Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105(Online)
Volume **3**, Number **2**, May **2018**

An Improving Data Compression Capability in Sensor Node to Support SensorML-Compatible for Internet-of-Things

Trong-The Nguyen, Thi-Kien Dao

Department of Information Technology
Haiphong Private University
No. 35 Danlap Rd.,Hai-Phong, Vietnam
vnthe@hpu.edu.vn, jvnkien@gmail.com

Jeng-Shyang Pan[1,2]

[1]Fujian Provincial Key Laboratory of Big Data Mining and Applications
[2]College of Information Science and Engineering
Fujian University of Technology, Fujian, China
jengshyangpan@gmail.com

Mong-Fong Horng, Chin-Shiuh Shieh

Department of Electronics Engineering
National Kaohsiung University of Applied Sciences, Taiwan
mfhorng@cc.kuas.edu.tw, csshieh@cc.kuas.edu.tw

Received November 2017; revised February 2018

Abstract. *Limited bandwidth and lacked compatibility information of the sensor nodes in Wireless Sensor Network (WSN) are critical issues in implementing WSN applications. This paper proposes a method of improvement to data compression capability to support SensorML interface for information exchange in the sensor nodes. The delivery of the packets with XML format of all nodes in WSN could cause the traffic load increases. This paper proposes a method of improvement of data compression capability for exchanging data in the sensorML for the Internet of Things (IoT). The delivery of XML formatted packets of all nodes in Wireless sensor networks (WSN) could cause the traffic load increases. The proposed method suggests data compression condensing the packets in the node and analyzes the relationship between the parameters and the performance. The settings of sliding window size and comparing length are factors to affect the performance of network traffic load significantly. The experimental results show that, with a format of 256 bytes and LZSS compression, the transmission performance is improved up to 48.8%, in comparison with the original approach. Besides, the proposed method can deliver up to 204% of information than the other works.*
Keywords: Internet of Things, SensorML, data compression, sensor nodes, LZSS.

1. **Introduction.** Cisco IBSG predicts there will be 26 billion devices connected to the Internet by 2018 and 40 billion by 2020[1]. Gartner defined the Internet of Things (IoT) is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment [2]. The goal of the IoT is to expect devices can be connected to communication and operation together over the network. One step further of IoT is to integrate data from different resources e.g.,

machines, systems, media in real time that allow intelligently interact such as humans with devices, devices with devices and devices back to humans upon the ultimate social media collaboration of man and machine. Sensor node can be found in applications of wireless sensor network (WSN) that is a typical form of the IoT [3]. Resource constraints such as memory and energy are important issues when implementing WSN, especially with having various kinds of sensors, in which acquired data can be a significant amount of virtual database.

WSN applications such as the environmental monitoring systems utilize several sensor nodes that have various kinds of sensors. These applications might need a large memory space for collecting all acquired data from equipped sensors[4],[5],[26]. Transmission of data over the wireless channel at node level in a network consumes higher energy than data processing within the node. The communication consumes the most substantial energy of a sensor node. Therefore, it is not appropriate to transmit data directly in the networks; on the contrary, minimization by applying data compression algorithm in sensor nodes before sending data over the air is one of the essential strategies for energy-efficient of sensor nets [6].The traditional data format of sensors was insufficient interoperability and could not be compatible with each other. The exchanging standards between objects have not been able to achieve the interoperability and compatibility so far. For example, the disaster-monitoring applications sensing data could not be used in other types of applications with different monitoring devices that need a lot of workforces to process [6][7]. Although the closed data format could reduce transmission bandwidth requirements, it also significantly decreased the readability of the information.

Awareness of interoperability between devices in advance for the IoT, the interoperability and compatibility have been developed for sensor information exchange standards by International organization known as the Open Geospatial Consortium (OGC) [8]. This organization OGC currently has included about 480 members, in which are accounted for 41 % of them are commercial organizations, the leading members namely NASA, Google, Oracle, etc. They continuously update their formulation improvements for Sensor Web Enablement (SWE) specifications to be able to carry out the heterogeneity of sensor integration, which including sensor information exchange, pre-processing, post-processing and computing mode functions. These sensing data from various sensors become more popular and available so that people can save them through the network, applications or platforms of geographic information and services in the world for further use [9]. SWE introduced a standard for the Sensor Model Language (SensorML) in sensing data observations and processing data. Definition of XML syntax is a standard way to communicate for geographic information and data processing services of detecting components of systems [10].

How to take advantage of the less amount of data transfer to meet the various sensor incompatibility and interoperability. This paper considers a new design of sensor node with data compression capability to provide a high-efficiency speed rate and more space of store for the data message. The models can make the IoT not only for sensor nodes can sense data, but also improve the efficiency of the transmission network bandwidth. In the proposed method, the lossless data compression algorithm is applied to reduce the amount of information and to occupy less bandwidth for data aggregation processing in while data transceiver that leads the IoT increasing more efficiency of bandwidth usage. Next sections present the details of the methodology and its related work.

2. **Related Work.** Applications of the IoT with relevant sensor networks have been growing significantly. The information exchange standards have received more attention from researchers [6][10]. The question is how to integrate the information from offering

different applications for exchanging between objects in the interoperability and compatibility that will be a significant challenge for the IoT technology and the information processing standards of the sensor network. For solving this critical issue, the integration of sensor networks with service-oriented architectures in Giovanni Aloisio et al. [6], requires explicit representations of sensor information and interfaces. A strong candidate language for modeling sensor information suggested is SensorML[8].

The main reason for using SensorML for sensor information exchange standards is to describe exchanging information details before or after processing to facilitate the exchange and calculation that in term of being better to improve interoperability. The SensorML, as opposed to other sensor modeling languages, supports a specification of a process model associated with a sensor system, Van Zyl et al. [11] also confirmed the SensorML in sensor networks would be more appropriate information exchange standards.

Also, integration SensorML into applications has been gradually increasing, such as applications of the sensing of the satellite measurements [12], the health care [13] and the electrical systems [6]. The drawbacks of these applications can store and exchange effects in the more massive data amount and a variant of the sensor nodes. Therefore, a design of the sensor node with itself is the ability to compress data is necessary needed. On the other hand, data compression issue is attracted as a mission-critical concern for solving the data storage and transmission efficiency.

The several conventional data compression methods [14][15] is reviewed as follows. Huffman coding uses a variable length code for each of the elements within the information [15]. It involves typically analyzing the data to determine the probability of parts of the data. The most likely items are coded with a few bits and the least likely coded with a higher number of bits. The Huffman coded values are then read from left to right, and the bits are listed from right to left. The significant advantage of Huffman coding is that, although each character is coded with a different number of bits, the receiver will automatically determine the style whatever their order. When transmitting or storing Huffman encrypted data, the coding table needs to be stored with the data. It is a proper compression technique, but it does not take into account higher order associations between characters.

Adaptive Huffman coding [16] uses defined word schemes which determine the mapping from source messages to codewords based upon a running estimate of the source message probabilities. The code is adaptive and changing to remain optimal for the current forecast. In this way, the adaptive Huffman codes respond to locality, and the encoder thus learns the characteristics of the source data. The decoder must then determine along with the encoder by continually updating the Huffman tree to stay in synchronization with the encoder.

The second advantage of adaptive Huffman coding is that it only requires a single pass over the data. In many cases, the adaptive Huffman method gives a better performance, regarding some bits transmitted, than static Huffman coding. This does not contradict the optimality of the static method as the static method is optimal only overall methods, which assumes a time-invariant mapping. The performance of the adaptive techniques can also be worse than that of the static method.

LZ77 coding [16] was developed the adaptive dictionary data compression techniques which took into account repetition in phases, words or parts of words. These repeated components can either be text or binary. A flag is typically used to identify coded and un-encoded portions. The encoded sequence could be modified with the flag sequence "$\#m\#n$" where m represents the number of characters to trace back to find the character sequence and n the number of replaced characters. Usually, a long chain of text has many repeated words and phrases, such as 'and', 'there', and so on. If the short sequences

replaced with codes that were longer than the actual series itself, it could cause longer files.

The Variable-length-code LZW uses a variation of the LZW algorithm where variable-length codes are used to replace patterns detected in the original data. It uses a dictionary constructed from the patterns encountered in the original data. Each new pattern is entered into it, and its indexed address is used to replace it in the compressed stream. The transmitter and receiver maintain the same dictionary. LZ compression substitutes the detected repeated patterns with references to a dictionary. Unfortunately, the larger the dictionary, the higher the number of bits that are necessary for the recommendations; the optimal size of the lexicon also varies for different types of data; the more variable the data, the smaller the optimal size of the directory.

For multimedia data (audio, video, and still images) is taking up too much storage space and high bandwidth transmission requirements, a significant reduction in the amount of data is needed. The approaches to compressing in frequent use are lossy compression, especially in applications such as streaming media and internet telephony such as H.264/MPEG-4 [17]. By contrast, lossless compression is required for text and data files, such as bank records and text articles. In many cases, it is advantageous to make a master lossless file that can then be used to produce compressed files for different purposes.

For example, a multi-megabyte data can be used at full size to generate a full-page advertisement in a glossy magazine, and a 10-kilobyte lossy copy can be made for a small image on a web page. To store data efficiently with the limited memory space, as well as to reduce the flow of network traffic over time, the data compression methods for captured data compression will reduce not only data storage space requirement, but also improve memory capacity and reduce the efficiency of the transmission time. The architecture in the context of medical sensors in compressed sensing algorithms for data compression in wireless sensors was introduced in [14] to address the energy and telemetry bandwidth constraints common to wireless sensor nodes. The drawback of this system is data exchange format does not conform to the SensorML standards, therefore; it cannot comply with the interoperability of the IoT.

This paper focuses on an approach to designing and implementing sensor nodes with data compression capabilities as the future IoT for sensor nodes. The sensor nodes are not only as data acquisition but also provide data compression function. For compatibility with SensorML standard of the IoT for sensor nodes, the approach should be able to meet a large of amount of sensing data storage and use bandwidth efficiently in the sensor network. Strategy of compressing sensing data before transmitting data is applied as a solution for this design. Moreover, the sensor nodes in the IoT requires of transmitting sensing information accurately, whether, during transmission processing, the receiver gets losing data in comparing the original data or losing its meaning of original data, this data could not be used. Therefore, we apply the lossless compression scheme in the design. In next section will present details this design.

3. **Sensor Node with Data Compression Capability for SensorML.**

3.1. **Sensor node System architecture.** In this section, we present framework of modeling and design construction of a sensor node for IoT as general implications of the integrating circuit analysis functions. In modern electronic systems, hardware, software and firmware interconnect in the design process, in which have some built-in assumptions for modeling of architecture system that has produced significant positive things [4][6][18]. Formal rules on the design of software have applied to the development of hardware systems. System development usually is a multi-stage process. It is iterative and a not

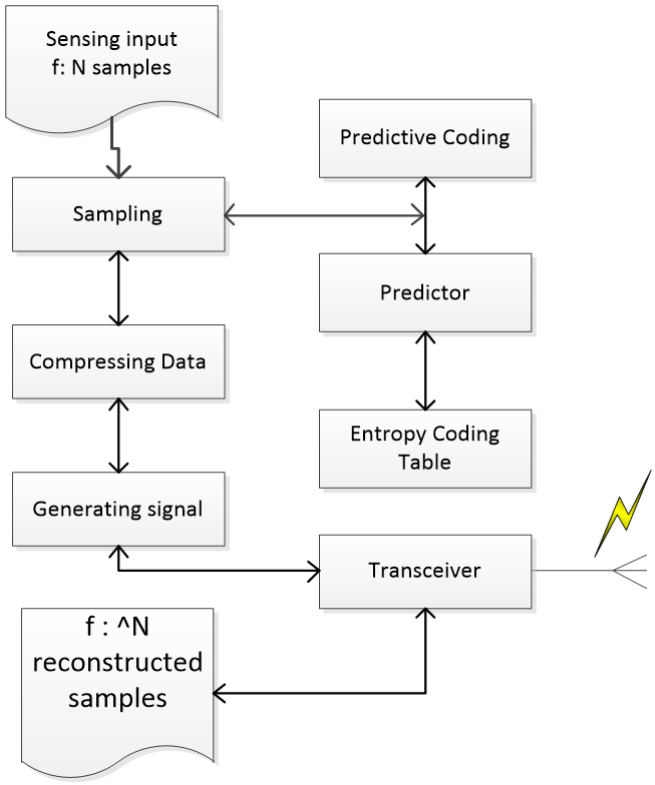

FIGURE 1. Implementing function modules of a design of Sensor Node

FIGURE 2. Sensor Nodes Data packet format

sequential process and feedback from any part updates earlier stages [19]. We started the phase of identification function system, a need for the system is defined to help a design team produce a designing specification about the functionality of the complete system. For example, assumed that the sensor nodes components perform ideally such that the sampling module, compressing module and transceivers module. The inputs of a presented modeling framework consist only terms of technical parameters, definition functions and system specifications in applying to a designing sensor node. One obvious extension of the model is to analyze the compression algorithm tradeoffs for SinsorML which are identical to functioning modules of the system has been noted in Figure 1.

Figure 1 shows design module of the tradeoff for sensorML to support underlying structure IoT for sensor nets. The input specification is defined as a sampling function of the system to sense data as the original data; the output specification is identified as an optimal a function of the system regarding data compression and improve throughput. There are some necessary described function modules as follows. First, the sampling module is responsible for building packet format and reading packet headers. This module allows for modifying header structures, packet types and the method for actually making the header. It defines explicitly methods for building configuration and data headers, as well as reading, received headers.

Figure 2 describes data header in sensor node of its MAC packet format to facilitate encoding sensorML. Second, the compressing data module provides methods for being ability compression data before transmission. This major component consists of the group of modules including predictive coding, predictor, and entropy coding table (see Fig. 1). These modules allow for new compression algorithms to be implemented and plugged into sensor node with minimal modification. Moreover, the sensor node design allows for changing implementations of predictors and entropy coding tables with little adjustment. With a standardized interface for compression algorithms, this module allows a new compression algorithm merge with the sensor node. A new module supports interface and modify the compression module to point a specific algorithm.

The predictive coding module implements the defining a compression radius and error bound as well as choosing and executing sync operations and lossless compression. The predictor module supplies a module with a well-defined interface for predictors. The coding module does the same as the predictor module for entropy coding. This allows for new predictors and entropy coding techniques to be implemented to match our interfaces and merely switch in and out for easy testing as well as customized algorithms to optimally fit the sensor data characteristics at given tasks. Finally, the generating signal module and transceiver module are responsible for everything related to streams. It provides methods for building and sending streams as well as passing stream data to other modules that may require these data. These modules provide the interface for initializing the lower level network, sending packets and receiving packets.

3.2. **Software Architecture and Data Compression Algorithms.** As describing resource of the sensor is through SensorML that use mechanism for describing IoT. For example scientific workflows and facilitating wholly distributed workflow descriptions on the web [11]. However, in the specific cases like sensor network whose has the open and distributed environments, sensorML should be extended by trading off to fully capture the requirements as relating to its open and distributed environments. This section, the data compression methods for SensorML sample message formats are considered as the application for a design of sensor node for the Internet of Things. The lossless compression algorithms (Huffman, Adapted Huffman, Arithmetic, and LZSS) [16] use a generated dictionary table for compressing or decompressing correctly. Due to being based on the dictionary table, the data compression ratio can make changes for the better.

Figure 3 shows the LZSS data compression algorithm is better than the other compression algorithms concerning it is more suitable for sensing node data compression. Therefore, this paper recommends and emphasizes on applying LZSS data compression algorithm in the design and to implementation for sensor nodes. Lempel-Ziv-Storer-Szymanski, otherwise known as LZSS that is a derivative of the LZ77 lossless data compression algorithm, which is a dictionary encoding technique. It attempts to replace a string of symbols concerning a dictionary location of the same string. For example, Input stream for encoding:

Pos: 1 2 3 4 5 6 7 8 9 10 11
Char:A A B B C B B A A B C

The LZSS data compression presented as the mandatory inclusion of the next non-matching symbol into each code-word that could lead to situations. The explicitly coded symbol despite the possibility of the next match as shown in Algorithm 1. For example, in the string of "abbca‖caabb" included the matches. The first match is a reference to "ca" (with the first non-matching symbol being "a"). And the next match then is "bb" while it could have been "abb" if there were no requirement to code the first non-matching symbol explicitly. Its algorithm uses fixed-length code-words consisting of offset (into the

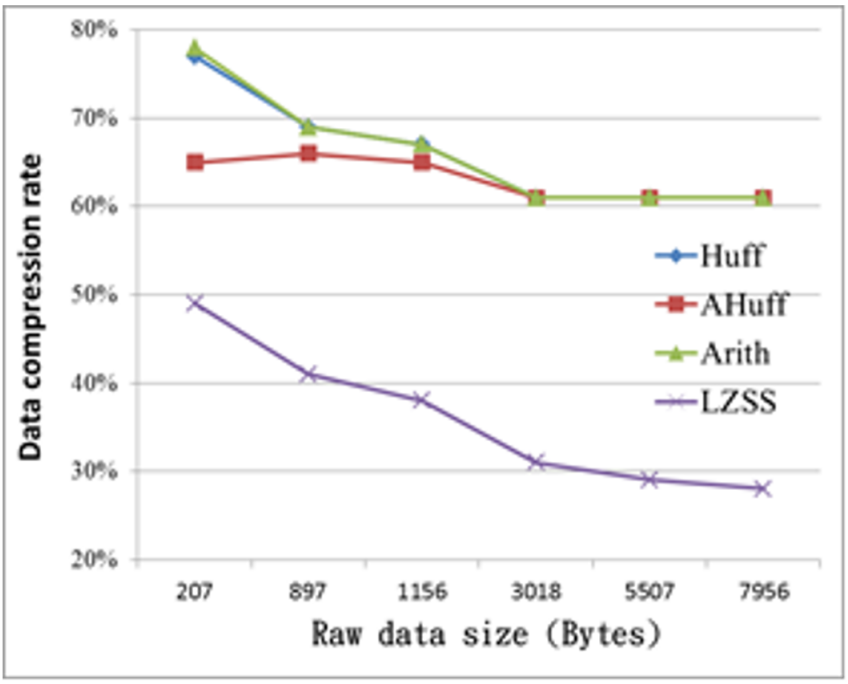

FIGURE 3. Typical efficiency of lossless compression algorithms

search buffer) and length (of the match) to denote references. Only symbols for which no match can be found or where the recommendations would take up more space than the codes for the symbols are still explicitly coded.

Algorithm 1 The LZSS algorithm

Step 1: place the coding position to the beginning of the input stream;
Step 2: find the longest match in the window for the lookahead buffer:
$P :=$ pointer to the match;
$L :=$ length of the match;
Step 3:
if $L >= MINLENGTH$ **then**
Output P and move the coding position L characters forward;
else
Output the first character of the look ahead buffer and move the coding position one character forward;
end if
Step 4:
if There are more characters of the input stream **then**,
go back to Step 2.
end if

Figure 4 shows the process of the encoding scheme in which, each pointer or character added an extra bit to distinguish between them. The output is packed with no unused bits. Implemented LZSS encoders and decoders can be simplified by numbering the input text characters modulo N. The window is an array of N characters. To shift in character number r (modulo N), it is merely necessary to overwrite element r of the variety, which implicitly shifts out character $r - N$ (modulo N). Instead of an offset, the first element of an (i, j) pointer can be a position in the array $[0..N - 1]$. It means that i is capable of

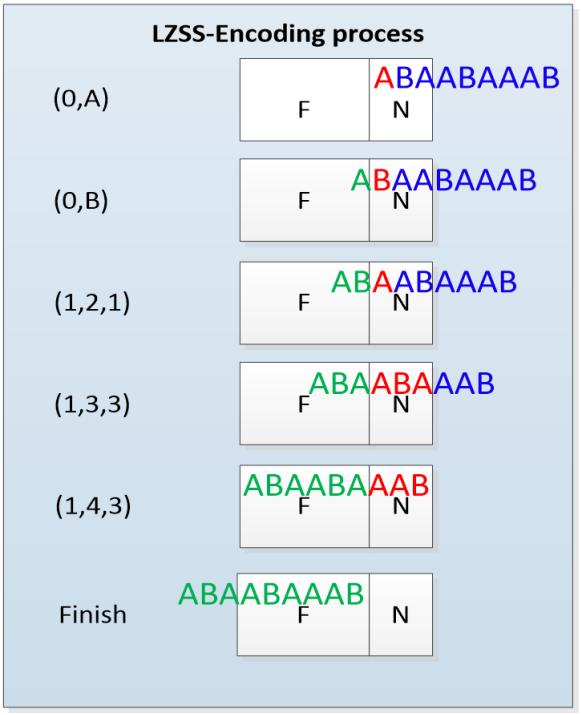

FIGURE 4. A phase of encoding LZSS compression algorithm

indexing substrings which start in the look-ahead buffer. The coding parameters N and F can be specified in terms of the number of bits they are to be coded in, which will be denoted as n and f, respectively. N and F are calculated from following equations:

$$p = |(1 + n + f)/8| \tag{1}$$

$$N = 2 \times n, \tag{2}$$

$$F = 2 \times f + p \tag{3}$$

To code the r^{th} character of a string s under LZSS, a window w of N characters and a look-ahead buffer l of F characters is used as given:

$$w = s \times (r + F - N, r + F - 1) \tag{4}$$

$$l = s \times (r, r + F - 1) \tag{5}$$

where s is a string of n characters, $s = s_1, s_2, ...s_n$, then s_i is written as $s(i)$, and the substring $sisj$ is written as $s(i, j)$. Given two strings s and t of length n, the match between the strings, $M(s, t)$, is defined to be the length of the longest common prefix of s and t. Given a set of q strings of length $n, U = [u_1, u_2, , u_q]$, the longest match, $LM(s, U)$, is a function on a strings and the set of strings U, which returns the ordered pair $(i, M(s, u_i))$, where $M(s, u_i)$ is maximal over all the strings in U. The ordered pair gives the position and size of a longest match for s in U. This need not be unique, but any one of the different possible values may be used, since the main concern is the length of the match. To be consistent with the implementation of the window, a special case is made for numbering the string w. Each character in w is given the same index that it has in the original string, modulo N. This means that $s(j)$ corresponds to $w(j \bmod N)$, provided that $s(j)$ is in the window. Substrings are extracted from the window using wraparound, so if $j > k$ then $w(j, k)$ is evaluated as if a copy of w is concatenated to the end of w. The strings of length F in the window are denoted by $x_j = w(j, j + F - 1), j = r + F - N$. Note that $l = x_r.X$ is the set of all distinct strings of length F in the window, apart from

l, i.e., $X = [x_j : j = r + F - N]$, where there are two substrings $x_i = x_j : ij$, the substring which entered the window most recently is chosen for inclusion in X. To perform LZSS coding, a procedure is needed to evaluate $(g, h) = LM(l, X)$. The first h elements of the look-ahead buffer may then be coded with the pointer (g, h).

For example, if the "$strings = abcabcbacbababcabc$" is being encoded with the parameters $N = 11$ and $F = 4$, and coding is up to character 12, then the window is

Pos: 5 6 7 8 9 10 0 1 2 3 4
Char: b c b a c b a b a b c

and $l = x_1 = babc$, and $x_5 = bcba$, $x_6 = cbac$, $x_7 = bacb$, $x_8 = acba$, $x_9 = cbab$, $x_10 = baba$, $x_0 = abab$. By inspection, the longest match is x_{10}, i.e., $LM(l, X) = (10, 3)$. The straightforward algorithm evaluates *M(l, X_i)* for each member of X. This requires up to N - F evaluations of M to obtain each pointer in the coded form of the input string s.

3.3. **Hardware architecture.** As described the LZSS compression algorithm in subsection B, and this subsection adapted its parameters into a set of the required hardware specifications. Figure 5 shows the initialization and the timer setting for sensing nodes' hardware. The procedure external interrupt is implemented as a design for compression algorithms. The external interrupt checks whether requests for information exchange, if any required data exchange sensing information will be packaged into the SensorML coding, and packed data compression. The sensor node encoder essentially amounts to performing a linear projection from the N-dimensional input, f, to an M-dimensional set of measurements, y, using the matrix, Φ. In the context of data compression, this amounts to transforming every block of N samples of f into M measurements *(s)*. B_f and B_y are the bits needed to represent the dynamic range of each sample in and respectively. Δ is to use a pseudo-random Bernoulli matrix where each entry, $\Phi_{m,n}$ is 1 [19][20]. The signal bandwidth in Hertz is BW_f. Any other choice of a full rank $M \times N$ matrix would result in circuit complexity, data storage, and computation requirements.

The data compression must be split after data partition items will depend on the total length, and the range of the partition after partition will start the data transfer mechanism, through the data transport mechanism to ensure will divide the excellent information entirely fed every data exchange a single chip will go through this process. To achieve the goal of the proposed method of design and implementation for sensor nodes: low power consumption, high-performance computing and full peripheral interface of the sensor node. Hardware core of sensor node for data processing, compressing and decompressing modules of forming the sensor nodes are beneficial for performance validation. The internal function blocks of the experiment are shown in Figure 6.

Analog-to-Digital Converter (ADC): the input signal is first amplified and then digitized by a single ADC sampling at the Nyquist rate, f_s[21]. In which, the sampling frequency of each ADC only needs to be f_s/N where $f_s \geq 2BW_f$ and N is the number of integration samples per compression block. The ADC output is passed to M parallel accumulators that accumulate the incoming sample based on their respective sequence of matrix coefficients $\Phi_i(t)$. The matrix coefficients $\Phi_i(t)$, need to be applied at the Nyquist frequency, f_s of the signal or higher in order to avoid aliasing [19]. For sensor node for Internet of Things application, systems are typified by low sampling frequencies, medium resolutions and small amplitude input signals. The output of each ADC produces one sampling block result, $f[_n]$, so the resolution of the ADC should be equal to the required measurement resolution, B_s. Because the coefficient matrix is a Bernouli random matrix where all elements are 1, so the multifunction could be implemented with XOR gate and the carry in input of the accumulator. The output of the accumulator is then captured every N samples at which time the accumulator is reset. The resulting power of the array

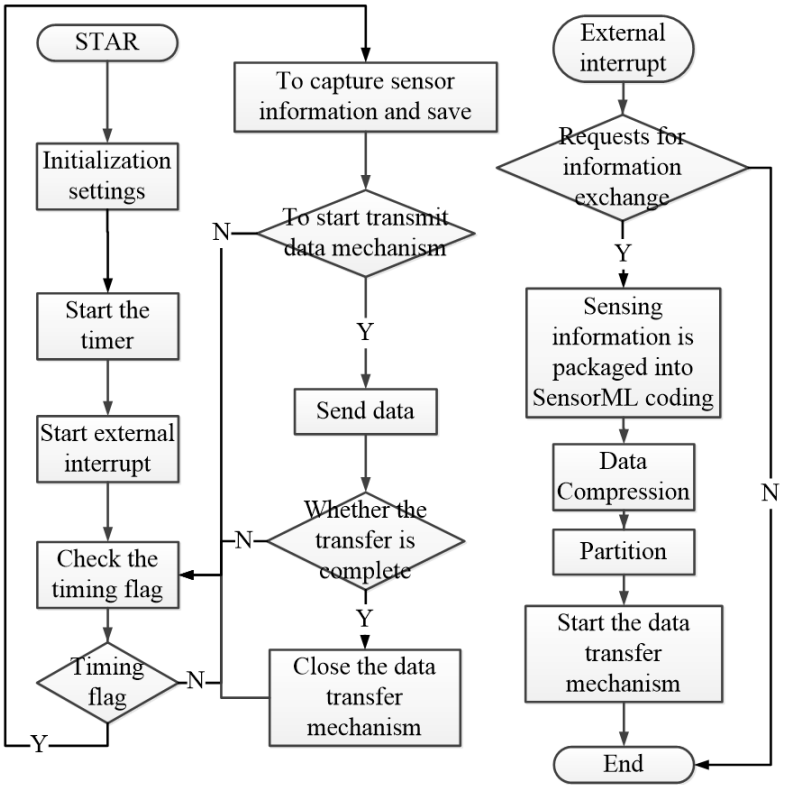

FIGURE 5. The hardware flowchart of sensor node

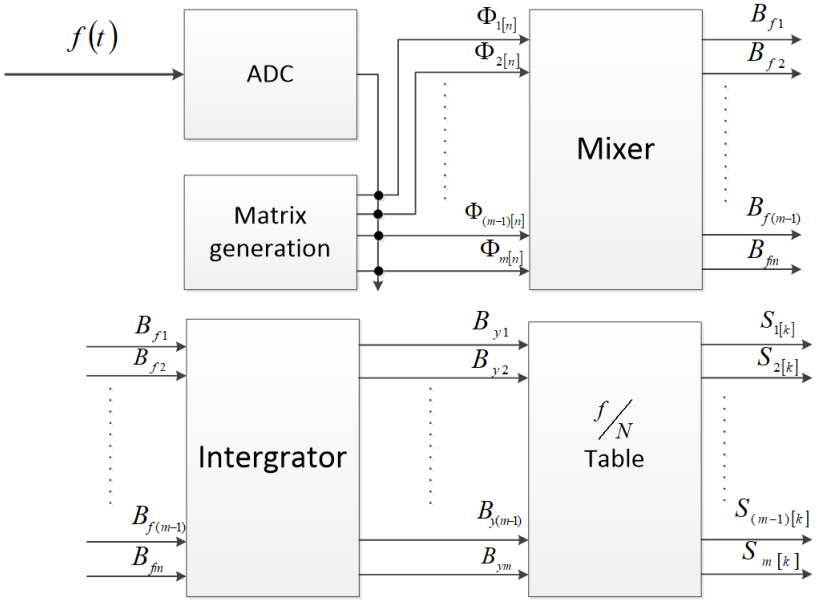

FIGURE 6. Internal design single chip of sensor node for sensorML

of ADCs is then

$$PADC = (M/N) * FOM * 2B_s * f_s \tag{6}$$

where FOM [21] is the figure-of-merit of the ADC, Bs is number bits needed for each sample s, fs sample frequency.

Matrix Generation: the generating the measurement matrix coefficients $\Phi_i(t)$ needs

to approximate a random matrix, one straightforward approach [22] is to use a lookup table or a memory to implement the matrix in running at the Nyquist rate. Since power consumption is important in sensor Internet of Things applications, the matrix generation that requires only two pseudo-random binary sequence (PRBS) generators. The matrix generation consists of the state of one PRBS generator XORd with the output of a second PRBS generator to create the columns of $\Phi(t)$ on a sample by sample basis. The seed and sequence length of each PRBS is programmable to enable the synthesis of a wide variety of pseudo-random matrices. The input is often over sampled; the inner product of a measurement matrix that is derived from a single shifted PRBS sequence and the input will appear correlated.

In mixer: each measurement, $s_i[k]$, requires an accumulator with at least B_s bits of resolution which results in B_f flip-flops and $XORs$, and a B_s -bit adder. In order to model the delay and energy costs associated with these circuits, a logical effort [23] model is adopted to determine the sizing of each gate and the methodology for sizing the adder is similar to [14]. A slightly simplified version of the alpha-power law delay model used in [23] is used to map the normalized delay of the logical effort model to real delay. The logical effort delay of the accumulator is used to scale until the timing constraint is just met, resulting in the following minimum operating. Mixer is described in [21],[23] where it is shown to have a theoretical voltage conversion gain G_c ranging between -3.92 dB and 2.1 dB and a measured noise figure (NF) of 3.8 dB. The impact of the mixer performance is its impact on the specifications for the operational trans-conductance amplifier. For a $G_c = -3dB$ and a $NF = 3.8dB$, the current noise density at the output of the mixer:

$$\overline{i^2} = \overline{i^2_{noise}}.10^{(G_C/2+NF)/10} \tag{7}$$

where i^2 is the noise current density out of the amplifier into the mixer. For a pseudo-random bit sequence of samples, the resulting noise accumulated during an integration window is N times the output noise of a single sample, where the output noise density is filtered by the gain and effective noise bandwidth of an integrator with $1/N_{th}$ the integration period. In integrator, the frequency response is a since pulse where the gain G_I and noise bandwidth BW_N of the integrator can be expressed as

$$G_I^2 BW_N = \int_0^\infty |H_i(f)|^2 df = \underbrace{\left(\frac{N}{f_s C_L}\right)^2}_{gain} \underbrace{\left(\frac{f_s}{2N}\right)}_{bandwidth} \tag{8}$$

where $H_i(f)$ is the transfer function of the integrator, N/f_s is the integration period. The outputs of the integrator are sampled as a string of n characters, $s = s_1, s_2, ..s_n$. and samples B_s are collected at each path.

Since the purpose of the design sensor node encoder sensorML is for being ability of data compression, with LZSS compression algorithm, compression block lengths between 100 to 1000 samples require roughly 11-17 measurements $s_i[k]$ per significant term to reconstruct 3-4 significant terms per block requires the range of specifications for the system.

4. Experimental Results and Analysis.

4.1. The experimental results.

Four results of the proposed method with data compression capabilities can be obtained through the experimental evaluations included the compressing rate, throughput, flexible adaption, and easy use.

The first, *the compressing rate* is a compression ratio with a format of 252 bytes and LZSS compression, the transmission performance is improved up to 48.8%, in comparison with the original approach with raw data. Figure 6 and Figure 7 show the comparative

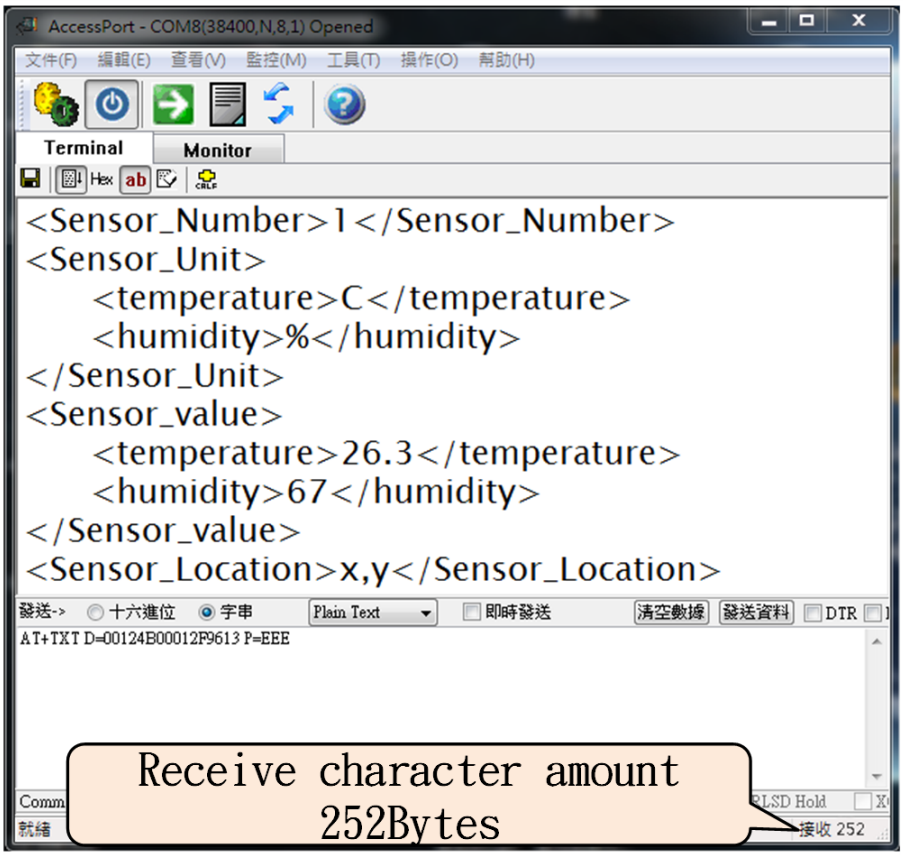

FIGURE 7. Original data exchange through sensor node with hardware architectures based on the IEEE 802.15.4/ZigBee protocol [24], which uses the W3C XML as the encoding for the exchange of information

performance between data compressed and uncompressed data on a new sensor node and sensor node with hardware architectures based on the IEEE 802.15.4/ZigBee protocol with the same data format of data packets 252 bytes.

Figure 7 shows a sensor node with hardware architectures based on the IEEE 802.15.4 /ZigBee protocol [24]. This sensor node uses the W3C XML schema definition language as the encoding in the binding of the data model defined in IEEE Std., 1484.11.1 TM-2004 [25], which allows for interoperability and the exchange of data-model instances between various systems. The figure shows the sensor node receives amount 252 bytes packet format of original data for the exchange of information.

Figure 8 shows a newly proposed sensor node uses data model defined in IEEE Std 1484.11.1TM-2004, as the encoding, with the same amount 252 bytes packet format of original data. Apparently, the proposed sensor node with data compression capabilities shows amount only receives 123 bytes compression data for the exchange of information. This receiving amount one occupied only of a haft of non-compression data sensor node.

The second, *the node throughput* is improved significantly. The figure 8 shows the comparison of amount delivering of cumulative information between data compression sensor node and non-data compression sensor node. In which, the solid line of data compression increased significantly, it got over 1409 kb in the 23th hours, whereas the dotted line of the non-data compression sensor node is not only the highest located approximately 700 kb at same time with solid line, this value only equal a half the solid line of data compression. This means the evaluation demonstrates that new proposed sensor node with applied compression algorithm can improve the compressed data streaming throughput

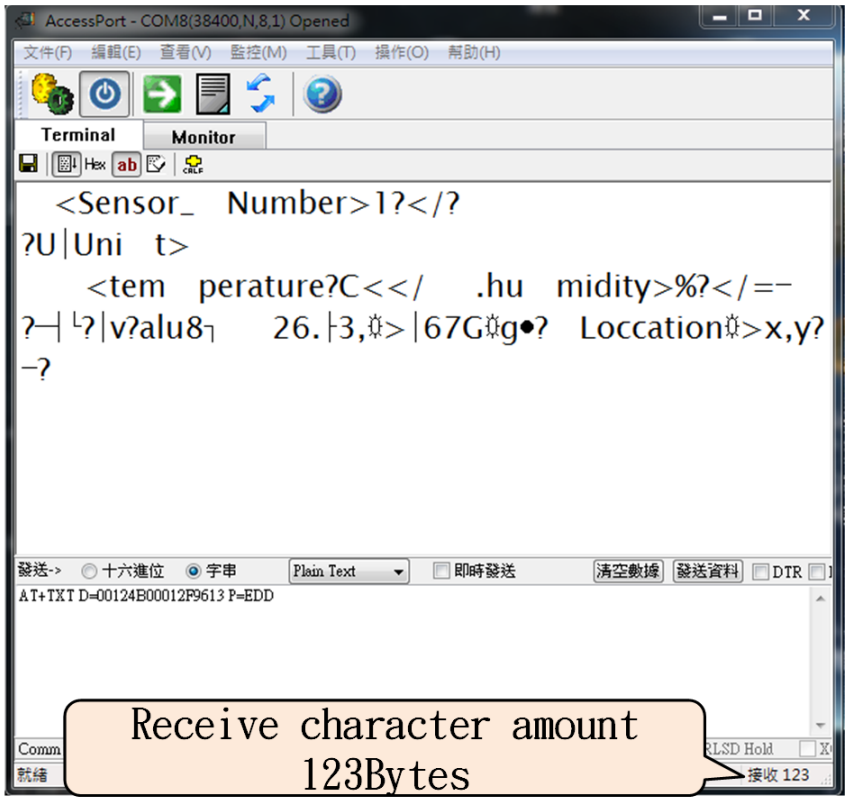

FIGURE 8. A new proposed sensor node as the for compressed data exchange

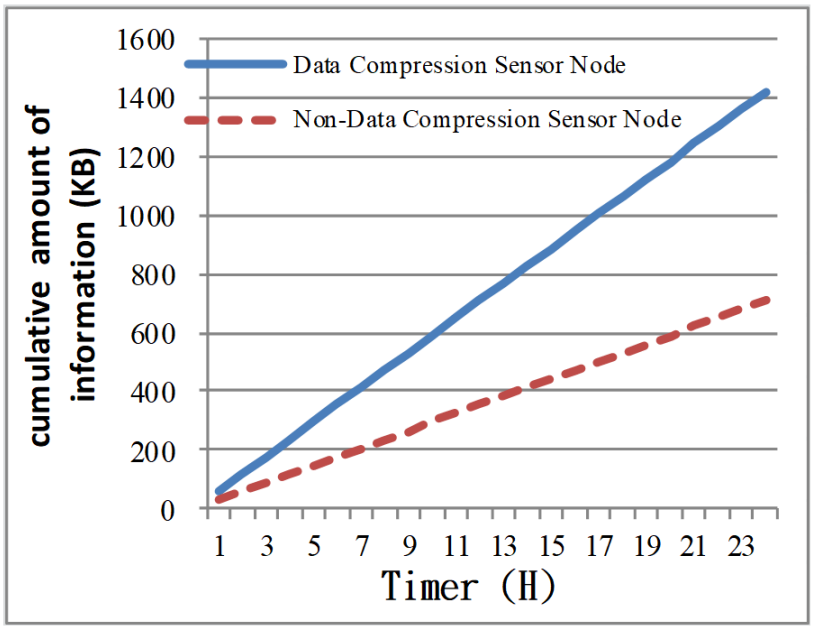

FIGURE 9. The comparison amount delivering of cumulative information between data compression sensor node and non-data compression sensor node

by over 204 % in comparison to use the original data transmission of the amount of information in represented by the same condition for transferring traffic, and with good the distances (network latencies) between interacting nodes.

In addition, it is assumed that the amount of information accumulated during the day, with the duty cycle working of sensor nodes is 30 seconds; the communication capabilities of the new proposed sensor node is better efficiency of transmission than the transmission of original data accumulate information on the same conditions. The third, *flexible adapted architecture* of the proposed architecture of sensor node has benefit of basic qualities: (i) sensor node supports SensorML standard self-descriptions and processes in the adopted SensorML model language, so as the entire network can be sufficiently described. Because, the formal SensorML for sensors and processes inside network complexes and rather heavyweight of describing in a dynamic manner. (ii) Openness: sensor node supports in a plug and play mode, so it can be added or removed from the network. The provided that, sensor node is self-described via a valid SensorML document, preserving this way the unobtrusive operation of the network. (iii) Sensor nodes can be reconfigurable in run-time, in the sense of adapting their operation logic. For example, the sampling frequency of a data sensing parameter of humidity and temperature may have to change, depending on certain personalized criteria, personalized thresholds for parameters of a physiological phenomenon should also be reconfigurable to meet requirement by using SensorML.

TABLE 1. The comparative length in a set of relationship between the compression of ratio (n = 18)

SW	Original Data (bytes)					
	207	**897**	**1156**	**3018**	**5507**	**7956**
32	105.3%	103.9%	104.2%	96.3%	97.7%	98.8%
64	53.6%	81.4%	80.8%	72.2%	76.1%	77.4%
128	53.1%	65.9%	64.0%	58.5%	62.4%	62.3%
256	49.3%	43.5%	45.2%	43.9%	42.9%	44.3%
512	49.3%	43.0%	39.5%	36.7%	37.8%	37.8%
1024	49.3%	41.4%	37.9%	33.1%	33.0%	33.2%
2048	49.3%	41.4%	37.9%	31.4%	30.4%	30.9%
4096	49.3%	41.4%	37.9%	31.4%	29.5%	28.5%

The finally, this new sensor node is flexible using in different sensor net applications with two created commands for requirement of formatting data packet and exchanging information e.g. "$AT + TXTD = MACP = Command$". The command for formatting data packet acts on the sensing nodes MAC address to require sensor node compression data of information exchange such as a command "$AT + TXT\ D = MACP = EDD$", if it needs the sensor nodes use original data for exchange of information, a command such as "$AT + TXTD = MACP = EEE$"

4.2. **Data compression performance analysis.** The experiment evaluation provides not only remarkable compression ratios, but the new sensor can also signicantly improve its sensor net throughput by reducing retransmissions in noisy wireless sensor net. The reducing data retransmissions and the total lower layer packet overhead altogether make substantial savings in overall transmitted bytes. As the experimental results are mentioned above, due to applying the compression algorithm LZSS to sensor node in both hardware and software design, it is based on standard sensorML tradeoff for senor net applications. The core compression module in Fig.6 includes a mixer for encoding/decoding

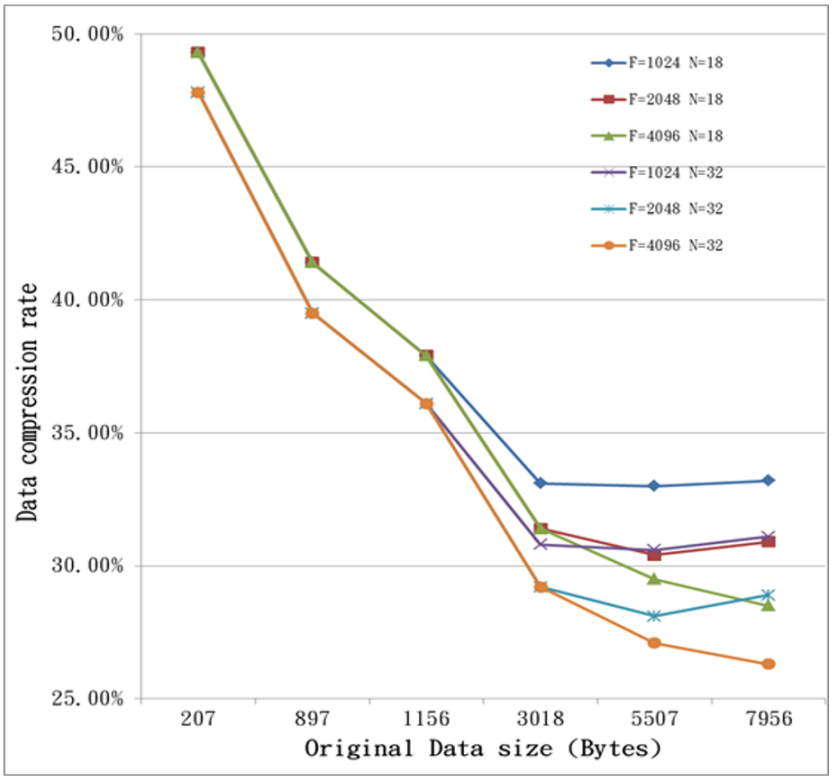

FIGURE 10. Compression performance diagrams for two parameters in deferential setting

to a window and dictionary size generator by matrix generation. The mixer and integrator act applicably to a compression algorithm.

The compression algorithm LZSS could be achieved compression by replacing common strings with pointers to previously seen occurrences of the string. In other words, LZSS works by trying to replace the current string being encoded with an offset to a matching, previously seen, string and its length. The compression algorithm uses a buffer of the preceding several bytes (called the sliding window) to look for the next several bytes (called the comparative length). The size of the sliding widow and the comparative length determine the size of the generated codes, the maximum compression possible. At each step of the compression process, the number of bits required by an optimum parsing of the input ending at the current position is known. There are two parameters are considered to have strong influence on the compression ratio and processing quality (as throughput).

Overall, the proportion of the data compression rate of two kinds of the comparative length (including 6 lines) decreased significantly over of increasing number of packet format in original data byte 207 bytes, 807 bytes, 1156 bytes, 3018 bytes, 5507 bytes and 7956 bytes. At the data format packet of 207 bytes, the percentage of the data compression rate gets highest made at peak is 49% for comparative length N =18, and 48 % for comparative length N = 32. In a while range of original data packet format from 207 bytes to 3018 bytes, the size of slide window F is not affected much in compression rate. However, for the size of 3018 bytes to 7856 bytes of original data, the value of F impacted in compression rate; the lowest of the data compression rate on size of 7956 bytes data raw is only 36 % with N= 18, and 26 % with N=32 respectively.

5. **Conclusions.** In this paper, we presented a novel method of the improvement to data compression capability to support SensorML interface for information exchange in

TABLE 2. A set of relationship between the compression of the ratio with the comparative length (n = 32)

SW	Original Data (bytes)					
	207	**897**	**1156**	**3018**	**5507**	**7956**
32	96.1%	108.8%	109.6%	111.4%	111.9%	112.1%
64	73.4%	88.0%	86.5%	77.8%	82.1%	83.1%
128	51.7%	67.2%	65.8%	59.5%	63.8%	63.9%
256	47.8%	41.6%	43.7%	46.1%	43.1%	44.2%
512	47.8%	41.1%	37.8%	34.9%	35.7%	36.0%
1024	47.8%	39.5%	36.1%	30.8%	30.6%	31.1%
2048	47.8%	39.5%	36.1%	29.2%	28.1%	28.9%
4096	47.8%	39.5%	36.1%	29.2%	27.1%	26.3%

the sensor nodes. Data compression capability implemented before transmitting reduces not only the data retransmissions but also less total byte transmitted in lower layer packet overhead.

The experimental result offers the remarkable compression ratios and improves the transmission performance is up to 48.8%, in comparison with the original approach with raw data. Besides, the proposed method can deliver up to 204

REFERENCES

[1] D.Evans, The Internet of Things - How the Next Evolution of the Internet is Changing Everything, *CISCO white Pap.,* no. April, pp. 111, 2011.

[2] J.Rivera andR.Van derMuelen, Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020, *Gartner*, 2013. .

[3] M. U.Farooq andM.Waseem, A Review on Internet of Things (IoT), *Int. J. Comput. Appl. (0975 8887)* , vol. 113, no. 1, pp. 17, 2015.

[4] M. A.Razzaque, C.Bleakley, and S.Dobson, Compression in wireless sensor networks: A survey and comparative evaluation, *ACM Trans. Sens. Networks*, vol. 10, no. 1, pp. 544, 2013.

[5] T.-T.Nguyen, T.-K.Dao, M.-F.Horng, andC.-S.Shieh, An Energy-based Cluster Head Selection Algorithm to Support Long-lifetime in Wireless Sensor Networks, *J. Netw. Intell.,* vol. 1, no. 1, pp. 2337, 2016.

[6] D. Meana-Llorian, C. G. Garcia, B. P. G-Bustelo, N. Garcia-Fernndez, . SenseQ: Replying questions of Social Networks users by using a Wireless Sensor Network based on sensor relationships, *Journal of Data Sci. and Pattern Recognit,* vol. 1, no. 1, pp. 1-12, 2017

[7] S.Li, L.DaXu, andX.Wang, Compressed sensing signal and data acquisition in wireless sensor networks and internet of things, *IEEE Trans. Ind. Informatics,* vol. 9, no. 4, pp. 21772186, 2013.

[8] M.Botts, Sensor Model Language (SensorML) v2.0, *OGC Implementation Specification,* 2013. .

[9] N.Dahal, S. S.Durbha, R. L.King, and N. H.Younan, Geo-enabled synchrophasor data exchange framework based on Sensor Web, *in IEEE Power and Energy Society General Meeting,* 2012.

[10] R.Liscano andK.Kazemi, Integration of component-based frameworks with sensor modelling languages for the sensor web, *in 2010 IEEE Globecom Workshops, GC10, 2010* , pp. 235240.

[11] T.VanZyl and A.Vahed, Using sensorML to describe scientific workflows in distributed web service environments, *in International Geoscience and Remote Sensing Symposium (IGARSS)* , 2009, vol. 5.

[12] N.Chen andC.Hu, A sharable and interoperable meta-model for atmospheric satellite sensors and observations, *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.,* vol. 5, no. 5, pp. 15191530, 2012.

[13] A.Triantafyllidis, V.Koutkias, I.Chouvarda, and N.Maglaveras, An open and reconfigurable Wireless Sensor Network for pervasive health monitoring, *in Proceedings of the 2nd International Conference on Pervasive Computing Technologies for Healthcare 2008, PervasiveHealth, 2008*, pp. 112115.

[14] F.Chen, A. P.Chandrakasan, andV. M.Stojanovic, Design and Analysis of a Hardware-Efficient Compressed Sensing Architecture for Data Compression in Wireless Sensors, *IEEE J. Solid-State Circuits*, vol. 47, no. 3, pp. 744756, 2012.

[15] J.Ziv andA.Lempel, A Universal Algorithm for Sequential Data Compression, *IEEE Trans. Inf. Theory,* vol. 23, no. 3, pp. 337343, 1977.

[16] G. G.Langdon, An Introduction to Arithmetic Coding, *IBM J. Res. Dev.* , vol. 28, no. 2, pp. 135149, 1984.

[17] K. R. Vijayanagar and J. Kim, Real-time low-bitrate multimedia communication for smart spaces and wireless sensor networks, *IET Commun.,* vol. 5, no. 17, pp. 24822490, 2011.

[18] N.Giannopoulos, C.Goumopoulos, and a.Kameas, Design Guidelines for Building a Wireless Sensor Network for Environmental Monitoring, *2009 13th Panhellenic Conf. Informatics*, pp. 26, 2009.

[19] X.Chen, Z.Yu, S.Hoyos, B. M.Sadler, and J.Silva-Martinez, A sub-nyquist rate sampling receiver exploiting compressive sensing, *IEEE Trans. Circuits Syst. I Regul. Pap.,* vol. 58, no. 3, pp. 507520, 2011.

[20] J. N. Laska, S. Kirolos, M. F. Duarte, T. S.Ragheb, R. G.Baraniuk, and Y.Massoud, Theory and Implementation of an Analog-to-Information Converter using Random Demodulation, *in 2007 IEEE International Symposium on Circuits and Systems,* 2007, pp. 19591962.

[21] B.Murmann, A/D converter trends: Power dissipation, scaling and digitally assisted architectures, *in Proceedings of the Custom Integrated Circuits Conference*, 2008, pp. 105112.

[22] M. S. J.Steyaert, W. M. C.Sansen, and C.Zhongyuan, A micropower low-noise monolithic instrumentation amplifier for medical purposes, *IEEE J. Solid-State Circuits,* vol. 22, no. 6, pp. 11631168, 1987.

[23] R. W.Brodersen, M. A.Horowitz, D.Markovic, B.Nikolic, and V.Stojanovic, Methods for true power minimization, *IEEE/ACM Int. Conf. Comput. Aided Des. 2002. ICCAD 2002.*, pp. 3542, 2002.

[24] IMS, IMS Meta-data Best Practice Guide for IEEE 1484.12.1-2002 Standard for Learning Object Metadata, *IMS Global Learning Consortium.* pp. 132, 2006.

[25] T.Richards andA.Barr, Catalyzing connected learning through standards, *in Proceedings - 2011 IEEE Global Humanitarian Technology Conference, GHTC 2011*, pp. 503505, 2011.

[26] T. T. Nguyen, J. S. Pan, T. K. Dao, S. C. Chu, Load balancing for mitigating hotspot problem in wireless sensor network based on enhanced diversity pollen. *J. Information Telecommunication,* vol.2, no. 1, pp. 91-106, 2018