C-RLE: Energy efficient compression approach for Wireless Sensor Networks

Marwen Roukhami[1*], Younes Lahbib[2], Abdelkader Mami[1]

[1]UR-LAPER, Faculty of Sciences of Tunis, University of Tunis El-Manar, 2092 Tunis, Tunisia
[2]LR-EμE, ENICarthage, University of Carthage, 2035 Tunis, Tunisia

Summary
In wireless sensor network (WSN), data transmission and acquisition are the two main activities. However, the actual transmission / reception of the collected data is often the most energy consuming task, which affects the lifetime of WSN. One promising approach to reduce the total power consumption of the sensor node is data compression before transmission. In this article, we propose and evaluate a new compression approach, called C-RLE. It is based on the principle of the K-RLE algorithm. Proposed C-RLE solves the problem of trade-off between energy consumption and compression rate efficiency in K-RLE. To prove the effectiveness of our proposed solution, we compare its performance with both RLE and K-RLE algorithms using real-world datasets. We checked this effectiveness also using software and hardware implementations. The C-RLE proposed algorithm is firstly coded in C and implemented on Cortex-M3 based CPU. Then, a hardware architecture is descripted in VHDL and integrated on Spartan 3A FPGA platform. Experimentation shows that our proposed C-RLE approach keeps the same K-RKE's performance in term of compression ratio while the energy consumed can decrease up to 27.03% et 16.67% compared to the K-RLE and RLE algorithms, respectively.

Key words:
Wireless sensor networks (WSNs), low power data compression, energy efficiency, hardware design FPGA, Cortex-M3 STM32L1.

1. Introduction

A wireless sensor network (WSN) is made up of a set of sensor nodes containing sensing and processing devices with a wireless communication and a small battery. The functionality of these WSN nodes is constrained by its power consumption and its limited computational complexity that the hardware can support. However, these nodes are mainly powered by non-rechargeable embedded batteries, making energy consumption one of the crucial characteristics in WSN, which consequently causes special challenges with energy efficiency in data processing and communication [1].

Various studies, such as [2] [3] demonstrated that the energy consumed to transfer or receive one bit of information is equal to the energy consumed to execute three thousands of instructions in the processing unit. Therefore, one of the main goals of the WSN designers is reducing the radio transmission, using data aggregation and/or compression techniques to reduce the amount of packets sending. These techniques have the potential to limit the power supply nodes in order to increase the life time of WSN.

On the other hand, data compression in WSN is highly discussed as a promising solution to energy optimization. In fact, the complexity of the same data compression can cause an increase in the processing energy. However, traditional compression algorithms, such as LZW, JPEG and WinZip, are not suitable for use in WSN due to their processing complexity and required hardware resources [4] [5], which increase energy consumption. As a result, it is better to use the ad-hoc compression algorithms in WSN.

This paper studies the performance of one of the compression ad-hoc algorithm which is K-RLE [9] and shows that that the trade-off between its energy consumption and compression efficiency is related to both implementation method and the definition of the precision parameter K. By defining a new precision parameter C, a new optimized K-RLE implementation is proposed, called C-RLE. The latter achieves an important gain in both compression ratio and energy consumption. Experimental results on an ultra-low power microcontroller show that C-RLE has a similar compression ratio as the K-RLE. It has in contrary, efficient energy consumption compared to the K-RLE and the traditional RLE algorithm.

In addition, a hardware design for the new C-RLE approach is proposed, which can be used in sensor nodes using reconfigurable hardware devices as proposed in [6] [7] [8]. The performance of the hardware implementation are evaluated on FPGA Spartan 3A according to a real-world datasets. A comparison against RLE algorithm is processed on the same platform with same condition. Results show that with our proposed C-RLE we can reduce the energy consumption more than 16.87% compared to the basic RLE algorithm.

The remainder of this paper is organized into five sections. Section 2 discusses the related works. Section 3 gives a brief theoretical description of both RLE and K-RLE algorithms and details the new proposed implementation of K-RLE. Section 4 presents our software solution with

Manuscript received June 5, 2018
Manuscript revised June 20, 2018

the measurement results obtained. In section 5, our hardware implementation of C-RLE and its performance in terms of execution time and energy consumption are presented. Finally, concluding remarks are drawn in section 6.

2. Related works

In literature, two main compression ad-hoc algorithms have been proposed especially for WSN [10] [11] [12], S-LZW and K-RLE. They are derived from traditional algorithms and used as a reference to evaluate the performance of a new compression algorithm in WSN [13]. S-LZW [14], for Sensor-LZW, is an adaptation version of the popular data compression algorithm ZLW [15]. It is a lightweight algorithm to use with the constrained resources of sensor nodes. K-RLE [9], is proposed to improve the effectiveness of the RLE algorithm to be useful in WSN. It is inspired from the RLE principle, namely by adding a precision parameter called K. This latter represents the difference between two input data streams and takes the same unit as the Pascal of pressure, the degrees of temperature, etc. In this article, our study focuses on the K-RLE algorithm.

In [9], the K-RLE implementations show a trade-off between efficiency of compression and energy consumption. When k equals to 2, 2-RLE affords the best compression ratio compared to both S-LZW and RLE [9]. Compression ratio exceeds to 40% than RLE, but energy, in contrary, is more consumed by 2-RLE than S-LZW and RLE [9]. In this regard, two other K-RLE implementations have been done in [16] and [17]. In Ref. [16], R.S. Pisal presented a software implementation of K-RLE on ARM7 microcontroller; and in Ref. [17], K. Yamin et al presented a hardware implementation on FPGA using the VHDL language. Both works proved again that K-RLE is widely better than RLE in terms of compression ratio, without improvements in terms of energy consumption.

In this paper, we propose a new optimized implementation of K-RLE called C-RLE, which overcomes the disadvantages of both original RLE and K-RLE algorithms. It is able to achieve the effectiveness in both energy consumption and compression ratio.

3. Theoretical background of K-RLE and C-RLE approach

In this section, a brief introduction of the RLE lossless compression algorithm is given. The K-RLE is then introduced showing the required details for the proposed C-RLE. Finally, the new proposed C-RLE solution is discussed.

3.1 RLE algorithm

Run Length Encoding (RLE) is a simplest lossless compression technique which takes advantage of repetitive values in a sequence of data. Generally, it is used to decrease the physical size of longer data sequences, which consists of repeating characters. In practice, this compression algorithm can be used to monitor every repetitive and redundant data, such as pressure, humidity, temperature, etc.

The main idea behind RLE is described in [18] as follows: "If a data item d occurs n consecutive times in the input stream, we replace the n occurrences with the single pair nd". A specific character is added before every single pair nd in the data compression to discover the new data pack compressed during the process of decompression. Moreover, the RLE is a lossless data compression, where the original data are restored without loss of information.

However, the simplicity of RLE is an asset compared to limitations related to the resources of the sensor nodes, such as the storage capacity and the lack of computing power. The effectiveness of RLE is narrowly related to the repetition number of the same data in the input data. Capo-Chichi et al. [9] overcome this limitation and proposed a new algorithm named K-RLE inspired from RLE to improve the effectiveness of RLE technique compression.

3.2 Principle of K-RLE algorithm

The main idea behind K-RLE algorithm is to improve the performance of RLE in terms of compression ratio. Based on a new parameter called K, the principle of the K-RLE algorithm is summarized as follows [9]: "If a data item d, d+K or d-K occurs n consecutive times in the input stream, we replace the n occurrences with the single pair nd".

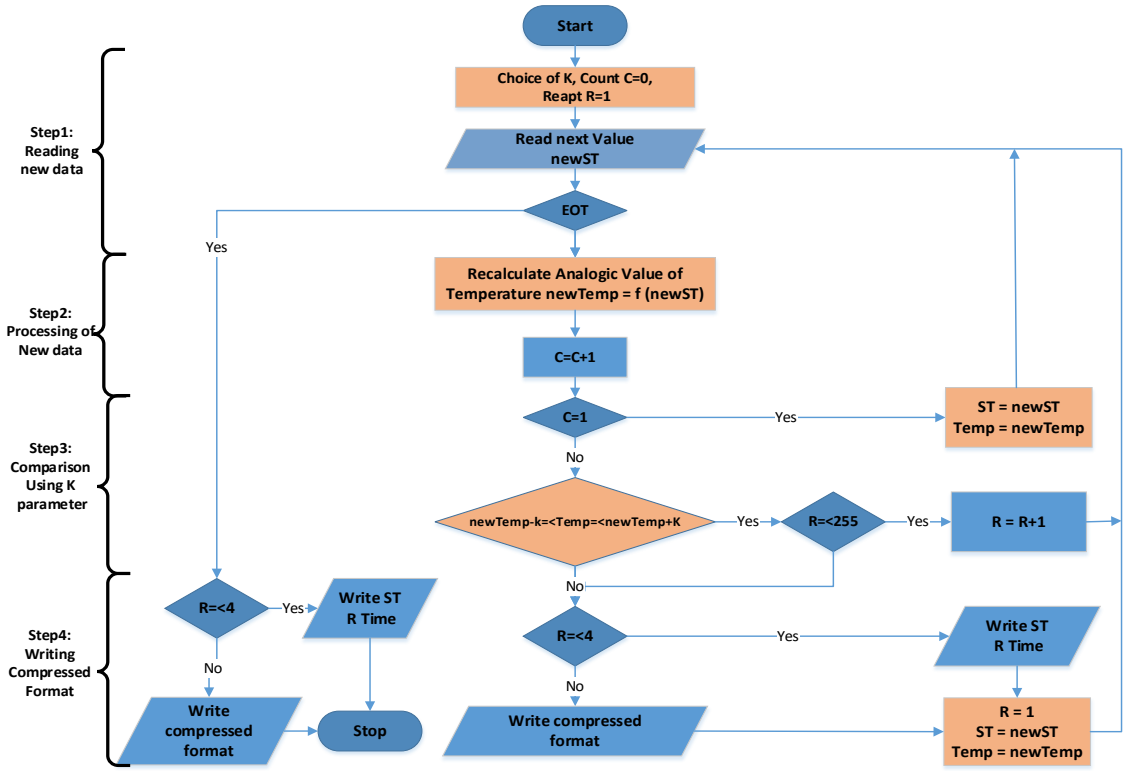

Fig. 1 Flowchart of K-RLE compression algorithm.

Fig. 1, shows the flowchart of K-RLE algorithm. Compared with that of RLE, the orange columns indicate the instructions added with the presence of new precision parameter K. This new precision parameter K is the allowed margin between two successive values, which allows to consider the elements d, d + K or d-K as identical. This technique can be useful when small variations in the input data is not significant. Finally, it is important to note that K has the same unit as the supervised data.

The performance of this algorithm is compared to two data compression algorithms (RLE and S-LZW) in terms of compression ratio and energy consumption. This study [9] is made on a low power microcontroller which is MSP430 using a real temperature dataset. The obtained experimental results prove that for K equal to 2, 2-RLE offers a better compression ratio that can achieve 40% greater than the RLE. On the other hand, its energy consumption is higher than RLE and S-LZW, hence the trade-off between compression efficiency and energy consumption [9].

In the next section, we propose a new method called C-RLE to implement the K-RLE algorithm where we show that the problem related to energy over- consumption lies in the definition of the precision parameter K.

3.3 Proposed C-RLE approach

In this section, we present the C-RLE approach, which is our main contribution to preserve the compression ratio as in K-RLE but with reduced the energy consumption.

As shown in Fig. 2, the K-RLE execution stages constitute a sequence of four main steps: ADC conversion for reading a new value, Data processing, Comparison phase and Data compression. The first step consists of reading the data derived from sensors after the analog-to-digital conversion process. Secondly, because the parameter K has the same unit as the oversaw data, it is necessary to retrieve each analog value for the comparison stage. This implies going through the phase of data processing to recover the converted values into analog one again. Once the analog value is processed, it will be used in the comparison phase with the precision parameter K. At the end, data compression is carried out.

Comparing the execution of RLE to that of K-RLE, it is not mandatory to go through the processing phase to recalculate the analog value. In fact the RLE does not introduce the term of precision parameter to execute additional instructions or to spend additional time to recover each supervised analog value as in the case of K-RLE. This explains the over-consumption of energy for K-RLE despite its best result concerning compression ratio.

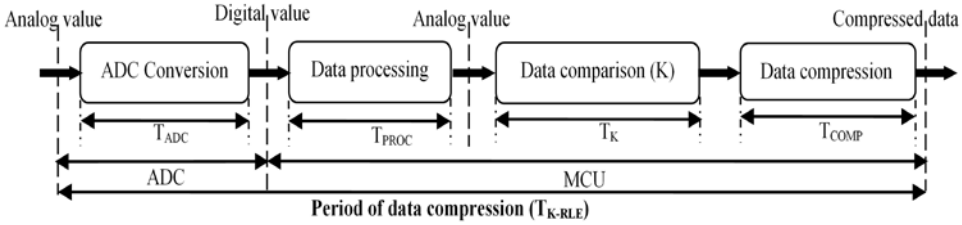

Fig. 2 The main execution stages of the compression algorithm K-RLE.

Proposed solution is founded on a new precision parameter C, which is proportional to the K-parameter. This new C-parameter will be used directly with digital values derived from the conversion phase without going through the data processing phase. Consequently, it will not be necessary to recalculate the analog values and execute additional instructions. This new precision parameter C, hence the label C-RLE, will not take any unit because it will be considered as the allowed margin between two successive digital values.

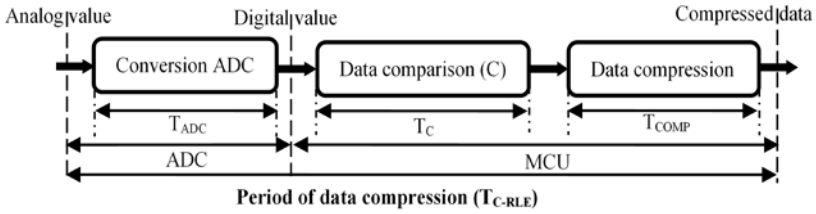

Fig. 3 The main execution stages of the proposed compression algorithm C-RLE.

According to Fig. 2 and Fig. 3, the conversion period TADC and compression period TCOMP are the same during C-RLE or K-RLE execution. The data comparison times TC and TK are approximately the same. However, an extra processing period (TPROC) is performed for reconverting digital data to real analog data in the case of K-RLE, which is not necessary in our new C-RLE method. Theoretically, our method will decrease the number of instructions and the execution time needed for this compression task.

The power consumption of the nodes is estimated according to P = IDD * VCC, where IDD is the average of the current MCU in a given period T and VCC is the polarization voltage. The energy consumption is calculated as follows E = P*T = IDD * VCC * T. As the MCU performs the same processing during the coordinated periods, the MCU will consume the same current (IDD). Consequently, an extra energy consumption is estimated in the case of the K-RLE implementation as presented in equations (1) and (2):

$$E_{C-RLE} = \mathrm{V_{CC}} * (I_{ADC} * T_{ADC} + I_C * T_C + I_{COMP} * T_{COMP}) \quad (1)$$

$$E_{K-RLE} = \mathrm{V_{CC}} * (I_{ADC} * T_{ADC} + I_K * T_K + I_{COMP} * T_{COMP}) + \mathrm{V_{CC}} * \mathrm{I_{PROC}} * \mathrm{T_{PROC}} \quad (2)$$

In order to achieve the main objective of this study, two separate physical implementations are performed. The first one, a software implementation of the RLE, K-RLE and C-RLE algorithms on an ultra-low power microcontroller, the STM32L1. We demonstrate the effectiveness and benefits of our C-RLE approach compared to previous proposals K-RLE and RLE. On the other hand, a hardware implementations of C-RLE and RLE algorithms on a Programmable Device (FPGA Spartan 3A). We present the hardware design of C-RLE and its performances compared with respect to the RLE in terms of execution time and energy consumption.

The performances of previous compression algorithms for both implementations are evaluated using a real-world benchmark of temperature datasets of 2196 bytes. This temperature data sets are collected from the Weather Underground website [24] during the period from the 1st of January 2016 up to the 31st of August 2016 in Tunis (Tunisia), Paris (France) and Toronto (Canada).

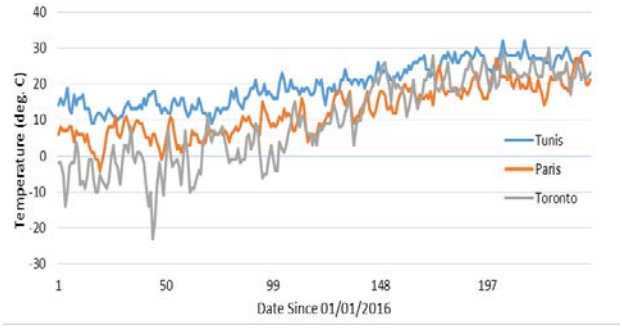

Fig. 4 The representation of the temperature archives of Tunis.

We choose different locations with different variances between the successive input data to study the behavior of the previous algorithms in different real conditions. During the compression task, these digital data will be used from the memory RAM of the STM32L1 or the FPGA.

4. Software implementation

The software implementation is intended to validate the performance and the efficiency of our proposed C-RLE. In this section we present the developmental tools and the experimental steps used in this implementation to measure the execution time, the number of cycles and the energy consumption for each algorithm RLE, K-RLE and C-RLE. Finally, we present the experimental results of the software implementation.

4.1 Implementation of C-RLE and measurement Framework

The standard STM32L1 discovery board was used in this practice. It is based on an Ultra-low power microcontroller STM32L152R including a high-performance Microprocessor ARM cortex-M3. The choice of this card is made for two reasons. First, this board includes an ultra-low power microcontroller which shows its effectiveness in several studies and very usefulness as a solution in wireless sensor nodes [19] [20] [21] [22]. Secondly, an integrated circuit in this card allows to measure the real current consumption of the microcontroller in various modes like run mode, sleep mode, etc… It is useful in our studies to find the power consumption during the execution of the algorithms and also in separate processing periods.
The software tool used in this section is the Integrated Development Environment (IDE) Keil vision ®, which can be interfaced with the ST-link debugger. In the debugging and simulation phases, this software has the advantage of supervising the real execution of the algorithms, instruction by instruction into the microcontroller, as well as estimating the execution time and the number of cycles for each program in real time.

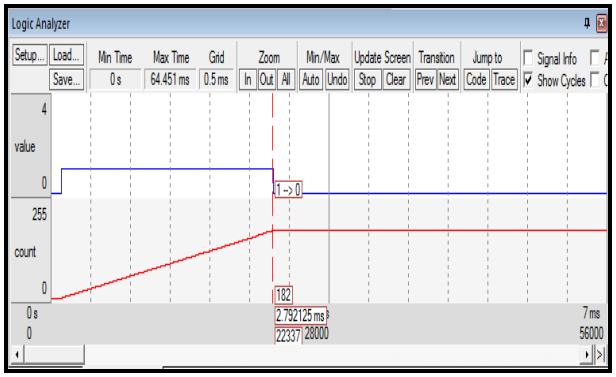

Fig. 5. GUI logic analyzer of keil development tools.

After measuring the execution time and the number of cycles required to execute each compression algorithm, we proceed to calculate its consumed energy through its consumed current, which is measured by the integrated circuit (IDD) in the STM32L1 board.
It should be noted that all the measurements of the current are read via a dedicate jumper called JP1 in the STM32L1 board by connecting it to a high-resolution ammeter.
The performance evaluation of our C-RLE method in this software implementation is described in the following section.

4.2 Software implementation results

Firstly, we check the compression ratio obtained with our compression approach. As mentioned above, to study the behavior of the previous algorithms in different real conditions, we use three different locations with variable deviations between the successive input data. Table.1 presents the data size of different datasets used for our evaluation as well as the size of the compressed data and the compression ratio obtained.

Table 1: Compression results by applying various compression implementations

Data	Algorithm	RLE	K-RLE		C-RLE	
			1K-RLE	2K-RLE	1C-RLE	2C-RLE
Tunis	Original Size (Byte)	732	732	732	732	732
	Compressed Size (Byte)	717	558	348	558	348
	Compressed Ratio	2.05%	23.77%	52.46%	23.77%	52.46%
Paris	Original Size (Byte)	732	732	732	732	732
	Compressed Size (Byte)	726	633	498	633	498
	Compressed Ratio	0.82%	13.52%	31.97%	13.52%	31.97%
Toronto	Original Size (Byte)	732	732	732	732	732
	Compressed Size (Byte)	732	678	639	678	639
	Compressed Ratio	0%	7.38%	12.7%	7.38%	12.7%

As shown in Table.1, the RLE compression algorithm supplies an overly low compression ratio in the three different data sets. For example, processing the data collected from Toronto, RLE provides a compression ratio of 0%. We can clearly see that RLE algorithm is highly dependent on the repetitive nature of data stream, which underlines the importance of using the precision parameter in K-RLE and C-RLE.

On the other hand, K-RLE and C-RLE achieve an improved compression ratio using different values of the precision parameter compared to the RLE. With a precision parameter equal to 2, the compression ratio in both K-RLE and C-RLE cases are respectively 52.46% and 31.97%. But, it can be noted that the high variance between the successive values influences the compression ratio. For example, for the data collected from Toronto, the compression ratio is equal to 12.7% even with a precision parameter equal to 2.

Finally, it can be seen that K-RLE and C-RLE provide the same compression ratio since theoretically they have the same principle, but with a difference in the definition of the precision parameter.

In a second step, we analyze the number of cycles and the execution time required for each algorithm during the compression task with an MCU frequency equal to 1 MHz. Also, we present the effect of the new precision parameter. We recall that with the new approach introduced to implement the k-RLE algorithm, there is no need to recalculate the analog value for each value deal.

Fig. 6 shows that 1K-RLE and 2K-RLE use more cycles compared to the other algorithms. There is a difference in the average of about 20.59% between 2K-RLE and RLE, which affirms the results obtained in [9]. However, we can see that the best results are obtained by our approach. From Fig. 6, it is clear that the C-RLE, with two precision parameters 1 and 2, is better than K-RLE and RLE algorithms.

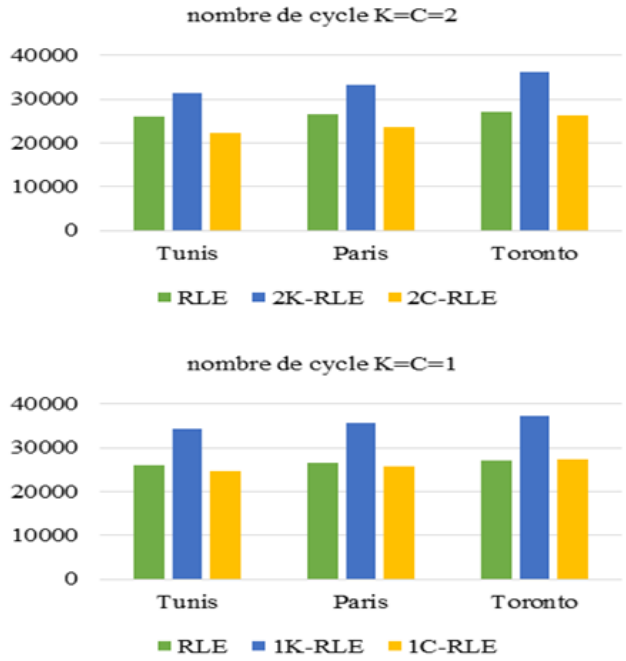

Fig. 6 Number of cycles required in various compression implementations.

Finally, the number of cycles required by K-RLE to perform the compression task of the data used for this study is reduced in the average of 28.06% when the new C-RLE approach is used. Also, the average gain for C-RLE can reach 15.17% compared to RLE.

These previous results have a direct effect on the execution time which is proportional to the frequency of the CPU system as well as the number of cycles needed for each task.

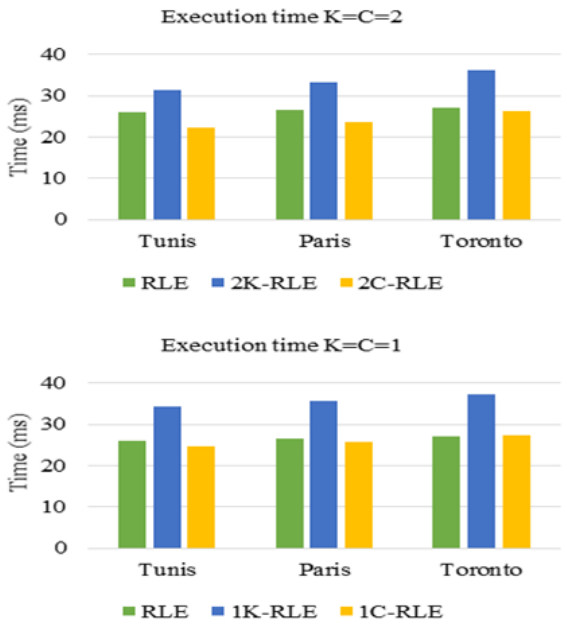

Fig. 7 Execution time required in various compression implementations.

Fig. 7 illustrates the execution time of the previous algorithms with different precision parameters, which enhances the previous evaluation of the number of cycles.
Finally, we present the most important evaluation criteria, the energy consumption, which is dependent on the execution time and the power consumed by the MCU.

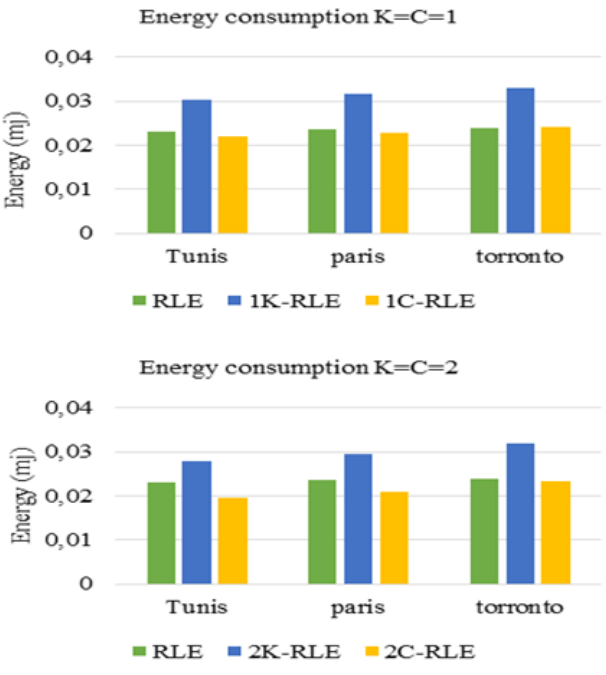

Fig. 8 Energy consumed for different compression implementations.

Fig. 8 shows the energy consumption for each algorithm according to different precision parameters. We can see that the energy needed to compress the data collected from Paris by RLE is 0.024 mj and most of the energy consumed by 2K-RLE is about 0.0294 mj. But, 2C-REL uses about 0.0209 mj, which is lower than the others.
It is clear, that the K-RLE algorithm is the most energy-intensive for all different cases. On the other hand, RLE and C-RLE approximately provide the same results for C=1, but for C= 2, C-RLE consumes less energy than RLE. In summary, Fig. 8 shows that our approach can save more than 27.03% of the required energy compared to the K-RLE algorithm. Also, compared to the original algorithm RLE, energy consumption is reduced to about 9.82% when C=2 and to 2.52% for C=1.

5. Hardware implementation

The goal of the hardware implementation is to present the hardware design of the C-RLE and evaluate its efficiency compared to the basic RLE algorithm in a reconfigurable device, which can be used in the future in sensor nodes either based on reconfigurable resources or as a co-processor. Yet, if we want to implement the K-RLE algorithm in sensor nodes, its hardware structure required to implement the transfer equations of each sensor used, which is a very complex and costly task.
This section presents the implementation details of the hardware design of the C-RLE compression technique, the developmental tools and the experiment steps used to measure the execution time and the energy consumption. Finally, we present the experimental results of this hardware implementation.

5.1 Architecture of C-RLE compression system

The details of our proposed Hardware architecture of C-RLE are presented in Fig. 9 which includes three basic modules.
The first module (read and compare a new value) is in charge of the comparison between the new value that is measured by the sensor and the previous one which was stored on a parallel 12-bit register. The result of this comparison “Eg” is sent to the second module, FSM controller, which has the role of generating the necessary signals either to increment the repeating counter "rep" as well as reading the next new value or allowing the processing of the compression phase. For the third module (compression formatting), by using the repeat value "rep", the decoder selects the compression form either the specific character "&" followed by the repeat value and the data repeated or the data repeated directly if this repeat value "rep" does not exceed 4.

In this proposed design, we take into consideration that the FPGA supports fully-parallel execution. We have implemented the two modes of comparison and compression to run at the same time just after the first data compression operation, which obviously results in a gain in the execution time. Regarding the implementation architecture of RLE, it is totally identical to that of C-RLE except the small comparator block, where the RLE does not use any precision parameter.

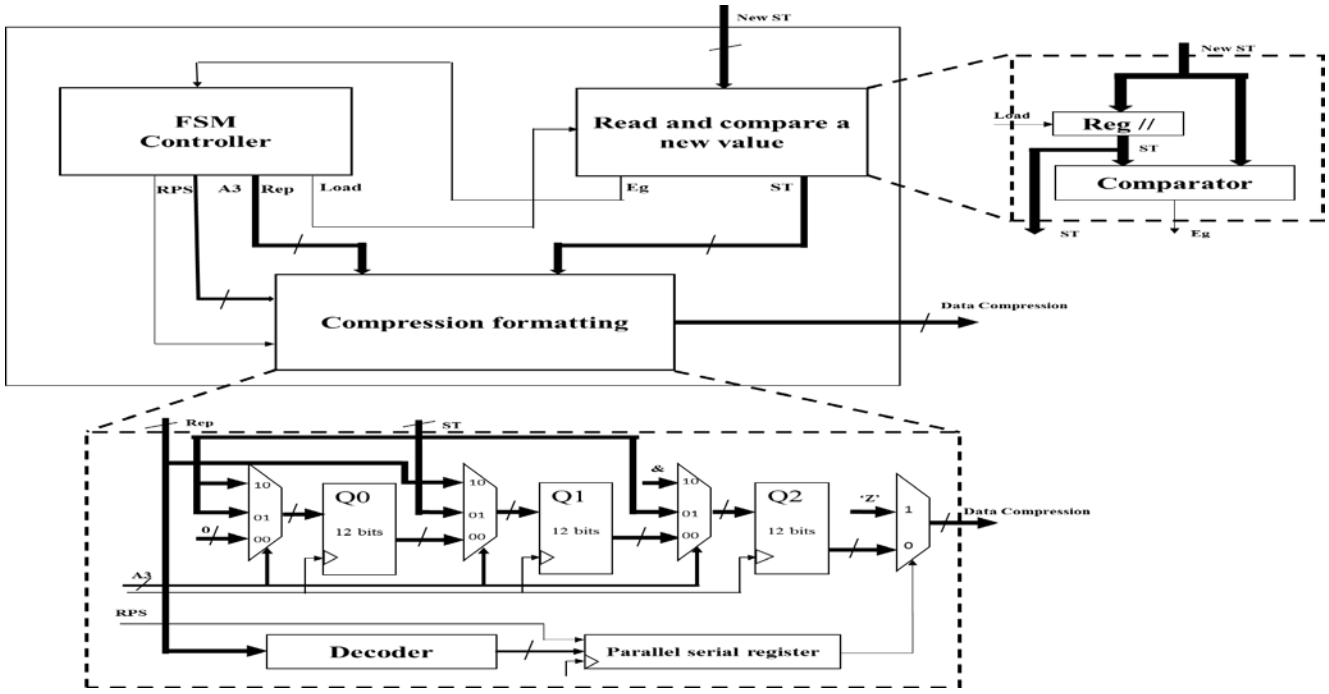

Fig. 9 The detailed architecture of the C-RLE compression.

These proposed hardware designs are synthesized and simulated on the Spartan 3A FPGA using the Xilinx 14.2 tools. While its embedded simulator ISim is used to verify the correct operation of our implementation and the process of "behavioral simulation" is used to determine the execution time for each algorithm.

On the other hand, measuring the power consumption for the internal core of FPGA can be a challenging procedure, especially if the board in which the experiments are carried out is not specifically designed for such experiments. Furthermore, most of the boards of FPGA do not provide the possibility of measuring the power consumption of FPGA itself.

In this framework, the reference of FPGA Spartan 3A starter kit has been created to conduct experiments and measurements in order to determine the actual power consumption. As described by Chapman in [23], the power dissipation of any hardware design for the FPGA Spartan 3A is presented in equation (3) [23] as follows:

$$P_{dissip}(mw) = (V_{ccint} * I_{ccint}) + (V_{ccaux} * I_{ccaux})$$
$$P_{dissip}(mw) = (1.2v * I_{ccint}) + (3.3v * I_{ccaux}) \quad (3)$$

Where the Iccint is the current provided for the internal FPGA core with a voltage Vccint equal to 1.2V and the Iccaux is the current provided for the I/O pins with a voltage Vccaux equal to 3.3V. All the measurements of currents Iccint and Iccaux are read by two available pins available on the board using an ammeter.

5.2 Hardware implementation results

The two hardware proposed design of the C-RLE and RLE algorithms are implemented in the FPGA Spartan 3A. The results of the resource occupation are presented in Table 2.

Table 2: Resources occupation for the Hardware implementations of C-RLE and RLE

Resources	C-RLE	RLE
Number of Slices (5888)	140 (2%)	118 (2%)
Number of input LUTs (11776)	257 (2%)	214 (2%)
Number of IOBs (372)	26 (6%)	26 (6%)

Table.2 shows that a large amount of FPGA resources are still available for the implementation of other tasks in parallel.

In order to prove the efficiency of our C-RLE proposed approach with the hardware implementation, we measure

the processing time and the energy consumption reached during the execution of C-RLE and RLE on the reconfigurable device FPGA Spartan 3A.
These measurements are performed just during the compression of the data collected from Tunis with different frequencies between 1 MHz and 50 MHz.

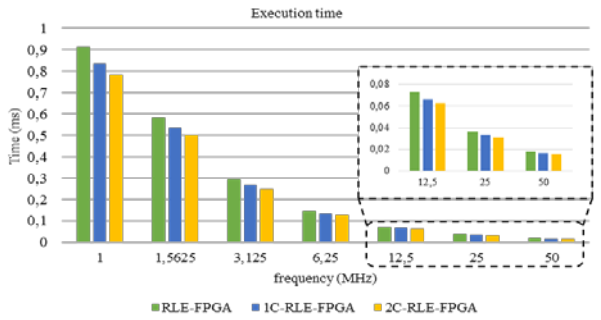

Fig. 10 Execution time required by the hardware design proposed.

With regard to the Fig. 10, we can see that the C-RLE with different precision parameters needed fewer execution time than RLE for all different frequencies. There is a difference of about 11.38 % in the average time between C-RLE and RLE. Although the two architectures of C-RLE and RLE are completely identical except the compression block, these results are mainly related to the use of precision parameter by the C-RLE algorithm, which also affirms the software results.
In this final part, we measure the energy consumption of our C-RLE approach and RLE algorithm for its hardware implementations. To do so, we measure the power consumption of the FPGA during the processing of compression tasks, which is in the order of 37.66 mW.
Fig. 11 illustrates the comparison of energy consumption between the hardware implementations of the C-RLE and RLE algorithms.

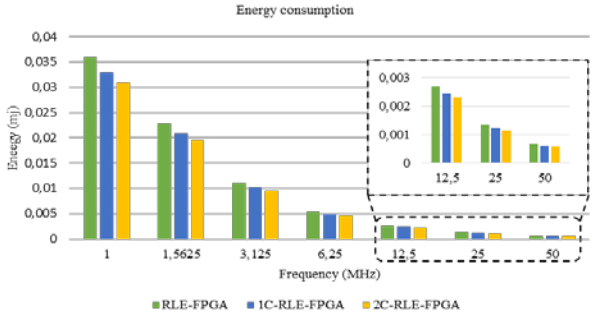

Fig. 11 Energy consumption of the hardware design proposed.

It is remarkable that RLE consumes more energy than C-RLE, for example RLE uses 0.036 mj to compress data with a frequency equal to 1 MHz while 2C-RLE uses about 0.03 mj. Due to this hardware design, the C-RLE can reduce the energy consumption more than 16.67% compared to the basic RLE algorithm.
In summary, results of software and hardware implementations obviously show that with our approach we can keep the performance of the K-RLE in terms of the compression ratio, which is higher than RLE, without having any overconsumption of energy.

6. Conclusion

A simple new approach named C-RLE is proposed based on the principle of the K-RLE algorithm, which shows a trade-off between energy consumption and compression efficiency. This solution focuses on the definition of the precision parameter K and exploits its principle to get an efficient data compression in WSN. Firstly, the performance of our proposed approach is compared to RLE and K-RLE algorithms on an ultra-low power microcontroller which is STM32L1using real temperature data sets from three different places. This software implementation shows that our proposed C-RLE implementation keeps the performance of the K-RKE in terms of the compression ratio and decreases the consumed energy to 27.03% and 9.82% compared to the K-RLE and RLE algorithms, respectively.
Next, a hardware design of our approach is presented and the performance of this hardware implementation is evaluated on FPGA Spartan 3A with real-world experiments and compared to the basic RLE algorithm. Based on the hardware implementation results, we have demonstrated also that C-RLE needs less execution time and energy consumption than the RLE algorithm.

References

[1] M.L. Kaddachi, A. Soudani, V. Lecuire, K. Torki, L. Makkaoui, JM. Moureaux: 'Low power hardware-based image compression solution for wireless camera sensor networks', Computer Standards & Interfaces, Vol.34 (1), pp. 14-23, January 2012.
[2] N. Kimura, S. Latifi: 'A survey on data compression in wireless sensor networks', In: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05), pp.8-13, 2005.
[3] G.P. Pottie, W.J. Kaiser: 'wireless integrated sensor networks', Magazine Communications of the ACM, Vol.43 (5), pp.51-58, May 2000.
[4] D. Faundez, V. Lecuire:' Error Resilient Image Communication with Chaotic Pixel Inter-leaving for Wireless Camera Sensors', In: Proceedings of the 2008 Workshop on Real-World Wireless Sensor Networks (REALWMSN'08), 2008.

[5] K.C. Barr, K. Asanović: 'Energy-aware lossless data compression', ACM Transactions on Computer Systems, Vol.24 (3), pp.250-291, Aug. 2006.
[6] Y. E. Krasteva, J. Portilla, J. M. E. de la Torre, and T. Riesgo, "Remote HW-SW Reconfigurable Wireless Sensor Nodes," Proc. of 34th Annual Conference of IEEE Industrial Electronics (IECON2008), pp. 2483- 2488, Nov. 2008.
[7] H. Hinkelmann, P. Zipf, and M. Glesner, "Design Concepts for a Dynamically Reconfigurable Wireless Sensor Node," Proc. of the 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06), pp. 436-441, Jun 2006.
[8] A. M. Obeid, F. Karray, M. W. Jmal, M. Abid, S. Manzoor Qasim and M. S. BenSaleh, "Towards realisation of wireless sensor network-based water pipeline monitoring systems: a comprehensive review of techniques and platforms", IET Science, Measurement & Technology, Vol.10 (5), August 2016.
[9] E.P. Capo-Chichi, H. Guyennet, J.-M. Friedt: 'K-RLE: a new data compression algorithm for wireless sensor network', In: Proceedings of the 3rd International Conference on Sensor Technologies and Applications (SENSORCOMM '09), pp. 502–507, June 2009.
[10] Jonathan Gana Kolo, S. Anandan Shanmugam, DavidWee Gin Lim, Li-Minn Ang, and Kah Phooi Seng: 'An Adaptive Lossless Data Compression Scheme for Wireless Sensor Networks', Journal of Sensors Volume 2012.
[11] CH. Eugene, F. Jean-Michel, G. Herve: 'Using Data Compression for Delay Constrained Applications in Wireless Sensor Networks', 4th International Conference on Sensor Technologies and Applications, pp.101-107, 2010.
[12] Kolo Jonathan Gana, Li-Minn Ang, Kah Phooi Seng, S.R.S. Prabaharan: 'Performance comparison of data compression algorithms for environmental monitoring wireless sensor networks', International Journal of Computer Applications in Technology, Vol.46 (1), pp.65-75, January 2013.
[13] E.A. Maher, GH. Alia, T. Samar, F. Ghaddar: 'Resource-Efficient Floating-Point Data Compression Using MAS in WSN', International Journal of Ad hoc, sensor & Ubiquitous Computing (IJASUC), Vol.4 (5), October 2013.
[14] Ch.M. Sadler, M. Martonosi: 'Data compression algorithms for energy-constrained devices in delay tolerant networks', In: Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys) 2006.
[15] T.A. Welch: 'A technique for high-performance data compression', Computer, Vol.17 (6), pp.8–19, 1984.
[16] R.S. Pisal: 'Implementation of Data Compression Algorithm for Wireless Sensor Network using K-RLE', International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE), Vol.3 (11), November 2014.
[17] K. Yamini, K.S.N. Raju, K. Miranji: 'Low Power Data Compression Algorithm For Wireless Sensor Networks Using VHDL', International Journal Of Engineering And Computer Science. Vol. 3 (11), November 2014.
[18] D. Salomon, G. Motta: 'Handbook of Data Compression', 5th Edition, Springer-Verlag London Limited, pp.31-33, 2010.
[19] H. Kacem, M. Glaoui, A. Gharsallah: 'Power Saving Solution For WSN Cases Studies based on Interrupt handler versus DMA', In: 12th International Multi-Conference on Systems, Signals & Devices, 2015.
[20] H. Kacem, M. Glaoui, A. Gharsallah: "Enhancing DPM Techniques in Outdoor Industrial WSN Applications", International Journal of Distributed Sensor Networks, Vol.12 (7), July 2016.
[21] I. Ioanna Tsekoura, G. Rebel, P. Glöseköttery, M. Berekovic: 'An evaluation of energy efficient microcontrollers' In: 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014.
[22] S. Nabiha, KH. Zaatouri, W. Fajraoui, T. Ezzeddine: 'New Design of Low Power Consumption Mote in Wireless Sensor Network', International Journal of Computer Science and Electronics Engineering (IJCSEE), Vol.3 (1), 2015.
[23] K. Chapman, "Practical power testing", Reference Design for Spartan-3A FPGA Starter Kit, Feb 2008, Xilinx Ltd.
[24] The Weather Underground website, https://www.wunderground.com/

Marwen ROUKHAMI received the master's degree in Electronic, Electrotechnical and Automatic (EEA) from the Faculty of Sciences of Tunis in 2013. Between 2013 and 2016, he occupies a teaching assistant position at the National Engineering School of Carthage (Tunisia). Currently, he is a PhD student in Electronics at the Faculty of Sciences of Tunis in the laboratory of the energy efficiency and renewable energies (LAPER), Tunis. His research interests include embedded system design and wireless sensor network.

Younes LAHBIB holds his Engineer diploma in electrical and electronic engineering from ENIM-Tunisia (Ecole Nationale d'Ingénieurs de Monastir-2000), his Master's (Electronic Devices and Materials) and his Ph.D (Physics-Microelectronics) degrees from the Faculty of Sciences-Monastir (2002-2006 respectively). He worked at ST Microelectronics from 2001 to 2006 as R&D engineer preparing his PHD. His research interests included HLS, SystemC SoC modeling and assertions-based verification of hardware systems. Since 2007, he is Assistant Professor in microelectronics and embedded systems at the University of Carthage (Department of electrical engineering and computer sciences, Ecole Nationale d'Ingénieurs de Carthage). He is also member of the EμE laboratory working on embedded cryptographic systems, DSP and FPGA accelerations methods.

Abdelkader MAMI received his Dissertation H.D.R (Enabling to Direct Research) from the University of Lille, France, in 2003. He is currently a Professor at the Faculty of Sciences of Tunis (FST) and a member of scientific advisor in the Faculty of Science of Tunis (Tunisia). He is president of the thesis committee of electronics at the Faculty of Sciences of Tunis and the Director of the laboratory of the energy efficiency and renewable energies (LAPER), Tunis.