# RESOURCE-EFFICIENT FLOATING-POINT DATA COMPRESSION USING MAS IN WSN

Maher El Assi[1], Alia Ghaddar[2], Samar Tawbi[3], Ghaddar Fadi[4]

[1, 2, 3] Lebanese University, Lebanon
[4] Saint Joseph University, Lebanon

***ABSTRACT***

*In a wide range of applications, large amounts of floating-point data are generated by Wireless Sensor Networks (WSNs). This data is often transferred between several sensor nodes, in a multi-hop fashion, before reaching its ultimate destination (the base station). It is well known that data communications is the most energy-consuming task in sensor nodes [1]. This can be a great concern when the nodes are constrained in energy. Therefore, the amount of data to be transferred between nodes should be reduced to save energy. In this paper, we investigate data compression for resource-constraint WSNs; we introduce MAS as a novel adaptive lossless floating-point data compression algorithm for WSNs. MAS exploits the disproportionality in energy consumption between data transmission and processing. Simulation results, obtained from OMNeT++ and Atmel Studio, show that MAS surpasses other tested compression algorithms in terms of compression ratio, compression speed, memory requirements and most importantly energy savings.*

***KEYWORDS***

*Wireless Sensor Networks, Lossless Compression, Floating-point Data, Energy Efficiency*

## 1. INTRODUCTION

In the last few decades, Wireless Sensor Networks (WSNs) has proven to be an interest grabbing technology, offering great contributions in several application domains. Wireless Sensor Networks can provide a low cost solution to a variety of real-world problems including but not limited to health care, industry process control, object tracking, volcanic and seismic monitoring, smart parking, home automation, etc. Moreover, WSNs can provide enhanced situation awareness in responding to today's public safety situations. For example, Sleep Safe project is designed for monitoring infants while they sleep. Sleep Safe sensor nodes can prevent sudden infant death syndrome (SIDS) by autonomously detecting the sleeping position of an infant and alerting the parents wirelessly in real time when the infant is lying on its stomach [2].

Typical WSNs are composed of a relatively high number of sensor nodes communicating through an infrastructure-less multi-hop wireless network architecture. These nodes usually perform three main tasks: data collection, data processing and data communication. The nodes capture data from their surrounding environment, they process it and finally they transfer it to a base station where decision makers can make use of it.

Sensor nodes are small, cheap and smart devices that are made up of four basic components [3], as shown in Figure 1:

DOI : 10.5121/ijasuc.2013.4502 

1) A sensing unit which captures a physical quantity from the environment.
2) A processing unit which processes and analyzes the captured data.
3) A transceiver which is responsible for data communication.
4) A power unit which is in most of the cases a battery.

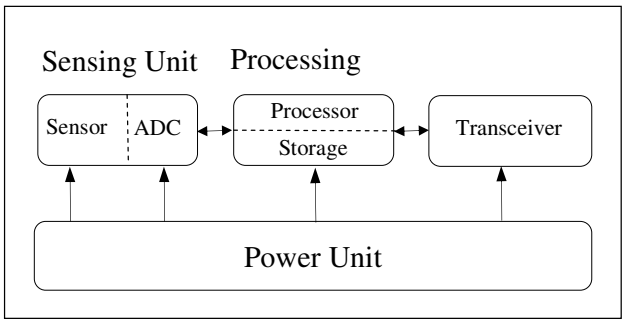


Figure 1. Main Components of a sensor node. [3]

Despite their promising range of applications, most WSNs are constrained in resources; they have limited amount of energy, limited processing capabilities, short range of communication and limited memory size. Out of these constraints, energy is considered the primary concern especially for battery-operated sensors; this is true because when a sensor node is depleted of energy, it would be useless for the network. This could affect the performance of the whole network especially if the node is used in critical locations, such as mines, volcanoes, etc.

Over the past years, different studies and techniques have been proposed for WSNs to reduce energy consumption and increase network performance and lifetime. Data compression has been adopted as a practical technique and reliable solution in terms of energy efficiency in WSNs. The efficiency of data compression techniques mainly bears on the drastically disproportionate energy cost between data transmission and processing. This can be seen in Figure 2, which shows how many compute cycles, on a Texas Instruments MSP430 microcontroller, would be performed for the same amount of energy required to transmit a single byte over three commonly used radios [7] (Chipcon CC2420 [4]: short range 125 m, the Chipcon CC1000 [5]: medium range 300 m, and MaxStream XTend [6]: long range 15 km).

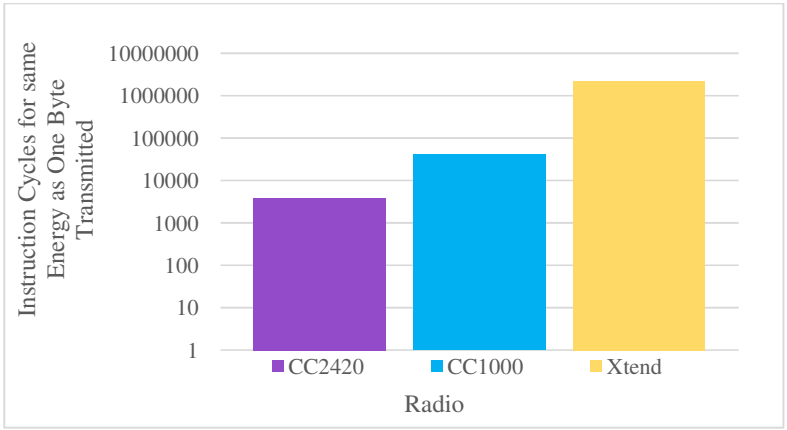


Figure 2. Number of TI MSP430F1611 compute cycles that can be performed for the same amount of energy as transmitting one byte over three radios. [7]

It is obvious that the most energy consuming part in WSNs is the communication. Approximately 80% of power consumed in each sensor node is used for data transmission [1]. Thus to save

energy and maximize network lifetime, data transmissions should be minimized without losing vital information. The lower the size of the transmitted data the lower the number of required transmissions. In our work, we study data compression as the technique for minimizing data size.
In this paper, we propose an energy efficient floating-point data compression algorithm for WSNs called MAS. MAS is a new adaptive streaming lossless compression algorithm that relies on an accommodative coding technique to achieve compression at low processing costs. MAS offers great contributions to WSNs, because it is one of the first algorithms to specifically compress floating-point data. By focusing on floating-point data, it is possible to achieve much better compression ratios because we can exploit the characteristics and the nature of numbers to build our algorithm. In fact, floating-point data is generated in a wide range of applications such as weather monitoring (temperature, pressure…), healthcare (blood pressure, cardiac activity…), localization and tracking (position, height, coordinates…), industry (temperature, vibrations, radioactivity…), etc.

The remainder of this paper is organized as follows: Section 2 gives an overview of common compression schemes specifically designed for WSNs. Section 3 presents our newly proposed algorithm. Section 4 presents an evaluation of the presented algorithms. Section 5 concludes this paper while Section 6 presents future works.

## 2. RELATED WORK

Energy efficiency has been a major concern in the design and development of WSNs. Since radio communication is known to be the main source of energy consumption, most of the proposed techniques in the literature, which aim to increase energy savings, have focused on reducing data communication (transmission/reception). Data compression is such a technique, which is often used in conjunction with data collection techniques to transmit the collected data in an energy efficient manner.

Due to the distributed nature of WSN applications, and the resource-constrained nature of sensor nodes, traditional data compression techniques cannot be easily used. It may not be feasible to run sophisticated data compression algorithms on sensor nodes. The limited resources available in these nodes demand the development of specifically designed algorithms. In this section, we present two famous compression algorithms used in WSNs: SLZW [7] and K-RLE [8]. We also present our proposed compression algorithm MAS [9] and compare it with these algorithms.

### 2.1. S-LZW

S-LZW [7] (Sensor-LZW) is an adaptation of the popular lossless data compression algorithm LZW [10]. S-LZW follows the same procedure used by the LZW algorithm, but with little restrictions regarding the size of the used data structures. The added restrictions ensure that the requirements of the algorithm are still within the bounds of the available resources in sensor nodes.

Before heading into the details of the modification, it is first important to understand why LZW is not suitable for WSN. LZW is a dictionary-based compression algorithm; it works by converting strings of symbols into integer codes. LZW does not use a static dictionary; instead, it builds the dictionary on the fly in a special way to allow both the encoder and the decoder to be able to generate the same dictionary from the input data. First, both the encoder and the decoder initialize the dictionary with 256 entries containing the symbols in the ASCII code. Then the dictionary continues to grow while parsing the input.

The dictionary is the main obstacle preventing LZW from being applicable to WSN. Throughout the compression mechanism, the dictionary keeps growing and can reach sizes much higher than the available RAM on sensor nodes, and this can clearly disrupt the stability of the system. Another problem that LZW faces is that it requires a predefined data volume, i.e. in order for it to start the compression procedure a significant amount of data must already be available. That is why S-LZW can only be used in delay tolerant networks.

Several modifications were done on LZW to make it portable to WSNs. Most of these modifications focus on reducing the amount of RAM required for LZW to operate. Here is a list of modifications that gave the birth to S-LZW:

- S-LZW uses a 512-entry dictionary. As we mentioned before, this dictionary will be initialized with 256 ASCII code symbols. With this size, the dictionary may get full while compressing or decompressing certain datasets. There are two protocols to follow when the dictionary fills, either fix the dictionary to its state whence it get full, or reset the dictionary to the 256 entries. The authors of S-LZW [7] proved that using the fixed protocol produces better results when compressing data of small block sizes (528 bytes).

- S-LZW divides input data into block sizes of 528 bytes, and then it compresses these blocks individually. It is important to note that S-LZW requires 528 bytes of data to be available in order to compress it, if this amount of data is not available, it has to wait until data accumulate and reach the required size because compressing data of smaller size will be inefficient as shown in [7]. WSN data sampling rate is relatively low and it may take some time to collect 528 bytes of data to be able to start the compression. This is why this algorithm can only be used in delay tolerant networks.

- The last modification enhances S-LZW by allowing it to benefit from the similarity of data generated by sensor nodes. This is done by adding a mini-cache, which is a hash-indexed dictionary of size N, where N is a power of two, which stores recently used and created dictionary entries. The authors show that it is best to use mini-caches of sizes 32 or 64 dictionary entries.

## 2.2. K-RLE

K-RLE [8] is a new compression algorithm whose idea is inspired from the lossless data compression algorithm RLE [11]. RLE stands for Run-Length Encoding, which is a very basic and simple compression algorithm that works in this way: if a data item *d* occurs *n* consecutive times in the input stream, we replace the *n* occurrences with a single pair *nd*.

RLE itself is very simple and can be used in WSN without any major changes, its RAM and processing requirements are very low. However, there is a major limiting constraint in RLE, for RLE to achieve good compression ratio, the input data must contain long sequences of repeated characters, and this rarely occurs in the data generated from sensors. To solve this problem, K-RLE algorithm has been proposed; K-RLE means RLE with K precision.

The idea behind this algorithm is: let *K* be a number, if a data item *d*, *d*+*K*, or *d*-*K* occur *n* consecutive times in the input stream, we replace the *n* occurrences with a single pair nd [8]. The new addition in K-RLE allows it to achieve higher compression ratios than RLE but at an even cost, which is data loss. In contrary to RLE, K-RLE is a lossy compression algorithm, and the amount of data loss is strongly related to the K parameter. Higher values for K means better compression ratio but more data loss, while lower values of K means lower compression ratio but less data loss.

There are two main advantages of K-RLE over S-LZW:

1) K-RLE uses much less amount of RAM than S-LZW, so it can be used in several sensor platforms where S-LZW cannot be used.
2) K-RLE has the streaming feature, which means it does not need to buffer data before being able to start the compression process. So K-RLE can be used in networks that cannot tolerate delay.

The main two disadvantages of K-RLE are:

1) It is a lossy algorithm, so it is not suitable for some applications
2) It requires the input data to contain long sequences of similar characters in order to have a good compression ratio.

## 3. MAS COMPRESSION ALGORITHM

MAS stands for Minimalist, Adaptive and Streaming compression algorithm. Minimalist means that it uses the minimum possible amount of resources. Adaptive means it generates variable-size output according to the number of digits in the input, and Streaming means that it does not require buffering of the data before starting the compression process.

MAS is a specialized lossless compression algorithm that only compresses single-precision floating-point data. MAS' implementation does not require any correlation or similarity in the input data, which makes it general and applicable in various domains.

MAS can encode any floating-point number satisfying the following two conditions:

1) The number of significant digits should be at most 7, if a floating-point number exceeds 7 significant digits it would be truncated.
2) The floating-point number when put in scientific notation must have a power of 32 or less.

Although these conditions mean that some numbers representable by IEEE standard 754 [12] will not be representable in MAS. However, these numbers almost do not exist in the data generated by WSN. Numbers that have decimal powers of more than ±32 are almost not found in any application in WSN.

One of the greatest merits of MAS is that it does not require any floating-point operation (addition, subtraction, multiplication, division) to compress floating-point numbers. This is very important because most microcontrollers and processors in sensor platforms are not equipped with an FPU (Floating point unit). The FPU is responsible for carrying out operations on floating-point numbers. In the absence of an FPU, these operations are emulated in software but at the cost of time and cycles, which could lead to higher energy consumption. Internally, MAS treats floating-point numbers as strings of characters to carry out the needed operations with a low number of cycles.

### 3.1. MAS Encoding Technique

The first step in encoding a floating-point number is to write it in scientific notation. Scientific notation allows the representation of very small or big numbers with ease. So any number is first written under the following format (d = digit, e = exponent):

$$\pm d.ddddd\ x\ 10^{\pm e}$$

To encode a number, the different parts in the above format must be encoded. MAS encodes them in five sections detailed below and are shown in Figure 3 from left to right:

- Number of significant digits (*n*) (number of *d's*): represented on 3 bits because its maximum value is 7.
- The exponent (*e*): represented on 5 bits because its maximum value is 32.
- Number sign (*ns*): represented on 1 bit. (0 for positive, 1 for negative).
- Exponent sign (*es*): represented on 1 bit. (0 for positive, 1 for negative).
- The integer formed by the *d's* without the decimal point (*ddddddd*): variable bit size depending on the number of *d's* (maximum 24 bits). Details are in Table 1.

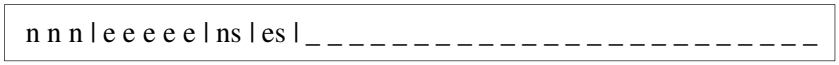


Figure 3. MAS encoding of a real number.

Table 1.Number of bits needed to represent an integer in binary formed by a certain number of digits.

| Number of digits | Number of needed bits |
|---|---|
| 1 | 4 |
| 2 | 7 |
| 3 | 10 |
| 4 | 14 |
| 5 | 17 |
| 6 | 20 |
| 7 | 24 |

It is clear that MAS exploits the significant number of digits to represent floating point numbers, for example: -0.0001 is represented on 14 bits while 92301.1 is represented on 30 bits.

Integers form a large part of real numbers, but the above representation may be unfair for integers because there is no need for the exponent and its sign when representing integers. Therefore, a special encoding has been chosen for integers, but there should be a discriminator for the decoder to know which type of number it is going to decode. We exploit the fact that the number of significant digits cannot be zero, and use this as a discriminator between the 2 encodings. Integer representation thus has four parts as shown in Figure 4:

- 3 bits that are all zeroes acting as a discriminator.
- Number of significant digits (*n*) in the integer: 3 bits.
- Number sign (*ns*): 1 bit.
- The integer: variable bit size following Table 1 (maximum 24 bits).

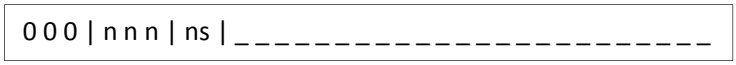


Figure 4. MAS encoding of an integer.

We also exploit the fact that in an integer the number of significant digits cannot be zero to make a special representation for the actual zero. A zero is represented in MAS as 6 zero bits: 000000.

Finally, since our algorithm generates codes of variable size, we need to make sure that the overall output is aligned to memory; we do this by filling the required number of bits by 1s. The

number of alignment bits is between 1 and 7, and they are added only once at the end of the output.

### 3.2. MAS Decoding Technique

The decoding technique is straightforward and is done following these steps:

1) Read the first three bits.

2) If the bits are zeroes, the encoded number is an integer.

    a. Read the next three bits to extract the number of significant digits.
    b. Read the next bit to determine the number's sign.
    c. Read the number of bits required to extract the number referring to Table 1, then extract the integer.
    d. Combine the readings so far to decode the integer.

3) If the bits are non-zeros, the encoded number is a real.

    a. The already read bits represent the number of significant digits.
    b. Read the next five bits to extract the exponent.
    c. Read the next bit to determine the number's sign.
    d. Read the next bit to determine the exponent's sign.
    e. Read the number of bits required to extract the number referring to Table 1, then extract the integer.
    f. Combine the readings so far to decode the real number.

The explanation of MAS' encoding and decoding techniques clarifies why MAS is considered a streaming algorithm. MAS can compress or decompress even one single value, and does not require predefined data volume as S-LZW.

## 4. EVALUATION

The evaluation metrics of any compression algorithm for WSN are based on the resource limitations of sensor nodes. Thus, we chose the following metrics to evaluate the presented algorithms: compression ratio, processing cost, memory requirements, and energy savings.

In the following sub-sections, we will start by describing the chosen platform and the chosen simulators, and then we will move to present the simulation results and evaluate the algorithms performance.

### 4.1. Platform Overview

Our chosen platform is the Waspmote [13] because of its interesting characteristics. A Waspmote can be connected to 60 sensor types and can support up to 8 different wireless technologies.
A Waspmote uses an 8-bit AVR microcontroller called ATmega1281 [14], which is low power microcontroller provided by Atmel [15]. It is supplied with a 128 KB flash (program) memory and an 8 KB RAM. The microcontroller can have frequencies between 0 and 16 MHz at an operating voltage of 1.8/5.5 V. In the active mode, as shown in Figure 5, the microcontroller consumes 1 mA at a voltage of 3.3 V and frequency of 1 MHz; this means this microcontroller consumes 3.3 nJ for one computation cycle in its active mode. This microcontroller does not have an FPU unit.

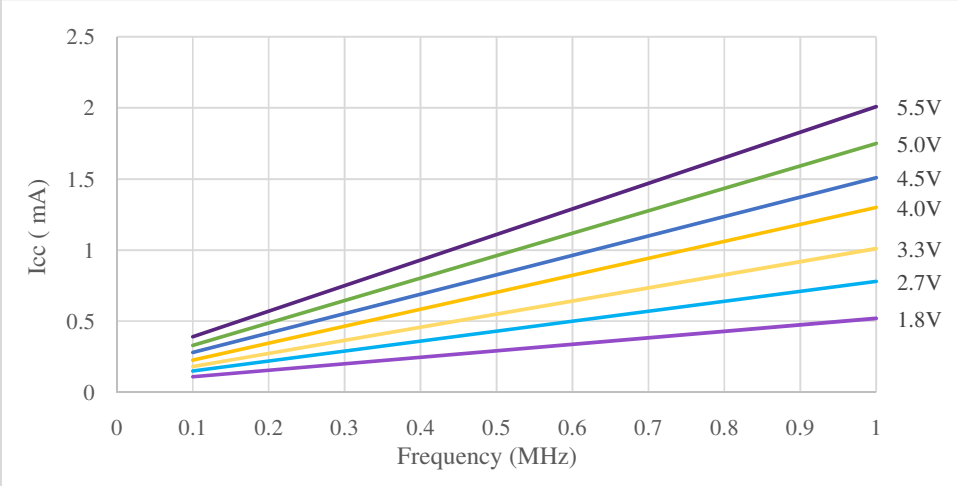


Figure 5. Active Supply Current vs. frequency (0.1MHz - 1.0MHz) on ATmega1281 microcontroller. [14]

We chose the wireless technology to be CC2420 RF [16] transceiver, which complies with the IEEE 802.15.4 standard [17]. This transceiver is designed for low-power and low-voltage wireless applications. It has low current consumption; for transmission, it consumes 17.4 mA and for reception, it consumes 18.8 mA. It has an effective data rate of 250 kbps.

### 4.2. Simulators Overview

In order to calculate the total energy consumed, we need to calculate two different kinds of energy, the computation energy, and the communication energy. To calculate the computation energy, we calculate the number of cycles needed to achieve the required computation and this is done by using the Atmel AVR Studio [18]. To calculate the communication energy, we calculate the transmission energy only at the node performing the compression using OMNeT++ [19].

Atmel AVR Studio [18] is an Integrated Development Environment (IDE) for writing and debugging AVR/ARM applications. It supports the complete range of Atmel AVR tools and devices. The simulator in AVR Studio can simulate the CPU, including all instructions, interrupts, and most of the on-chip I/O modules. We use Atmel AVR Studio to calculate the required number of cycles as well as the memory requirements of the algorithms.

OMNeT++ [19] is an object-oriented modular discrete-event network simulation framework. OMNeT++ itself is not a simulator, but rather provides infrastructure and tools for writing simulations. It is considered the best simulation framework for WSN as demonstrated in [20]. In order to simulate WSN, we use MiXiM [21], which is an OMNeT++ modelling framework created for mobile and fixed wireless networks. We use OMNeT++ to calculate transmission energy consumption and to model realistic behaviour of nodes in an environment close to reality.

### 4.3. Datasets

To make our results more realistic, we use real-world datasets from various application domains. Our datasets include carbon dioxide monthly measurements in ppm above Mauna Loa (CO2), monthly mean water levels in meters in the lake of the wood at Warroad (Water), the radioactivity in the ground at one minute intervals over one day (Radio) [22], temperature measurements in a garden (Temp) [23], average humidity at Limoges (Hum), sea low level pressure (Pressure) [24]. Table 2 shows the sizes in bytes of the used datasets.

Table 2. The different used datasets and their sizes in bytes.

| **Dataset** | CO2 | Hum | Radio | Temp | Pressure | Water |
|---|---|---|---|---|---|---|
| **Size (bytes)** | 3070 | 264 | 49450 | 10192 | 2405 | 5395 |

## 4.4. Simulation Scenario

Our OMNeT++ simulation model consists of two sensor nodes that are 100 meters apart. One of these nodes sends the data in compressed form while the other one only receives them and replies with MAC acknowledgements. We assume that the data is transmitted in packets having payloads of 64 bytes. No noise or interference was added to the simulation model so no packets were dropped. Sensor nodes are equipped with battery having a nominal voltage of 3.3 V and a nominal capacity of 1000 mAh. Regarding the algorithms, we use S-LZW-MC32 (mini cache of size 32) and K-RLE with K = 2.

## 4.5. Simulation Results and Analysis

The following subsections present the simulation results along with their respective analysis.

### 4.5.1. Compression Ratio

The compression ratio is a very important metric when comparing compression algorithms. In WSNs, having higher compression ratios means lesser amount of data to be transmitted, which means more energy savings. The compression ratio is calculated according to the following equation:

$$Compression\ ratio = 1 - \frac{compressed.size}{original.size}$$

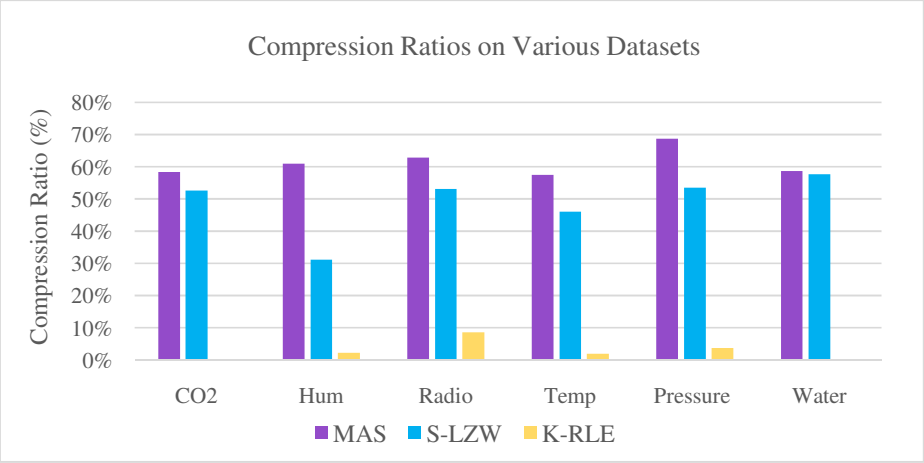


Figure 6. Compression ratios on various datasets.

The results shown in Figure 6, show that MAS beats both S-LZW and K-RLE in all the datasets. MAS' highest compression ratio is about 68.7% and its lowest compression ratio is 57.5%, S-LZW highest compression ratio is 57.7% and its lowest compression ratio is 31.1%, K-RLE does not perform well in compressing these data sets, its highest compression ratio is 8.3% while its lowest is 0%. This could be justified by the fact that numbers found in datasets generated by sensor nodes often do not contain long sequences of repeated symbols.

These results show a great advantage of using MAS to compress floating-point data, since it always achieves better results than the other algorithms.

### 4.5.2. Computation Time and Energy

After compression ratios have been presented it is important to see how much computation cycles does the microcontroller run in order to achieve these results. From computation cycles, we can calculate the compression time required by the microcontroller by assuming that the microcontroller is operating at a frequency of 1 MHz.

Figure 7 shows that in all cases K-RLE requires the least number of computation cycles. This is justified by the fact that K-RLE compression ratios are low, thus the algorithm is not performing all the required procedures. The results for the K-RLE algorithm are not reliable to be used for comparison with the other algorithms since they do not reflect the actual performance of the K-RLE algorithm. For the other algorithms, we notice that S-LZW requires more computation cycles than MAS in all the datasets. In most cases, MAS requires about one-third the amount required by S-LZW. The same justification applies to compression time and computation energy (shown in Figure 8) since they are directly proportional to computation cycles.

Compression time is calculated using the following formula:

$$Compression\ time = \frac{Number\ of\ computation\ cycles}{Frequency}$$

In our experimentation, the microcontroller has in its active state a frequency of 1 MHz.

As for the computation energy, it is calculated using the following formula:

$$Computation\ energy = Computation\ cycles\ x\ Energy\ of\ one\ cycle$$

The energy of one cycle is calculated in section 4.1, and was found to be 3.3 nJ.

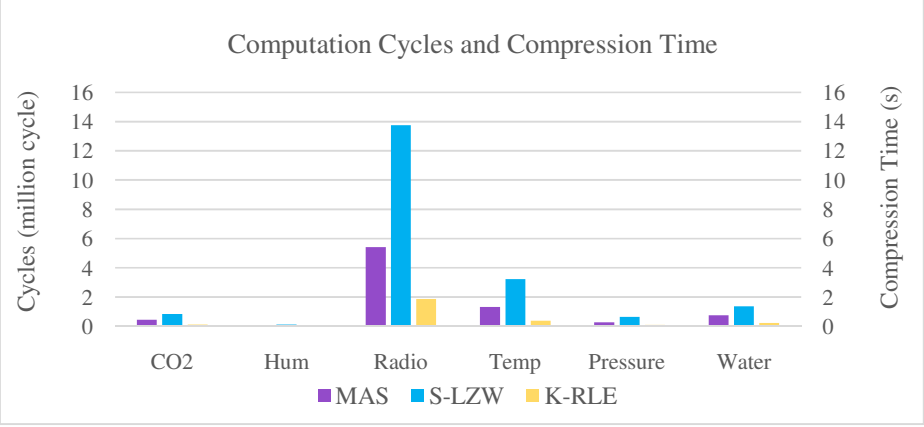


Figure 7. Required computation cycles and compression time on ATmega1281 microcontroller.

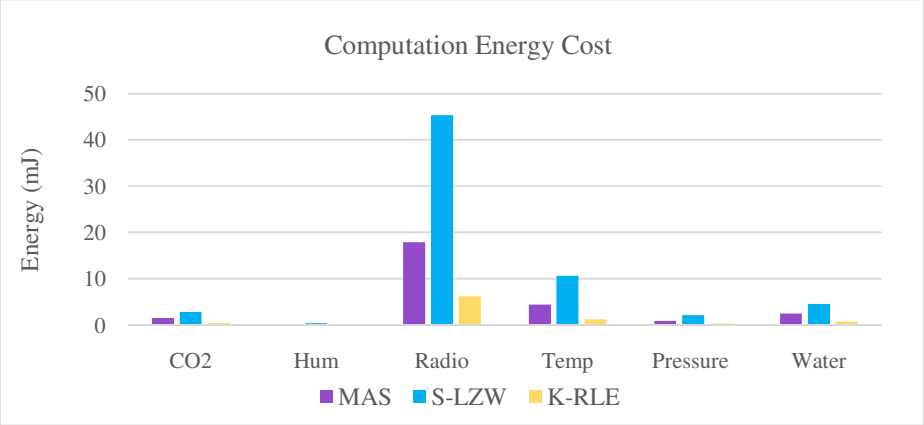


Figure 8. Computation energy cost.

### 4.5.3. Transmission Energy

Computation energy alone is not sufficient to reflect the energy efficiency of an algorithm. The energy efficiency depends on both computation and transmission energy. Transmission energy is the energy consumed by the sensor to send the compressed data wirelessly. Figure 9 shows the energy required to transmit the compressed form of the datasets.

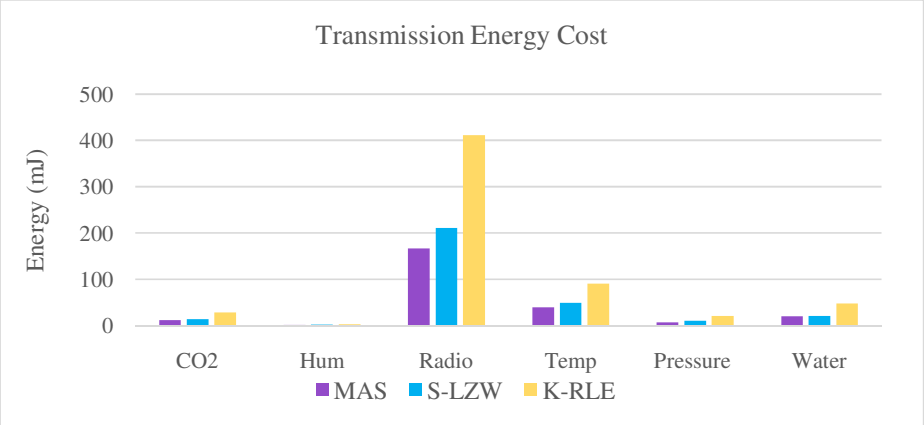


Figure 9. Transmission energy cost

Regarding transmission energy consumption, it is obvious that K-RLE consumes the most energy. This is because K-RLE compression ratio is low, thus it is transmitting larger amounts of data than the other algorithms. Again, MAS consumes the least amount of transmission energy and beats the other algorithm.

### 4.5.4. Total Consumed Energy

The total energy consumed could better reflect the energy efficiency of the three algorithms. Despite the fact that K-RLE consumes the least amount of computation energy, results in Figure 10 show that it consumes the most amount of total energy. This is because it sends large amounts of data over the network. For the other two algorithms, MAS consumes the least energy and thus proves to be a strong candidate for compression in WSN.

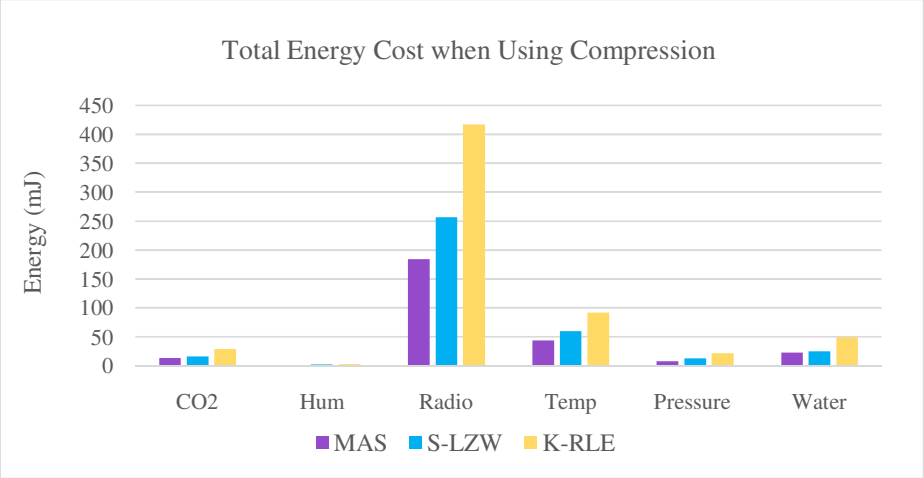


Figure 10. Total energy cost when using compression.

To be able to calculate energy savings of each algorithm, we have to calculate the energy consumed when not using any compression algorithm. The energy consumed when not using a compression algorithm is the energy required to send the data in uncompressed form, so it depends greatly upon the data sizes. That is why we see, in Figure 11, that the Radioactivity dataset is consuming the most energy since it has the largest size.

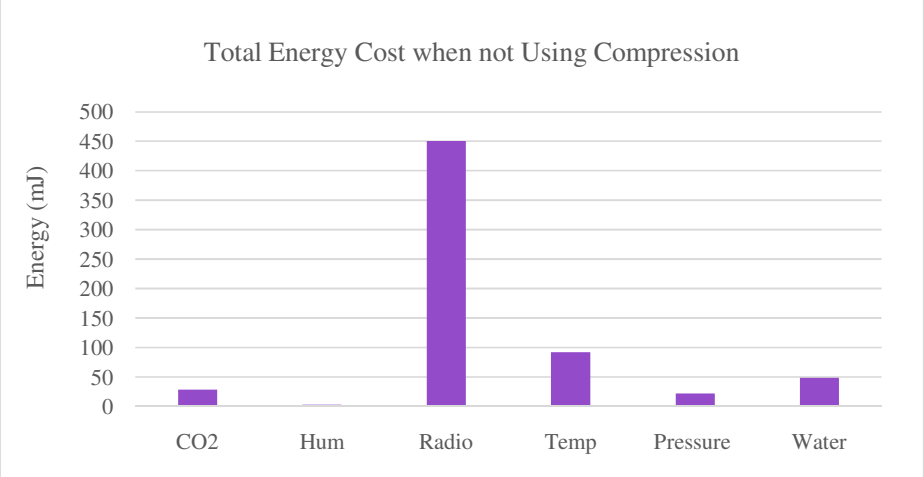


Figure 11. Total energy cost when not using compression.

### 4.5.5. Energy Savings

Figure 12 shows the percentage of energy saved when using each of the three compression algorithm. Energy savings is calculated according to the following formula:

$$Saved\ energy = 1 - \frac{Energy\ with\ compression}{Energy\ without\ compression}$$

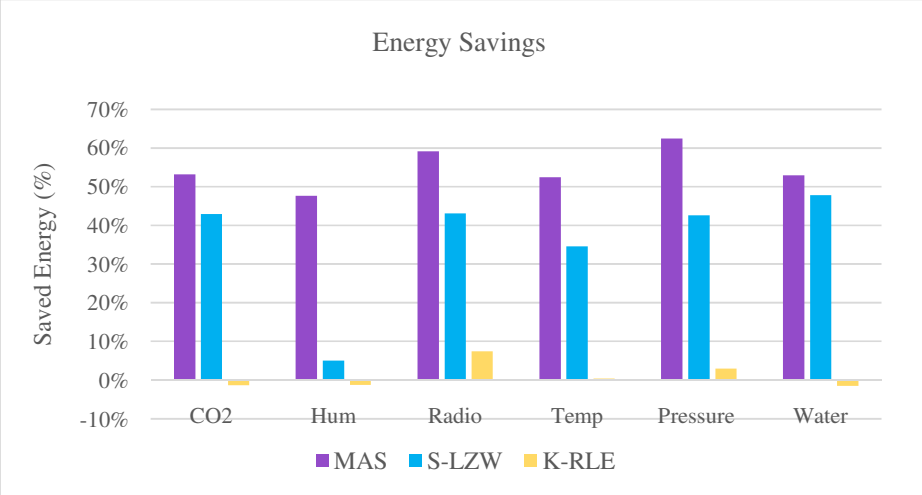


Figure 12. Energy saved when using compression algorithms.

It is clear that MAS achieves the most energy savings with results better than both S-LZW and K-RLE. MAS' energy savings are more than that of S-LZW by an average of 20%, and better than that of K-RLE by an average of 48%.

In some datasets, when using the K-RLE algorithm, the saved energy is negative, it means that the compression algorithm is not saving energy; instead, it is leading to more energy consumption. This is due to the low compression ratios of K-RLE on some datasets.

### 4.5.6. Memory Requirements

To complete the evaluation we must calculate the amount of memory consumed by each algorithm. As a reminder, our platform has a flash memory of 128 KB and a RAM of 8 KB. It is important to note that the memory results are independent of the datasets used. This is because these results are obtained just when building the algorithm and before running any operation or procedure. So these values represent the amount of memory allocated by the algorithm when they are loaded into RAM and before operating on any dataset. Figures 13 and 14 show the absolute and relative memory consumption of each algorithm for flash memory and RAM respectively.

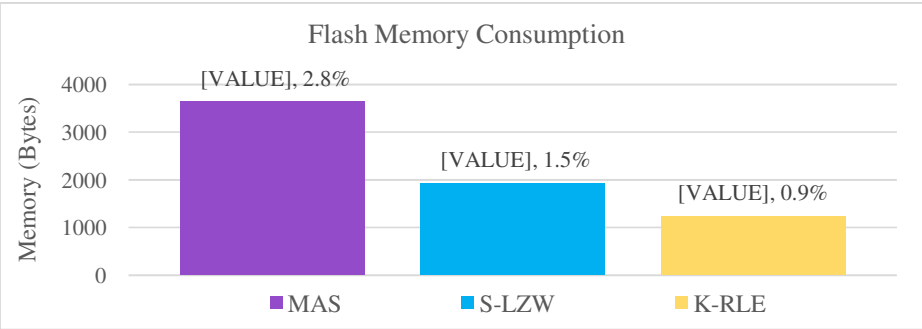


Figure 13. Flash memory consumption.

MAS consumes the largest amount in the flash memory. In fact, MAS program code is a little long since it has two representations for integers and real numbers. Flash memory consumption is not important and it is not a concern, since flash memory is always of a large size, and MAS is

only using 2.8% of that memory. This does not introduce any problems in performance since the flash memory is reserved for program code and not for random access.

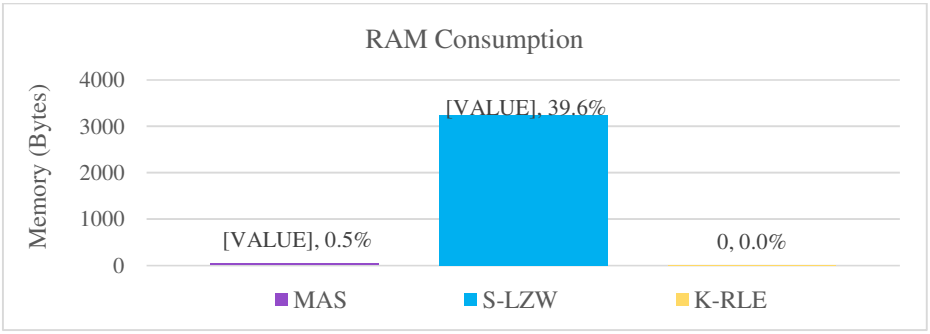


Figure 14. RAM consumption.

In terms of RAM usage, which is the important memory concern, MAS consumes only 44 bytes while K-RLE consumes almost zero bytes. S-LZW turns out to be the most RAM consuming algorithm, and this is because of the dictionary it uses, S-LZW consumes 3240 bytes, which is equivalent to about 40% of the RAM.

These results prove that MAS is a strong candidate for compression in WSNs, since it beats the other algorithms in all the proposed metrics. It achieves compression ratios better than S-LZW by an average of 13% and better than K-RLE by an average of 59%. MAS saves the most amount of energy, it saves by an average of 20% more than S-LZW and 54% more than K-RLE. In terms of memory, MAS and K-RLE use a very little amount of RAM, MAS uses only 0.5% of the total amount of RAM, while K-RLE consumes almost zero bytes of RAM. S-LZW uses the most amount of RAM; it consumes 3240 bytes that is equivalent to 39.6% of the total available RAM.

## 5. CONCLUSIONS

In this paper, we propose MAS, a new lossless floating-point data compression algorithm for WSNs. MAS is applicable to a variety of sensor hardware and platforms due to its low memory and processing requirements.

Simulation results show that MAS' energy savings are on average 54% on all the tested datasets, while maintaining the highest compression ratios. MAS surpasses the other tested compression algorithms in terms of compression ratio, compression speed, memory requirements and energy savings. These results, which are obtained from accurate and trustworthy simulators, present MAS as a strong and competing candidate for data compression in WSN.

## 6. FUTURE WORKS

As a short-term step, we would like to improve MAS to exploit the correlation and the similarity in the data generated by sensor nodes. Such an improvement would allow MAS to achieve higher compression ratios. We would also like to implement a transformation that aims at reducing the number of digits in the input. This transformation is expected to increase MAS' compression ratio since MAS relies mainly on the number of digits in the data to achieve compression

As a long-term step, we would like to study the efficiency of using MAS with an aggregation technique. The main challenge here is to prove that introducing MAS to aggregated networks

does not lead to more energy consumption at the level of aggregators, which are supposed to follow this cycle to achieve their job: decompression – aggregation – compression.

## REFERENCES


[1] N. Kimura and S. Latifi. "A survey on data compression in wireless sensor networks," In Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on, volume 2, pages 8–13 Vol. 2, 2005.

[2] Jennifer Yick, Biswanath Mukherjee and Dipak Ghosal, "Wireless sensor network survey", Department of Computer Science, University of California, Davis, CA 95616, United States, 2008.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks," in IEEE Communications Magazine, Vol. 40, No. 8, August 2002, pp. 102-114.

[4] Chipcon AS. Chipcon SmartRF CC2420 Datasheet rev. 1.3. http://www.ti.com/product/cc2420, October. 2005.

[5] Chipcon AS. Chipcon SmartRF CC1000 Datasheet rev. 2.3. http://www.ti.com/product/cc1000, August. 2005.

[6] MaxStream, Inc. XTend OEM RF Module: Product Manual v1.2.4. http://www.maxstream.net/, October. 2005.

[7] C. M. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," in Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys), 2006.

[8] Capo-chichi, E. P., Guyennet, H. and Friedt, J, "K-RLE a new data compression algorithm for wireless sensor network," in Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications.

[9] Maher Assi, Alia Ghaddar, Samar Tawbi, Ali Jaber, Rami Tawil. MAS: A New Floating Point Compression Algorithm for Wireless Sensor Networks. In Ocean & Coastal Observation: Sensors and observing systems, numerical models & information Systems (OCOSS 2013), October (28-31), Nice, France, in press.

[10] Mark Nelson, "LZW revisited", in Dr. Dobb's Journal, Volume 15 Issue 6, June 1990, Pages 126 – 127, CMP Media, Inc., USA.

[11] Heinola, Finland, "RLE compression", Baltic Olympiad in Informatics, BOI 2006, DAY-2.

[12] IEEE 754: Standard for Binary Floating-Point Arithmetic, http://grouper.ieee.org/groups/754.

[13] Libelium Company, Waspmote wireless sensor platform, http://www.libelium.com/products/waspmote.

[14] ATmega1281, Atmel 8-bit AVR RISC-based microcontroller, ATmega640/1280/1281/2560/2561 Datasheet, revision P, 10/2012.

[15] Atmel Corporation, http://www.atmel.com.

[16] Texas Instruments, CC2420: Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee™ Ready RF Transceiver, Datasheet Rev. C, http://www.ti.com/product/cc2420, 07 Mar 2013.

[17] 802.15.4e-2012 - IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks, http://standards.ieee.org/about/get/802/802.15.html.

[18] Atmel AVR Studio, version 6.1, http://www.atmel.com/tools/ATMELSTUDIO.aspx.

[19] OMNeT++, version 4.3, http://www.omnetpp.org

[20] Xiaodong Xian, Weiren Shi and He Huang, "Comparison of OMNeT++ and other simulator for WSN simulation," college of automation, Chongqing university, Chongqing, 400044, China, 2008.

[21] MiXiM an OMNeT++ modeling framework, version 2.2.1, http://mixim.sourceforge.net

[22] Hyndman, R.J. Time Series Data Library, http://data.is/TSDLdemo

[23] Mischa Dohler Jialiang Lu, Fabrice Valois, « Optimized data aggregation in WSNs using adaptive arma". SensorComm, July 2010.

[24] Weather Forecast and Reports, http://www.wunderground.com

[25] Sameer Tilak, Nael B. Abu-Ghazaleh, and Wendi Heinzelman, "Taxonomy of wireless micro-sensor network models". Mobile Computing and Communication Review, 6, April 2002.

[26] Subhankar Mishra, Sudhansu Mohan Satpathy and Abhipsa Mishra, "Energy efficiency in ad hoc networks", in International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC) Vol.2, No.1, March 2011.

[27] Stephen Wolfram. A New Kind of Science, Wolfram Media, Inc. Champaign IL 2002, United States. IBSN: I-57955-088-8.

[28] Zeeshan Ali Khan and Mustafa Shakir, "Interplay of communication and computation energy consumption for low power sensor network design", in International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC) Vol.3, No.4, August 2012.
[29] D. Salomon, Data Compression: The Complete Reference, Second edition, 2004.
[30] Alia Ghaddar, Tahiry Razafindralambo, Isabelle Simplot-Ryl, David Simplot-Ryl, Samar Tawbi and Abbas Hijazi. "Investigating Data Similarity and Estimation through Spatio-Temporal Correlation to enhance Energy Efficiency in WSNs". In International Journal of Ad Hoc & Sensor Wireless Networks, Vol. 16, pp. 273-295, 2012.
[31] K.Ramanan and E.Baburaj, "Data gathering algorithms for wireless sensor networks: a survey", in International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC) Vol.1, No.4, December 2010.
[32] C.Y. Chong and S. Kumar, "Sensor networks: evolution, opportunities, and challenges", in Proceedings of the IEEE, Vol. 39, No. 8, pp. 1247-1256, 2003.
[33] Eduardo F. Nakamura, Fabiola G. Nakamura, Carlos M. S. Figueiredo, and Antonio A. F.Loureiro. "Using information fusion to assist data dissemination in wireless sensor networks". Telecommunication Systems, pages 237–254, November 2005.

## AUTHORS

**EL ASSI Maher** is a PhD student starting from October 2013. His PhD is arranged by joint supervision between the University of Franche Comte, Besançon – France, and the Lebanese University, Beirut – Lebanon. He received M.S. degree in the Lebanese University in 2012. He also graduated as a Civil Engineer in 2012 from the Lebanese University. His research interests include Internet of Things (IoT), Wireless Sensor Networks (WSN), energy efficiency in embedded devices, network modelling and simulation and lightweight data analysis techniques.

**GHADDAR Alia** has PhD in Computer Science with more than 6 years of teaching experience and 4 years of web development. Obtained her Master degree in Computer Science from the Lebanese University. She started her PhD at the University of Lille-1, Science and technology in France. During that time, she was member in POPS-project team; A joint project of INRIA, University of Lille-1 and CNRS. Her interests now lie in the Internet of things, mobile sensors, data communication and knowledge discovery in the wireless sensor networks.

**TAWBI Samar**, PhD in Computer Science. She is an associate professor in the Computer Sciences department in the Lebanese University since 2005. She received the M.S. degree in "Mathematical Modelling and Scientific Software Engineering" by the Lebanese University and the universities of Rennes, Reims (France) and EPFL (Switzerland). She obtained her PhD in 2004 at 'Paul Sabatier' University, France. Her research interests include data aggregation and communication in wireless sensors networks, auto configurable WSN, Internet of things and Cloud computing.

**GHADDAR Fadi**, MS in Network and System Security. He has more than 15 years of experience in software analysis and development, network and system security and team leading. He obtained recently his Master degree from Saint Joseph University. His research interests include information security, data aggregation and security in wireless sensors networks.