between the previous value in the input stream. The codeword is *S* bits long to allow for differences ranging from -2^{S-1} to $2^{S-1}-1$. Another type of the codeword is the original current value, when a delta is too large to be represented by *S* bits. So, the compression performance of Delta encoding depends on *S*. Keeping *S* small may get higher compression ratio, but if *S* is too small, the delta may be out of the range of the *S* bits signed binary value.

C. Environmental Temperature Data Information

SensorScope project develops a low-cost and reliable system, based on WSN for environmental monitoring in Switzerland. The sensor node is Shockfish TinyNode for environmental data collection [7]. The sensor is equipped with a 16-bit microcontroller, running at 8MHz, and a radio transceiver, operating in the 868 MHz band, with a transmission rate of 76Kbps. It has 48KB ROM, 10KB RAM, and 512KB flash memory. The data packet records the station ID, time, ambient temperature, relative humidity, rain meter, and so on. In this paper, we focus on temperature data analysis.

According to the temperature data from SensorScope project [8], we notice that *temperature differences between adjacent values are small* and the *temperature distribution is uniform*. Fig. 1 shows the dataset of temperatures from pdg2008-meteo-5. Fig. 2 shows the frequency distribution of temperature differences from dataset pdg2008-meteo-5.

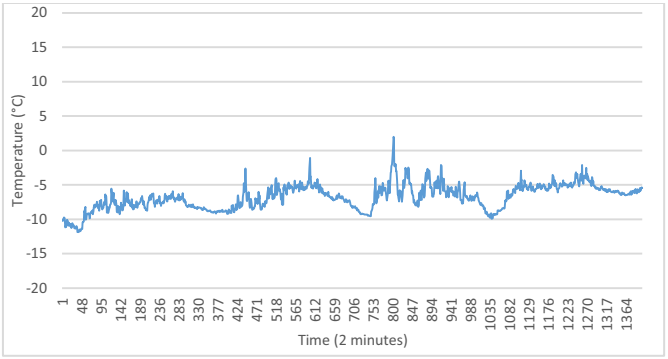


Fig. 1. The pdg2008-meteo-5 temperatures.

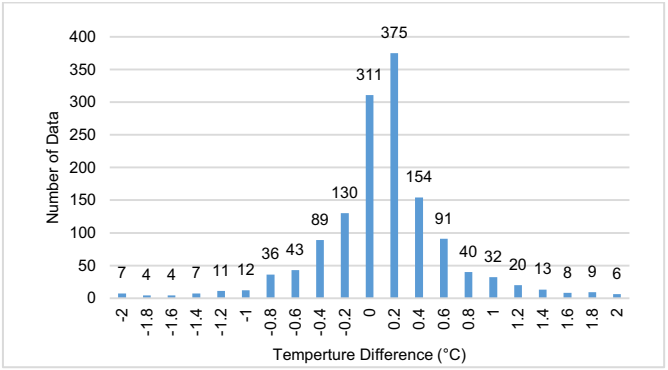


Fig. 2. The temperatures difference frequency distribution.

Based on the two observations, we believe that Delta encoding is a good candidate for temperature data compression. Let T be the *current* temperature, T' be the previous temperature and ΔT be the difference of the temperature data.

$$\Delta T = |T - T'|$$

TABLE I lists the mean $\overline{\Delta T}$ and standard deviation $\sigma(\Delta T)$ of temperature deltas from different sensor stations. We can see that the standard deviation of pdg2008-meteo-5 is the largest beyond the other datasets. That is, the temperature deltas of pdg2008-meteo-5 are ranged in a wider distribution. In contrast to pdg2008-meteo-5, the temperature deltas of pdg2008-meteo-15 are ranged in a narrow distribution. To improve the compression ratios of various temperature datasets, it is indeed difficult to select just one delta codeword to fit different temperature distributions. In order to improve Delta encoding, we propose a new algorithm for data compression, Symmetric Segmented Delta encoding (SSD encoding).

TABLE I
The Mean and Standard Deviation of pdg2008-x Deltas

Station ID	1	5	10	15	16
Mean	0.3490	0.3543	0.1582	0.1586	0.2436
Standard deviation	0.2584	0.4084	0.2728	0.1282	0.2740

III. Symmetric Segmented Delta Encoding

The main idea of Symmetric Segmented Delta encoding (SSD encoding) algorithm is similar to Huffman encoding, which utilizes fewer bits to represent data with higher occurrence frequencies. That is, the range of temperature delta is partitioned into several segments. The temperature delta segment with high occurrence frequency will be assigned short codes.

We first calculate differences by subtracting the current value *T* from the previous value *T'*. Then, *delta* is the absolute value of the difference. The distribution of deltas is segmented into four continuous segments. Each segment has a *base* field and an *offset* field. For *delta* beyond the *base* of fourth segment, we keep the original temperature data unchanged. That is, the *offset* field stores the absolute value of the current temperature *T*. If the *delta* is less than the *base* of the fourth segment, denoted as $base_4$, we use *offset* field to store the difference between *delta* and *bases* as follows.

If $delta < base_4$, the delta is in the first three segments and satisfy the given rules:

$$\begin{cases} delta = |T - T'| \\ offset = delta - base_i \\ 0 \le offset < 2^{offset\ bits_i} \end{cases}, i = 1, 2, 3$$

Otherwise,

$$\begin{cases} offset = |T| \\ 0 \le offset < 2^{offset\ bit_i} \end{cases}, i = 4$$

The code format is divided into three fields as shown in Fig. 3.

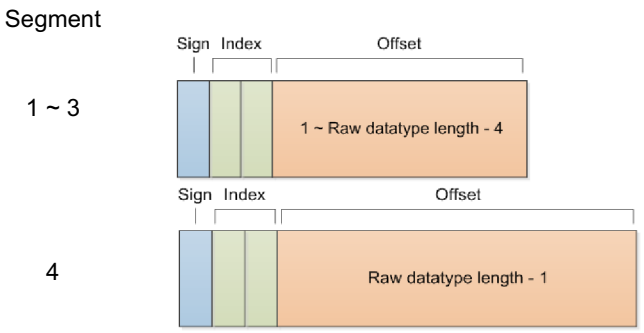


Fig. 3. The format of Symmetric Segmented Delta encoding.

The index field determines which segment a delta belongs to. When the delta is in the first three segments, the offset filed identifies offsets of the delta within that segment and the sign bit indicates the sign of the delta. In contrast to above segments, the *offset* field of the fourth segment represents the absolute value of the raw data and the sign bit indicates the sign of the raw data.

Figures 4 and 5 illustrate the flow charts of the SSD encoding compression and decompression, respectively. *T* is the current temperature; *T'* is the previous *T*; *C* is the code; *i* is the index of the segment; B_i is the base of the *i*-th segment; *O* is the offset. When the first data read in, *T'* is 0. Next, the second data read in, *T'* is the first data. The third data read in, *T'* is the second data, and so on. Exit on the end of data transmission. The compression and decompression algorithm both have time complexity *O*(*N*).

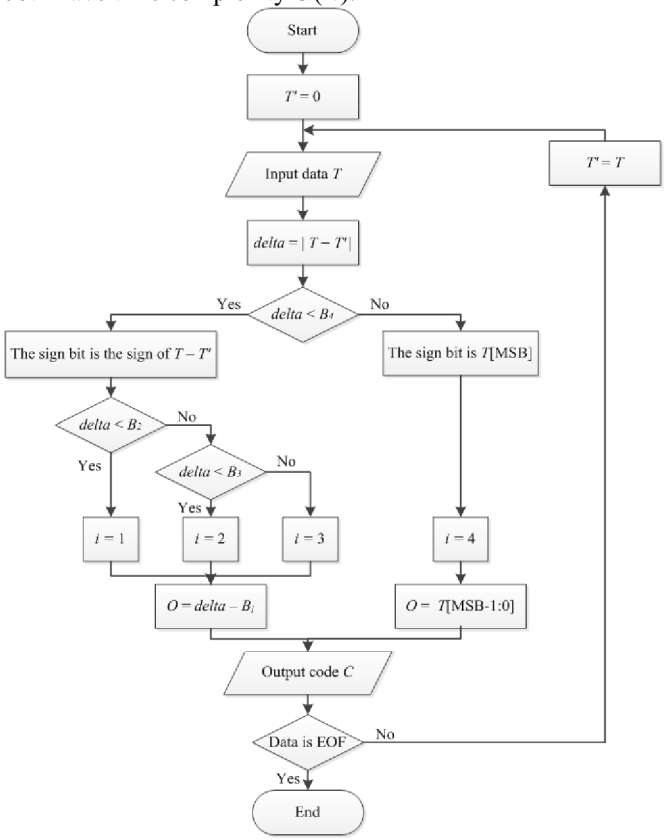


Fig. 4. Flow chart of SSD encoding (compression).

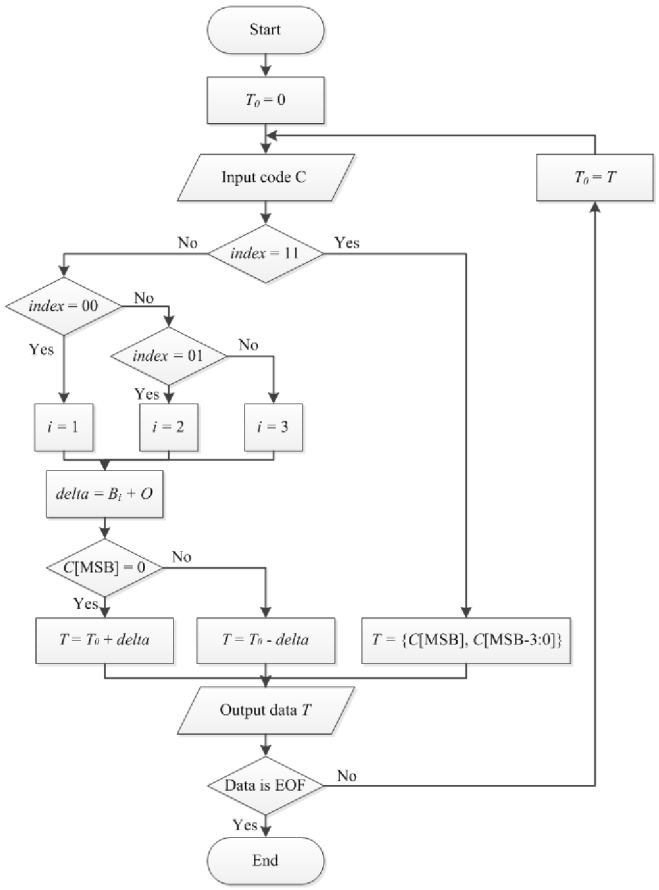


Fig. 5. Flow chart of SSD decoding (decompression).

According to the SensorScope historical temperature datasets, the values are in the range -20 to 20°C and have a precision of 0.01. The raw datatype is defined as a 12-bit signed integer. The maximum offset field length of the first three segments must be less than 9-bit (12-1-2=9). As a result, the first three segments can contain delta values in the range 0 to 2.56 ($2^8 \times 0.01$). We then use a brute-force approach to find the optimal segment combination with the best compression ratio by adjusting the *offset* field length of the first three segments. Take pdg2008-meteo-5 as an example. The first segment uses a 4-bit *offset*, and the $base_1$ is 0. The second segment uses a 5-bit *offset*, and the $base_2$ is 16 ($0 + 2^4$). The third segment uses a 6-bit *offset*, and the $base_3$ is 48 ($16 + 2^6$). The fourth segment uses an 11-bit *offset* to store the absolution temperature values, and the $base_4$ is 112 ($48 + 2^6$). Figures 6 and 7 list step-by-step compression and decompression for SSD encoding and decoding, respectively.

Input	Process		Code		
T	$T - T'$	$delta$ - $base_i$ = $offset$	Sign	Index	Offset
-10.08	-10.08	1008 – 0 = 1008	1	11	01111110000
-10.26	-0.18	18 – 16 = 2	1	01	00010
-10.10	0.16	16 – 16 = 0	0	01	00000
-10.00	0.1	10 – 0 = 10	0	00	1010
-9.72	0.28	28 – 16 = 12	0	01	01100

Fig. 6. Step-by-step compression.

Code			Process	Output
Sign	Index	Offset	$base_i$ + $offset$ = $delta$	T
1	11	01111110000	—	-10.08
1	01	00010	16 + 2 = 18	-10.26
0	01	00000	0 +16 = 16	-10.10
0	00	1010	0 +10 = 10	-10.00
0	01	01100	16 + 12 = 28	-9.72

Fig. 7. Step-by-Step decompression.

TABLE II shows the compression results for pdg2008-meteo-5. Although the fourth segment increases 0.95% data volume, this segment only contains 6% of all data. The advantages of saving more bits in the other segments after compression outweigh this disadvantage. For example, the first segment contains 40% of all data, and saves 16.66% data volume. Unlike conventional Delta encoding, which represents the deltas by a fixed offset field, our SSD encoding dynamically represents the deltas by different length bits according to the occurrence frequency. Therefore, SSD encoding increases the flexibility of compressing delta and improves the compression performance.

TABLE II
The Compression Results for Pdg2008-meteo-5

Segment	Offset	Quantity	Compressed size (Bits)	Compression ratio (%)
1	4-bit	561 (40.0%)	3,927	16.66
2	5-bit	493 (35.1%)	3,944	11.71
3	6-bit	269 (19.2%)	2,421	4.79
4	11-bit	80 (5.7%)	1,120	-0.95
Overall		1,403	11,412	32.21

IV. Experimental Results

To implement the SSD algorithm and to compare with other state-of-the-art compression techniques, LZW, Huffman encoding and Delta encoding, we calculate compression ratios for different datasets and estimate the required hardware resource for compression.

We use distinct regional temperature datasets with various weather conditions. SensorScope datasets [8] have a precision of 0.01, so the raw datatype is a 12-bit signed integer. Daily temperature for one year in Taipei [9], Tokyo [10] and New York [11] have precision of 0.1, so the raw datatype is defined as a 10-bit signed integer. Datasets for the experiment are listed in TABLE III.

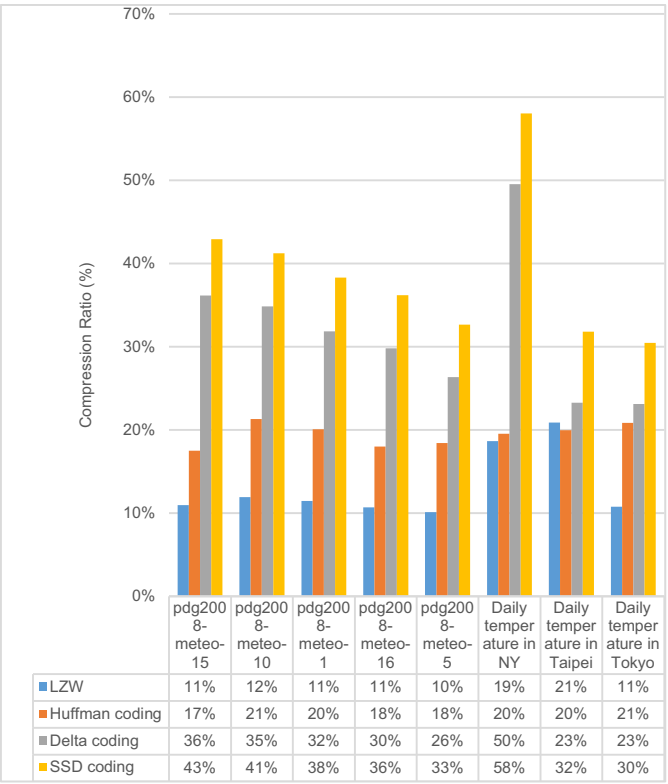
The performance of a compressed algorithm can be defined by compression ratio as shown below:

$$\text{Compression Ratio} = \left[\frac{(Uncompressed\ Size - Compressed\ Size)}{Uncompressed\ Size}\right] \times 100\%$$

Fig. 8 shows the compression results for the above data compressed using LZW, Huffman encoding, Delta encoding

and SSD encoding. From Fig. 8, LZW performs poorly for temperature because LZW compresses data by using the dictionary that is inefficient for numerical data. Huffman encoding performs better than LZW. The arithmetic average compression ratio is 20% due to highly repetitive values, which are suitable for Huffman encoding.

As compared to LZW and Huffman encoding, Delta encoding gives higher compression results. This is because Delta encoding analyzes and models numerical data before compressing data. For compressing deltas, the compression ratios of SSD encoding are approximately 7% more than Delta encoding. The highest compression ratio is 58% for daily temperature in New York. Unlike Delta encoding, SSD encoding is not limited by the fixed length representation of deltas. Therefore, SSD encoding improves Delta encoding by the use of symmetric segment method for deltas.



	pdg2008-meteo-15	pdg2008-meteo-10	pdg2008-meteo-1	pdg2008-meteo-16	pdg2008-meteo-5	Daily temperature in NY	Daily temperature in Taipei	Daily temperature in Tokyo
LZW	11%	12%	11%	11%	10%	19%	21%	11%
Huffman coding	17%	21%	20%	18%	18%	20%	20%	21%
Delta coding	36%	35%	32%	30%	26%	50%	23%	23%
SSD coding	43%	41%	38%	36%	33%	58%	32%	30%

Fig. 8. Compression ratio comparisons.

TABLE IV compares required hardware resource of LZW, Huffman encoding, Delta encoding and SSD encoding. The width of data sample is 12bits. The LZW dictionary storage capacity is 6K. As a result, the resource utilization and power consumption of memory is much higher than the others. Huffman encoding use 4-bit buffer, and the data sample is divided into 3 parts. The Huffman tree stores codes and the length of each code. Huffman encoding compresses data by accessing a symbol value in a Huffman table, so comparators and adders are not used, which is very energy-efficient.

Delta and SSD encodings perform addition and subtraction operations for input data. The memory requirement to store the base values is much less than LZW and Huffman encoding. The SSD encoding utilizes more memory and performs 2 more arithmetic operations as compared to the Delta encoding.

TABLE III
Datasets for Compression

Name		File Information		
Dataset	Region	Quantity	Spec	File Size
pdg2008-meteo (Precision is 0.01)	1	3,665	12 bits	43,980
	5	1,403	12 bits	16,836
	10	3,657	12 bits	43,884
	15	4,662	12 bits	55,944
	16	3,072	12 bits	36,864
Daily temperature for one year (Precision is 0.1)	Taipei	365	10 bits	3,650
	Tokyo	365	10 bits	3,650
	NY	365	10 bits	3,650

TABLE IV
Hardware Resources

Algorithm / Sources	LZW	Huffman encoding	Delta encoding	SSD encoding
memory	2048 * 24 bits	16 *19 bits	2 * 12 bits	4 * 12 bits
comparator	12 bits	–	12 bits	12 bits
Maximum number of comparisons	2048	–	1	3
adder	–	–	12 bits	12 bits
buffer	12 bits	4 bits	12 bits	12 bits

V. Summary and Conclusions

We have proposed a lossless data encoding algorithm for sensor nodes to optimize data-transmission energy in WSN. We also make a comparison with other typical data compression techniques for WSN applications. The results indicate that our approach achieves better compression performance. The proposed Symmetric Segmented Delta encoding (SSD encoding) technique alleviates the problem of Delta encoding by making a simple distinction of deltas. Taking sensor data compression hardware cost, compression ratio, and overall energy consumption into account, we would suggest SSD encoding algorithm for environmental sensing data. We are now working on the heuristics to determine the size of offset fields to avoid brute-force computations.

References

[1] Pottie, G. and Kaiser, W., “Wireless Integrated Net- work Sensors,” Communications of the ACM, Vol. 43, pp. 51-58 (2000).

[2] Ian F.Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, Erdal Cayirci, “A Survey on Sensor Networks,” *IEEE Communications Magazine*, pp. 102-114, August 2002.

[3] Naoto Kimura, Shahram Latifi, "A Survey on Data Compression in Wireless Sensor Networks," *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, Vol. 2, pp. 1-6, April 2005.

[4] Khalid Sayood, *Introduction to Data Compression* (4th ed.), Newnes, 2012.

[5] Christopher M. Sadler, Margaret Martonosi, "Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks," *in Proceeding of the 4th International Conference on Embedded Network Sensor Systems*, pp. 265-278, 2006.

[6] Asral Bahari Jambek, Nor Alina Khairi, "Performance Compression of Huffman and Lempel-Ziv Welch Data Compression for Wireless Sensor Node Application," *American Journal of Applied Science 11 (1)*, pp. 119-126, 2014.

[7] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli and O. Couach et al., "SensorScope: Application-Specific Sensor Network for Environmental Monitoring," *ACM Transactions on Sensor Networks*, vol. 6, num. 2, pp. 1-32, 2010.

[8] Sensorscope: Downloads, http://lcav.epfl.ch/cms/lang/en/pid/86035.

[9] CWB Observation Data Inquire System, http://e-service.cwb.gov.tw/HistoryDataQuery/index.jsp

[10] Japan Meteorological Agency Data Search, http://www.data.jma.go.jp/obd/stats/etrn/index.php

[11] National Oceanic and Atmospheric Administration Data Tools: 1981-2010 Normals, http://www.ncdc.noaa.gov/cdo-web/datatools/normals