IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

MIHBS: A Mobile Interface of High Bandwidth for Wireless Sensor Networks

LIANG SUN, LEI WANG, JIAN FANG, JIALIN LIU, CAN MA.
[1]Dalian University of Technology, China

Corresponding author: Lei Wang (e-mail: lei.wang@dlut.edu.cn).

This work is supported by “the National Natural Science Foundation of China" with No. 61733002 , “the Fundamental Research Funds for the Central Universities” with No. DUT17LAB16, No. DUT2017TB02. This work is also supported by Tianjin Key Laboratory of Advanced Networking (TANK), School of Computer Science and Technology, Tianjin University, Tianjin China, 300350.

ABSTRACT In modern days, wireless sensor networks and smart phones have been widely used in various application domains including healthcare, environment, and intelligent building monitoring. Although the existing smart phones, such as iPhone, SAMSUNG, and Lumia, have in-built sensors of location, camera and accelerometer, their sensing capabilities are limited. Besides, the transmission rate of ZigBee can reach 250kbps, which is much higher than that of smart phones' headset port. It is necessary to keep the balance between them. In this paper, we propose an enhanced Lempel-Ziv-Welch coding scheme which is introduced to help transmit data faster, called LZW-Huffmam. Further, we apply it in a self-designed mobile extension, MIHBS (Mobile Interface of High Bandwidth for Sensor network), which is a universal interface that can transfer data to and from sensors using standard headset port of high bandwidth. LZW-Huffman can help increase the bandwidth of MIHBS by 52%.

INDEX TERMS Computer Peripherals, Mobile Communication, Wireless Sensor Networks.

I. INTRODUCTION

With the development of wireless communication, sensor technology and high-bandwidth embedded system, wireless sensor networks (WSNs) [11], [27] has been widely applied in various fields. Nowadays, WSN access is still relatively professional and not approachable to populace. Consumers should freely access WSN via their own portable devices instead of purchasing special peripherals. However, there are two main problems to achieve this. On one hand, it is necessary to find an open, standardized and pervasive personal interface to communicate with WSN. On the other hand, the bandwidth of ZigBee can reach 250 Kbps. How to design a high bandwidth communication scheme to keep balance, is also our main work.

Nowadays, mobile phone is the most common communication tool, and there are more and more peripherals for mobile phone [10]. Among its various interfaces, headset port is the only one open, standardized, backward and forward compatible. In this paper, we propose MIHBS (Mobile Interface of High Bandwidth for Sensor network) a universal interface that can transfer data to and from sensors using standard headset port of high bandwidth. We have implement MIHBS in iOS, Android and Windows Phone, respectively, which are three main operating systems of smart phone in the market.

Researches of modern information theory have provided various ways of data compression. Since Shannon and Fano introduced the fundamental coding scheme (not published), Shannon coding has been proved to be optimal and is applied to Huffman coding [7], [19]. LZW [21] is a lossless information compression scheme which introduces variable length codes, and it deals with bit strings directly. In LZW, one code increases 1 bit at a time to generate a new code, and then the index of the original code are used to form the new codeword. Previous works have been trying to make LZW more efficient. Samanta et. al. [23] presented an efficient hardware architecture for LZW to perform both encoding and decoding procedure simultaneously with a CAM array. Amit Setia and Priyanka Ahlawat [25] improved the performance of LZW by eliminating spaces from data file. Saravanan et. al. [24] proposed an enhanced Huffman coding scheme by conducting LZW procedure after Huffman code. In this paper we present LZW-Huffman, which is an enhanced LZW algorithm that changes the coding part of LZW and then combines it with Huffman coding. The LZW-Huffman algorithm is tested and proved to be efficient. Further, we apply it in a self-designed interface, MIHBS (Mobile Interface of High Bandwidth for Sensor network), which is a universal inter-

face that can transfer data to and from sensors using standard headset port of high bandwidth. A preliminary version of this paper appeared in IEEE INFOCOM 2013 [18].

The rest of the paper is organized as followings. Section 2 is the related work. Section 3 introduces our combined LZW-Huffman algorithm in detail. Section 4 presents MIHBS architecture, introducing data communication and data compression. Section 5 discusses the implementation of hardware of ours system and describes the sensor node GF-100 and Gen-OS. In Section 6, we present a fire monitoring application to validate MIHBS's feasibility. Section 7 gives the conclusion.

II. RELATED WORK

A. NON-HEADSET PERIPHERALS

Numbers of designs have been created to allow external sensor peripherals to communicate with mobile phones. In place of using a headset port, they use Bluetooth, USB, or other interface selections. These selections are standardized, but not always available, open or able to support from the device.

Researchers from University of Washington and Microsoft Research Institute developed a sensor platform named FoneAstra [6]. This hardware plug-in enables sensing applications on low-tier mobile phone featuring in low-costing and programmable. The system is made up of ARM7 microprocessor, which is connected by serial I/O and communicates with mobile phone. It can be powered by mobile phone batteries or using separate external batteries.

Little Rock [22] has proposed a new mobile phone architecture, where the sampling and processing of sensor data is unloaded to the secondary low-power microprocessor. This method allows the main processor to enter sleep mode to achieve long-term perception. If integrated into the new mobile phone successfully, Little Rock can achieve longer sampling interval and improve efficiency. However, HiJack has the advantage of being compatible with today's mobile phones without modifying the hardware.

Low power Bluetooth (BT) [8], [30] may become a popular standard for connecting wireless sensors to mobile phones in the future. However, low power BT may be limited to smart phones without supporting functional phones. More importantly, low power BT still needs external energy sources to supply power to sensor peripherals. AudioDAQ [9] is different from these systems because it gets energy from the phone's microphone line and supplies power to the sensor peripherals, thus avoiding the use of external power.

B. AUDIO HEADSET PERIPHERALS

HiJack [16] uses the audio interface to provide power and setup a bi-directional data communication link with external sensing attachment. However, it is designed specifically based on the iPhone's audio interface characteristics and it is not clear if it can be easily extended to be generally applicable on other operating systems such as Android and Windows Phone. Besides, their energy harvester can only delivers 7.4mW to a load which cannot even satisfy the consumption of ZigBee controller (CC2420) of sensors. Further, HiJack only supports sporadic data transmission and offers 8820 baud bandwidth which is the bottleneck of Phone-WSN system.

Like HiJack, RedEye Mini [1] also utilizes the audio output driver to drive LED as the TV remote control. It integrates scalable microcontrollers and similar power acquisition circuits. It supplies greatly higher power for the load by using the left and right audio output channel, but the result is that the battery of the mobile phone runs faster. Although HiJack and RedEye Mini can provide more output power, its cost is the power consumption of the whole system. For large class sensing applications, this high output power is not necessary and makes them unrealistic for long-term data collection on mobile phones.

Square [2] is a credit card reader, which can be connected to Android and iOS devices through 3.5 mm audio jack to support mobile payment. Based on our own disassembly, we found that it could detect the voltage peak (30 mV peak-to-peak) produced by the oscillating magnetic field generated by the credit card being pushed into the magnetic head. It has powerful algorithms to extract data and use error correction solutions to improve detection rate. It is a completely passive device without the power supply of earphone port. We have learned from a simple hardware design that can be enhanced by powerful algorithms to achieve complex functions.

S. Verma et al proposed AudioDAQ [29], a new platform for continuous data acquisition using the headset ports of mobile phones. Unlike the existing mobile phone peripherals, AudioDAQ extracts all the necessary power from the microphone bias voltage, encodes all the data into analog audio, and uses a built-in voice memo application (or custom application) for continuous data acquisition.

C. INTERFACE BETWEEN WSN NODE AND MOBILE PHONE

Abhishek Bhardwaj and others proposed that MELOS [5] (low energy sensing mobile extension) is a low-cost mobile phone extension that can work with almost all mobile phones, including low-cost phones, because it is connected to a cell phone through a standard audio port and Bluetooth. This paper discusses the architecture and various operation modes, and shows that the energy consumption of MELOS nodes is reduced. The author also introduced a case study of energy monitoring and equipment control system using MELOS nodes. Chunxiao Jiang [14] proposed an SD-based universal sensor data entry card, named uSD which is plug-and-play. Although uSD can help consumers to access WSN conveniently with portable devices, the card slot of SD is disappeared in most smart phone such as iPhone 4S and Lumia. As a result, there will be less and less market for uSD in the future. Since MIHBS interfaces with a smart phone over standard headset port, it is generic enough to work with a wide variety of smart phones.

III. SYSTEM DESIGN

The architecture of MIHBS is shown as Figure 1. The smart phone can communicate with the base station via the four-core headset port which consists of the left channel, the right channel, the ground ring and the microphone. The left channel provides encoded data from the phone to the micro-controller which embedded in the base station. The microphone provides encoded data from micro-controller to the smart phone. The right channel and the ground ring provide AC power to the energy harvester.

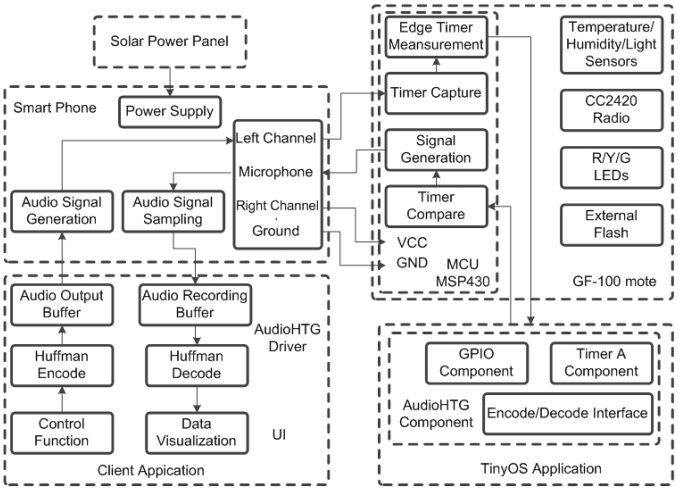


FIGURE 1: Architecture of MIHBS. Hardware device contains smart phone and GF-100 mote. Software contains client application and TinyOS application.

A. DATA COMMUNICATIONS

The main goal of our design is to enable bi-directional interfacing between the smart phone and sensor with high bandwidth. Because it must operate in the audio frequency range, the phone audio sampling rate is an important factor which constraints the bandwidth. The method we choose is to transmit UART [26] data over the headset port. However since the audio channel is AC signal, a long sequence of zeros or ones saturates the channel and is difficult to decode. To solve this problem, we use Manchester encode the UART data stream. Manchester encoding is a balanced 1:2 code that reduces the overall data rate by a factor of two but transmits an equal number of zeros and ones. The encoder works by replacing every 1 in the input stream with 01, and every 0 with 10.

Figure 2 shows the UART data stream and its corresponding Manchester encoding as seen on the output of the headset port of a phone.

We noticed two things: First, the waveform is not square, so it is band limited. Second, the UART bit is indicated by every other bitwise direction in the Manchester encoded data. The latter observation is the key to our decoder synchronization. If the UART line is high when there is no communication, the Manchester code of the idle UART is a

FIGURE 2: UART data and Manchester encoding

sequence of binary 01. However, the receiver cannot run for a long time from 01s for 10 seconds, resulting in potential loss of synchronization.

Our design solves the resynchronization problem in idle, as shown below. Once the transmitter sends the UART start bit, the receiver will detect a double baud rate positive sign, as shown by the first high-to-low transition of the UART data. This symbol synchronizes the receiver with the bitstream because it shows that the current UART bit is a zero crossing of zero, so the previous bit sequence is a sequence of 1s.

We use microcontroller peripherals again to achieve efficient, low-power implementations. Using a comparator at Vcc/2, we measured the zero-crossing time of the Manchester coded signal, which drives the input line of the capture unit at the rising edge and the falling edge sensitive time. The edge is captured to edge time and compared to the baud rate, then the state machine decodes and outputs the received UART bit.

Similar methods can be applied to the decoder running on mobile phones [4]. However, rather than using hardware peripherals, we implement comparators and time capture in software. The comparator simply implements a less than test for slicing the input audio samples to 1 and 0. In order to simulate a timer unit, we take advantage of the fact that the phone acquires audio signals at a frequency of 44.1 kHz. Therefore, the time between consecutive samples is 22.68 us. By calculating the number of samples between a pair of zero crossings, we estimate the symbol width so that the Mancheter encoded UART signal can be decoded in the same way as the microcontroller.

We test several baud rate combinations to determine the limits of our current combination of smart phones and the MSP430F1611. The result is shown in Table 1.

We use serial communication protocol to analyze valid data. Figure 3(a) shows traditional frame format which contains 1 start bit, 8 valid data bits, 1 parity bit, 1 stop bit and 1 idle bit. It is obvious that the valid data ratio is about 66.7%

TABLE 1: Baud Rate Test

Smart Phone	Sample	Baud rate	Error ratio
iPhone4S	4	11025	12%
	6	7350	0.003%
	8	5513	0%
Xperia lt18i	6	7350	1%
	8	5513	0.04%
	10	4410	0%
LG E900	4	4000	8%
	8	2000	1%
	16	1000	0%

regardless of transmission error. In order to improve valid data ratio, we change the 8 bits valid data to a dynamic buffer as shown in Figure 3(b). Considering the synchronization and real-time of the communication, we find 128 bits buffer can achieve the balance, of which the valid data ratio is about 96.96%.

Although 8 bits is more stable than a specific long data format, it is not flexible enough in many application scenarios, and the bandwidth is very limited. Compared with 8 bits specific long data formats, dynamic buffer encoding can transmit valid data of any length in one data transmission, which greatly improves communication efficiency and flexibility. The advantages of dynamic buffer communication coding are more reflected in the improvement of bandwidth and communication capabilities. Under the condition of the same bit rate, the proportion of valid data in the dynamic buffer coding mode is greater than 8 bits specific long data. Moreover, in a special scenario, sometimes a large amount of data is required for one transmission, and the researcher can customize the buffer length according to specific requirements, thereby greatly improving the practicability and flexibility of the system. The dynamic buffer is encoded in the same way as 8-bit valid data. The data format is: start bit, data bit, parity bit, stop bit and idle bit. Different from the 8-bit effective data encoding method, the valid data is converted from 8 bits to n bits of arbitrary length, as shown in Figure 3.
(1) When n=8, the dynamic buffer encoding method is equivalent to the 8-bit fixed length encoding method.
(2) When n>8, the effective data rate of the dynamic buffer coding mode is reasonable under the same baud rate condition. On the upper than 8 fixed length coding method. The theoretical effective data rate of 8-bit fixed-length coding mode is: valid data 8 bits / (start bit 1 bit + valid data 8 bits + check bit 1 bit + stop bit 1 bit + idle bit 1 bit) = 75%; The effective buffer rate of the dynamic buffer encoding method is: valid data n bits / (start bit 1 bit + valid data n bit + check bit 1 bit + stop bit 1 bit + idle bit 1 bit) = n/(n + 4). In the experiment, we usually use 256 bits of valid data bits, and the effective data rate is about 98%.

As Figure 4 shown, it demonstrates the performance of HiJack with 128 bits buffer.

B. DATA COMPRESSION

Two factors limit the achievable baud rate using this technique. The first is that the phone's audio sampling rate at 44.1kHz, 44.1kHz and 16kHz of iOS, Android and Windows

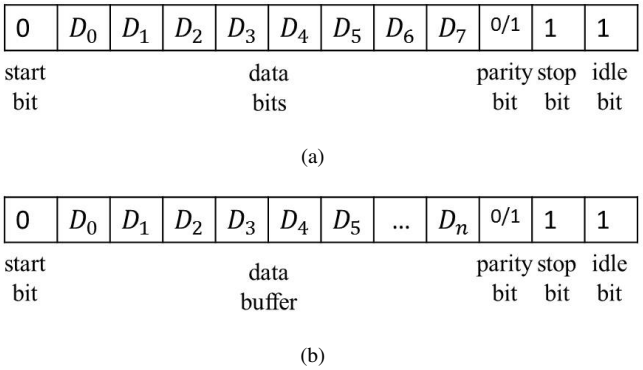


FIGURE 3: Serial and Dynamic buffer format

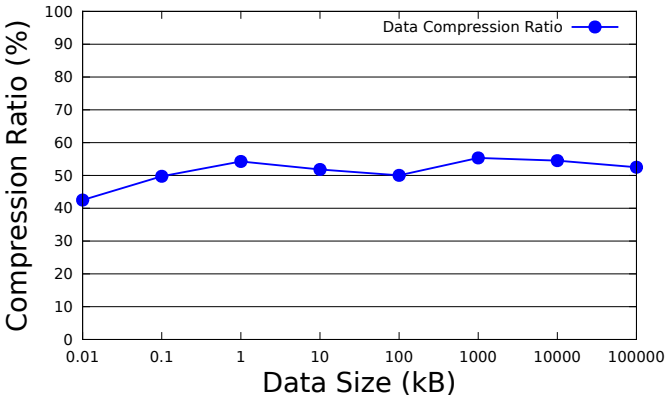


FIGURE 4: The data compression ratio. The range of the samples is from 0.01 kB to 100000 kB. The compression ratio is stable about 52%.

Phone respectively, resulting in a maximum UART baud rate of 22.05 kbaud, 22.05 kaud and 8 kbaud. The second is the clock source used on the microcontroller (MSP430 F1611). Commonly used crystals is 32768Hz. It is important to balance clock speed and data rate. We test several baud rate combinations to determine the limits of our current combination of smart phones and Texas Instruments MSP430F1611. The result is shown as Table I.

We adopt a data compression algorithm which combine Huffman and Lemple-Ziv-Welch, to increase the data transformation speed.

Huffman coding [13] is a way to assign binary codes to symbols which reduces the overall number of bits used to encode a typical string of those symbols. It is a greedy way of minimizing codeword length expectation which changes the codeword into variable length format after building a Huffman coding tree with the weight of each codeword (frequency of appearance).

Letters from the alphabet do not appear at the same frequency. And the number of letters in a specific text file may not be very large. Huffman coding deal with the redundancy in this type of situation. It minimizes the expectation of code word length by making those more frequently appeared

codeword shorter.

Lempel-Ziv-Welch coding (LZW) [28] [21] is another type of lossless data compression scheme. Not like Huffman coding, LZW compresses data directly with bit strings. LZW introduces the idea of variable width code. Traditional LZW algorithm takes the index of codes followed by one 1 or 0 bit to represent a new code. Codeword set of LZW is much larger than that of Huffman scheme, and it gets larger when data string gets longer.

In our enhanced LZW coding we made a little change on the coding part of the algorithm and there is actually a slight increase in the compression efficiency.

The detail of our compression algorithm is introduced in the later section of this paper.

C. ENERGY HARVESTING

As Figure 5 shown, the input of AC step up transformer are right channel and ground ring. The transformer amplifies the audio signal of right channel, then the signal goes through a full bridge rectifier and be filtered by C1, C2, C3 electric capacity. Finally, the output of voltage stabilization chip is DC power for peripherals. The Schottky diodes D1 can prevent from owing backward. The lithium battery and electric capacity enable the circuit to work in high load.

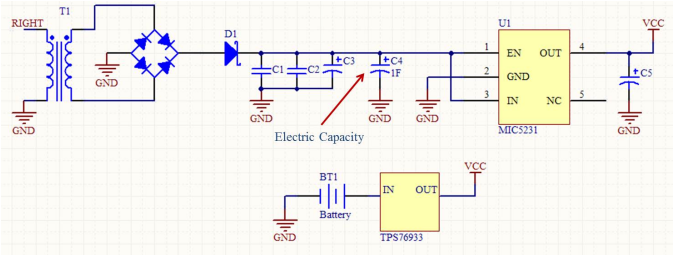


FIGURE 5: The energy harvesting circuit.

Although original HiJack energy harvester can supply 7.4 mW from a single audio channel on the headset port, we have to face the problem that the HiJack energy harvester cannot satisfy the radio frequency of ZigBee which should be about 10 mW [12]. Besides, some other devices on sensor board, like MSP430, also cost energy. Our first solution is to provide a lithium battery for wireless sensor. It's really an easy and convenient method. Another solution is to embed an electric capacity which can harvest energy when the load is lower than 7.4 mW and provide higher power when the load is high [15].

IV. THE ALGORITHM

We propose a compression algorithm called LZW-Huffman, which takes LZW as the basic algorithm for our system data compression, and we enhance it with Huffman coding. Saravanan et al. [24] enhanced the efficiency of Huffman coding by adding an LZW procedure after Huffman coding, which is in reverse order of our proposed scheme. As Figure 6 shown, LZW-Huffman can be divided into four steps, which will be introduced in the following parts.

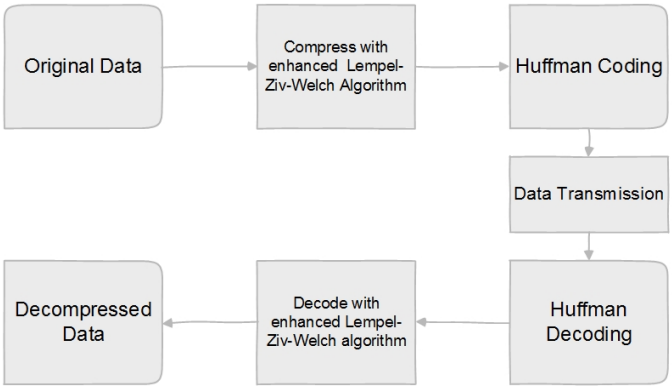


FIGURE 6: Data Compression and Decompression Procedure

A. LZW ENCODING

LZW encoding deals with bit strings directly. Take the following bit string as an example: 0100101010100111. The LZW encoding algorithm will split this string into several smaller variable length substrings. The above 16 bits string will be split as:

$$\underbrace{0}_{1}\ \underbrace{1}_{2}\ \underbrace{00}_{3}\ \underbrace{10}_{4}\ \underbrace{101}_{5}\ \underbrace{01}_{6}\ \underbrace{001}_{7}\ \underbrace{11}_{8}$$

Each substring has an index number which indicates its location in the substring list. It is hard to figure out what is going on with the original string, here the algorithm is introduced:

Let S denote the list of substrings, initialized as an empty set, which is $S = \phi$. For any string s, len(s) denotes the length of string s, s_n^m denotes s's substring which starts at nth element of s and ends with the mth element. Take 'string' as an example, let a, then $len(a) = 6$ and $a_1^3 =' str'$. Let I be the bit string that the algorithm is to split.

The substrings are added to the end of the list sequentially. To add a bit string to S we have to find a string s where $s = I_1^m$ and $\forall n \in N^*$ and $I_1^n \in S, m - 1 \geq n$, then add it to S as its last element. There are three exceptions: the first single '1' bit, the first '0' bit and the last possible possible substring, these bits or strings does not have to subject to the conditions described above.

The LZW encoding generates tuples to represent encoded words. Each tuple consists of a number which indicates the index of the prefix of the encoded word and a single bit which is the last bit of the encoded word. For the first '1' and '0' bit the index is dented as 0.

With S the coding dictionary can be represented as the Table 2 below:

B. ENHANCING THE LZW

Our proposed scheme enhances the LZW algorithm by improving the coding procedure.

LZW uses constant length codeword to represent bit strings with variable length. The index of a specific codeword is the key to find the original information.

TABLE 2: LZW Coding Dictionary

Index	Substring	Encoded words	Index	Substring	Encoded words
0	ϕ	None	1	0	(0,0)
2	1	(0,1)	3	00	(1,0)
4	10	(2,0)	5	101	(4,1)
6	01	(1,1)	7	001	(3,1)
8	11	(2,1)			

However, a specific index can only be used at most twice to represent bit strings. So in the best case if a dictionary has 1020 codeword, then there are at least about 800 different index numbers (about two third of 1020, theoretical analysis will be given later). It takes 10 bits to represent numbers ranging from 0 to 800.

Since the coding is conducted sequentially, it is better if we use the difference of two indexes to represent the previous part of the string that is being encoded. For any string a that is to be added to S, and the index of a and $a_1^{len(a)-1}$ is n and m ($n > m$), then the code word tuple of a can be represented by $(n-m, a[len(a)])$, where $a[k]$ represents the kth bit of the string a. This way, index numbers that take 10 bits to represent can be represented with 9 or even fewer bits.

For the previous example, the new coding dictionary is shown in the following Table 3:

TABLE 3: New LZW Coding Dictionary

Index	Substring	Encoded words	Index	Substring	Encoded words
0	ϕ	None	1	0	(0,0)
2	1	(0,1)	3	00	(2,0)
4	10	(2,0)	5	101	(1,1)
6	01	(5,1)	7	001	(4,1)
8	11	(6,1)			

C. HUFFMAN CODING

Huffman encoding comes after the procedure of the enhanced LZW. As described in the previous part, the enhanced LZW uses the index difference to represent a specific code word. The same index number may appear in the text again and again. Huffman coding may help to lower the expectation of the length of total information string by building a Huffman coding tree.

The first step of building a Huffman coding tree is to find appearance frequency of each codeword. For the codeword in Table 3 the appearance frequency table is shown in Table 4:

TABLE 4: Frequency of Index Numbers

Index	0	1	2	4	5	6
Frequency	0.25	0.125	0.25	0.125	0.125	0.125

A binary coding tree is then built with this table. Let C be the set of nodes of index difference numbers. Each node has its appearance frequency as its weight. Here is the algorithm:

E1. Remove two nodes with lowest frequency from C, and denote the removed nodes as a and b.

E2. Insert a new node (denoted as r) into C, the weight of r is the summation of the frequency of a and b. Construct the tree by making a and b the children of r.

E3. If $|c| = 1$ terminate the process, else go back to E1.

For the previous example, the Huffman tree is shown in Figure 7:

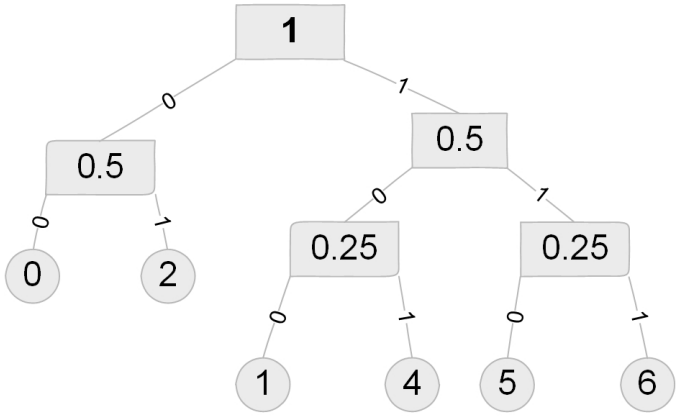


FIGURE 7: Huffman Coding Tree

After Huffman encoding procedure the coding dictionary is shown in Table 5.:

TABLE 5: New LZW Coding Dictionary

Index	Substring	Encoded words	Index	Substring	Encoded words
0	ϕ	None	1	0	(00,0)
2	1	(00,1)	3	00	(01,0)
4	10	(01,0)	5	101	(100,1)
6	01	(110,1)	7	001	(101,1)
8	11	(111,1)			

For traditional LZW encoding, the coding dictionary is shown in Table slowromancapii@. The index numbers range from 0 to 4, which takes at least 3 bits to represent each of these numbers. The total length of the encoded information of traditional LZW is at least 32 bits($(3+1) \times 8 = 32$). The enhanced LZW however, takes 28 bits in total to carry these information. The encoded bit string is shown as follows: 00000101001010011101101111111.

It may be confusing, for this so called data compression algorithm takes 28 bits to carry a 16 bits string. It is necessary to mention here that LZW performs much better when it deals with larger amount of data.

D. DECODING

The decoding part is much easier to describe since it simply reverses the encoding procedure.

After the encoded data is transmitted, the receiver decodes received bit string R with the algorithm shown below:

E1. Pick up a code word from R with Huffman tree, decode it. Let a be the decoded information. Then remove the code word from R.

E2. Pick up a single bit from R and denote it as b. Remove that bit.

E3. Append the tuple (a,b) to the end of C_0. If R is empty then terminate the program, else go back to E1.

C_0 is a list of decoded tuple. For any $k = (a, b) \in C_0$, let n be the index of k in C_0, then $(a-n, b)$ is the actual LZW code word for k. This way the original LZW coding list L can be retrieved. L is then decoded with LZW decoding algorithm and finally the original data can be generated by the receiver.

E. PERFORMANCE ANALYSIS

In this section, we analyze the performance of our LZW-Huffman scheme. Modern information theory has given us rigorous analysis of both methods, here we present both theoretical and experimental performance analysis of the combined algorithm.

Data compression ratio [3] shows how well a specific data compression algorithm performs. Let r be the data compression ratio, u the uncompressed data size and c the compressed data size, then $r = \frac{u}{c}$. Noticing that the definition of data compression ratio here is different from the ratio used in section 2, the new definition is defined mainly for the use of theoretical and experimental analysis in this section.

Suppose we have a string with n bits, and it is broken up into $b(n)$ pieces. Then the precious phrase can be represented by at most $\log_2 c(n)$ bits. Thus the length of the encoded data after traditional LZW encoding is at most $c(n)(E[l]+1)$ bits, where $E[l]$ is the expectation of the length of each code word. For traditional LZW algorithm, $E[l] = \log_2 c(n)$. Since there are two main steps in our proposed scheme, we let $E_1[l]$ be the be the expectation of codeword length after the first step. It can be easily proved that for any $c(n)$, $\exists k > 1$, $E_1[l] \leq \log_2 c(n)$, which is $E_1[l] = O(\log_2 c(n))$. The real code word length after the first step depends on the real situation.

For variable length codewords the expectation is defined as follows:

$$E[l] = \Sigma_i P[i] l_i$$

where $P[i]$ is the appearance frequency of a specific codeword i, and l_i is the length of i.

Huffman coding generates variable length codewords. In the second step Huffman encoding procedure is conducted, the codeword length expectation depends on the appearance frequency distribution of index difference number. For any specific bit string, after we conduct the first step, we denote the index difference number set as W. The entropy of the whole codeword set is defined as:

$$H[X] = -\Sigma_{i \in W} P[i] \log_2 P[i]$$

Huffman encoding can lower the codeword length expectation, previous research in information theory has proved that Huffman coding is optimal [20]. Optimal code is a widely used definition of information theory.

Optimal code has minimized codeword length which subject to:

$$H[x] \leq E^*[l] \leq H[X] + 1$$

where $E^*[l]$ is the codeword length expectation of optimal code.

Thus the length of the encoded bit string is $L = c(n)(E^*[l] + 1)$, and

$$c(n)(H[X] + 1) \leq L \leq c(n)(H[X] + 2)$$

Thus the data compression ratio can be calculated this way:

$$r = \frac{n}{c(n)(E^*[l] + 1)}$$

As the data amount grows larger the ratio r always depends on the appearance frequency distribution, as we have:

$$\frac{n}{c(n)(H[X] + 2)} \leq r \leq \frac{n}{c(n)(H[X] + 1)}$$

According to the formula listed above we can see that the proposed LZW-Huffman combination performs better with uneven distribution of index difference number set. To understand how the combined scheme works in real time situation we tested a 116 kB .tex text file. We first read every character byte and turn it into 0 and 1 bit string, then we conduct the combined algorithm. We tested the data string several times, with data string of different length ranging from 1000 bits to 100 kB.

After all this average codeword length (codeword length expectation) and data compression ratio are then calculated to show how the algorithm works.

Figure 8 shows how average codeword length changes when the data string gets larger.

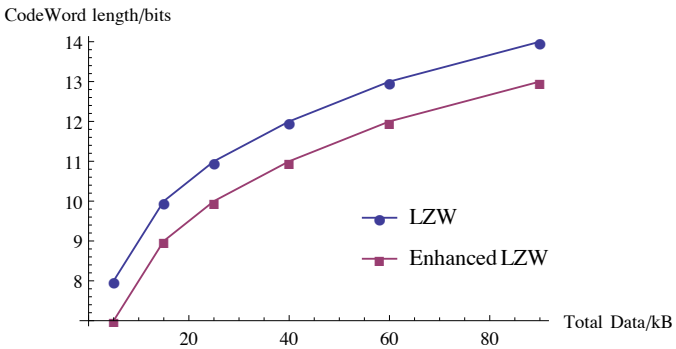


FIGURE 8: Codeword Length Expectation

We can see that as the data string increases the codeword length expectation of combined LZW-Huffman is always larger than that of traditional LZW, and with more data tested it is true that the combined algorithm increases with a lower speed.

With the new definition of data compression ratio, total performance of both LZW and combined LZW-Huffman is shown in Figure 9:

Noticing that there are two peaks in Figure 9 on both performance curves. The performance of LZW gets better as the data string gets longer. However, every time the codeword length of the encoded data gets one bit longer, the performance will be slightly disturbed. LZW-Huffman always performs better than single LZW algorithm.

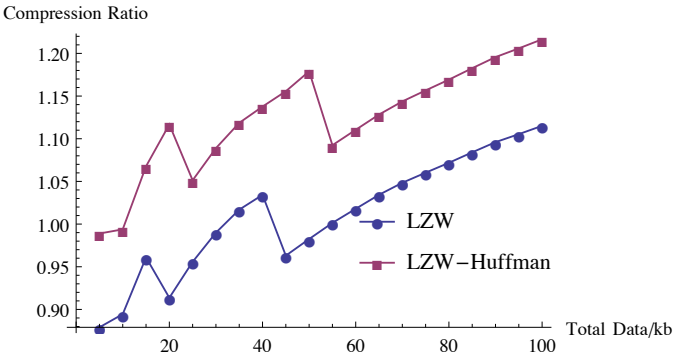


FIGURE 9: Data Compression Ratio of LZW and LZW-Huffman

V. IMPLEMENTATION

We separately develop experiments on different operating system of smart phones. According to the system design discussed above, the MIHBS system is implemented as Figure 10. The MIHBS system consists of four subsystems:

- Smart Phone: iPhone 4S, Sony Ericsson lt18i and LG E900.
- Solar power panel: The energy harvester enables smart phone to work indefinitely.
- Headset Audio-Jack: We use a 3.5 mm phono jack/plug to output audio to headphones and receive input from a microphone.
- Sensors: GenOS-301 and GF-100 motes is our hardware platform, SHT11 module senses temperature, ELT-S-100 module senses CO_2.

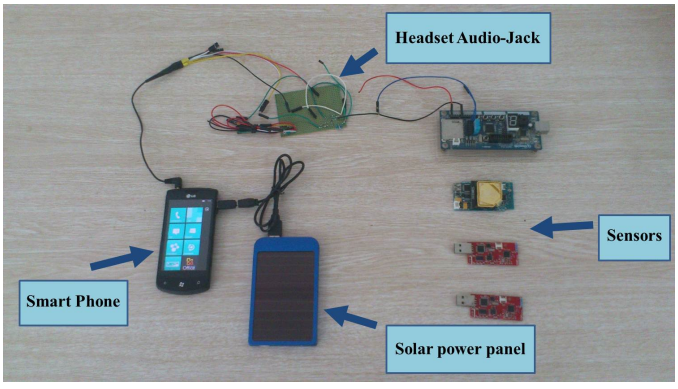


FIGURE 10: The hardware architecture of MIHBS system, in which a solar power panel is associated to provide continuous energy support.

In order to make TelosB Motes more tiny and compatible, we design our own motes, GF-100 and GenOS-301, on which temperature, humidity, CO_2 and other sensors are integrated.

GF-100 is a full TelosB compatible WSN mote with lithium cell and power supply function. A standard mote includes a mini-USB port for programming and data transfer, a IEEE802.15.4 radio TI CC2420, a low power MCU MSP430 F1611 with 10k RAM, an external flash chip up to 1MB. The characteristic of GF-100 is following:

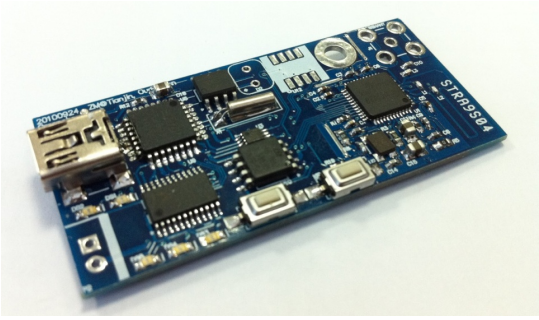


FIGURE 11: Picture of GF-100 mote

- GF-100 enables the experimentation for communications and is a low power mote with one lithium cell, which allowing for longer battery life.
- GF-100 mote works with the open-source operating system. TinyOS is a small, open source, energy-efficient OS developed by UC Berkeley. By using TinyOS, GF-100 supports large scale, self-configuring sensor networks.
- GF-100 is powered by no more than 2 lithium cells, or other type of battery, such as AA/AAA batteries, or any DC power-suppliers less than 10V. When the mote is plugged into the USB port, the computer will provide power and communicates though the Mini-USB port.
- Optional temperature/humidity sensor and light sensor integrated, which decreases the size and cost of GF-100.

GenOS-301 is has GPRS/GPS modules and is more complicated than GF-100. As GenOS-301 is designed in split pattern, it is easier to integrate with MHIBS interface for experiment. The architecture and specification of GF-100 is shown in Figure 12 and Table 6 respectively.

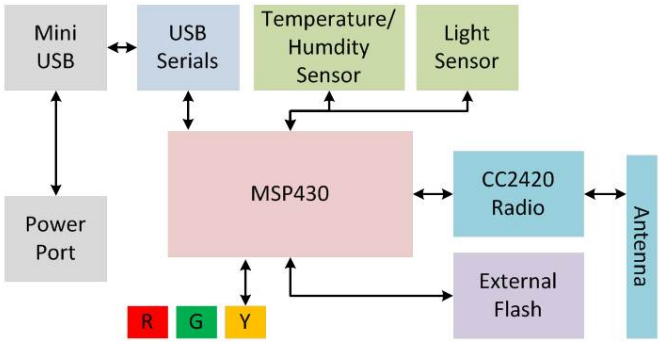


FIGURE 12: Architecture of GF-100 which contains temperature sensor, CC2420 radio, MSP430 microcontroller and other modules.

VI. CASE STUDY

Nowadays, wireless sensor networks can be applied to smart building monitoring, especially fire monitoring. Due to the

TABLE 6: Specification of GF-100

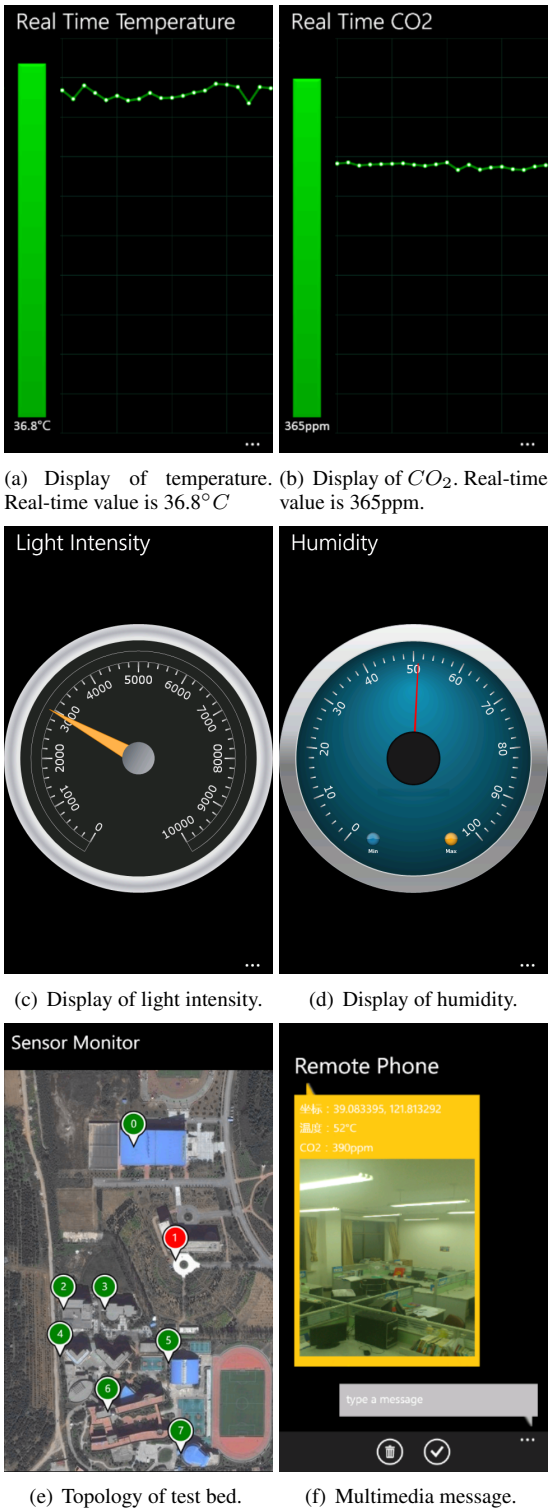
Specifications	Parameters	Remarks
Module	**Microcontroller Storage**	
Processor	MSP430F1611	16bit-RISC
Program Flash	48k Bytes	
Running RAM	10k Bytes	
Information Flash	256 Bytes	
External Flash	1024k Bytes	M25P80
Communications	UART-USB	FT232
ADC	12 bit ADC	8 Channels
DAC	12 bit DAC	2 Ports
Current Draws	2mA	Active Mode
	10uA	Sleep Mode
Module	**Radio Transceivers**	
Radio	CC2420	TI
Frequency	2.4000GHz-2.4835GHz	ISM Band
TX Data Rates	250kbps	
TX Power	0dBm	
RX Sensitivity	-90dBm	
Current Draw	23mA	RX Mode
	22mA	TX Mode
	1uA	Sleep Mode
Module	**Power Supply Interface**	
USB Power	Mini-USB Port	5V DC
Power Connectors	2PIN DC	3V - 10V DC
		1 Lithium Cell
Communications	USB	USB2.0, USB1.1
Module	**Sensors(Optional)**	
Light Sensor	320nm - 730nm	S1087
Humidity Sensor	0 - 100% RH	SHT11
	0.03% RH	Resolution
	3.5%RH	Accuracy
Temperature Sensor	-40°C - 120°C	SHT11
	0.01°C	Resolution
	0.5°C	Accuracy
Module	**Physical Size**	
Size	55 * 26 * 6	L/W/H mm

limited functions of scalar sensor nodes, in some hot weather areas, e.g., Saudi Arabia, using only temperature and CO_2 sensor nodes cannot correctly detect and report the fire event. More often than not we need real-time alarm, analyzed data and photos of scene [17]. The smart phone is an excellent choice for gathering multimedia data to describe some complicated event, e.g., taking an image of the fire or smoke.

In this section, we design a demo to validate MIHBS's feasibility and platform independence, called "Fire Monitoring System", which is specifically used to monitor buildings and prevent from fire. In the system, the sensor nodes (GF-100) broadcast temperature and CO_2 information via ZigBee chip in 802.15.4 protocols. While base station (GenOS-301) receives such information and send it to smart phones via MIHBS. When both the sensory reading of temperature and CO_2 is higher than a threshold, the smart-phone will take photos of scene, and in case of seeing fire or smoke send alarm message to a remote phone.

The demo contains seven MIHBS's systems and one remote phone in our campus. And we successfully executed the prototype system. We will demonstrate the following:

- Real-time display of temperature and CO_2. As the system starts, SHT11 senses temperature and ELT-S-100 senses CO_2 of the air, then they transmit data to the sink node which is connected to a smart phone by audio



(a) Display of temperature. Real-time value is 36.8°C

(b) Display of CO_2. Real-time value is 365ppm.

(c) Display of light intensity.

(d) Display of humidity.

(e) Topology of test bed.

(f) Multimedia message.

FIGURE 13: Windows Phone application. The alarm contains coordinate, temperature and CO2.The part whose temperature or CO2 exceeds the threshold turns red and normal parts stay green.

jack. The smart phone uses Windows Azure for storage. As shown in Figure 13(a), 13(b), user's remote phone has access to the cloud and monitors the real-time data of buildings by GPRS.

- Visualization of test bed topology. The application provides a map of building surrounding and a topology of sensors-and-phone which contains GPS device, shown in Figure 13(c).
- Multimedia message alarm. When the value of temperature and CO_2 exceeds the threshold, the connected smart-phone will take photos of the scene, and send multimedia message alarm to user's remote phone, shown in Figure 13(d). After receiving alarm, the dangerous part of the topology will become red. It is clear and convenient for us to determine the position.

VII. CONCLUSIONS

We show that it is possible to adapt the ubiquitous smart phone headset port to provide power and data, allowing the phone to interface with sensors easily and economically. In this paper, we propose an enhanced Lempel-Ziv-Welch coding scheme to help transmitting data faster, called LZW-Huffmam. Further, we applies it in a self-designed mobile extension called MIHBS, which interfaces with a mobile phone over generic audio. By using LZW-Huffman, the bandwidth of MIHBS increased by 52%. It not only can help consumer with smart phones to access WSN conveniently and efficiently, but also are appropriate for researcher constructing test bed. Data compression is very important for expanding bandwidth of our system design. With the LZW-Huffman coding, the data compression ratio is increased efficiently.

REFERENCES

[1] Redeye mini universal remote for iphone, ipod touch & ipad.
[2] Square. square card reader.
[3] Mohammed Ashraf, Hassan Mostafa, and Ahmed A El-Adawy. A low-power area-efficient design and comparative analysis for high-resolution neural data compression. In Microelectronics (ICM), 2016 28th International Conference on, pages 217–220. IEEE, 2016.
[4] Priyanka Bagade, Ayan Banerjee, and Sandeep KS Gupta. Rapid evidence-based development of mobile medical iot apps. In Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on, pages 1–6. IEEE, 2016.
[5] Abhishek Bhardwaj, Pandarasamy Arjunan, Amarjeet Singh, Vinayak Naik, and Pushpendra Singh. Melos: a low-cost and low-energy generic sensing attachment for mobile phones. In Proceedings of the 5th ACM workshop on Networked systems for developing regions, pages 27–32. ACM, 2011.
[6] Rohit Chaudhri, Eleanor O'Rourke, Shawn McGuire, Gaetano Borriello, and Richard Anderson. Foneastra: enabling remote monitoring of vaccine cold-chains using commodity mobile phones. In Proceedings of the First ACM Symposium on Computing for Development, page 14. ACM, 2010.
[7] Thomas M. Cover. On the competitive optimality of huffman codes. Information Theory, IEEE Transactions on, 37(1):172–174, 1991.
[8] Joshua F. Ensworth and Matthew S. Reynolds. Ble-backscatter: Ultralow-power iot nodes compatible with bluetooth 4.0 low energy (ble) smartphones and tablets. IEEE Transactions on Microwave Theory and Techniques, 65(9):3360–3368, 2017.
[9] Petko Georgiev, Nicholas D. Lane, Cecilia Mascolo, and David Chu. Accelerating mobile audio sensing algorithms through on-chip gpu offloading. In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, pages 306–318, 2017.
[10] Carol Habib, Abdallah Makhoul, Rony Darazi, and Christian Salim. Self-adaptive data collection and fusion for health monitoring based on body sensor networks. IEEE Transactions on Industrial Informatics, 12(6):2342–2352, 2016.
[11] Guangjie Han, Li Liu, Jinfang Jiang, Lei Shu, and Gerhard P Hancke. Analysis of energy-efficient connected target coverage algorithms for industrial wireless sensor networks. IEEE Transactions on Industrial Informatics, 13(1):135–143, 2017.
[12] Fred Harris, Richard Bell, and Vamsi Krishna Adsumilli. Spectrum sharing between a zigbee frequency hopper and an fsk modem. In Dynamic Spectrum Access Networks (DySPAN), 2015 IEEE International Symposium on, pages 3–4. IEEE, 2015.
[13] D.A. Huffman. A method for the construction of minimum-redundancy codes. Proceedings of the IRE, 40(9):1098–1101, 1952.
[14] C. Jiang, N. He, Y. Ren, C. Chen, and J. Ma. usd: universal sensor data entry card. Consumer Electronics, IEEE Transactions on, 56(3):1450–1456, 2010.
[15] Junaid Ahmed Khan, Hassaan Khaliq Qureshi, and Adnan Iqbal. Energy management in wireless sensor networks: A survey. Computers & Electrical Engineering, 41:159–176, 2015.
[16] Y.S. Kuo, S. Verma, T. Schmid, and P. Dutta. Hijacking power and bandwidth from the mobile phone's audio interface. In Proceedings of the First ACM Symposium on Computing for Development, page 24. ACM, 2010.
[17] Chenyang Lu, Abusayeed Saifullah, Bo Li, Mo Sha, Humberto Gonzalez, Dolvara Gunatilaka, Chengjie Wu, Lanshun Nie, and Yixin Chen. Real-time wireless sensor-actuator networks for industrial cyber-physical systems. Proceedings of the IEEE, 104(5):1013–1024, 2016.
[18] Can Ma, Lei Wang, Zhenquan Qin, Ming Zhu, Weifeng Ying, Jiaqi Xu, Lei Shu, and Canfen Chen. Improving the throughput of smart phone audio-jack based gateway for sensor networks. In Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on, pages 89–90, 2013.
[19] Manuel Martinez, Monica Haurilet, Rainer Stiefelhagen, and Joan Serra-Sagristà. Marlin: A high throughput variable-to-fixed codec using plurally parsable dictionaries. In Data Compression Conference (DCC), 2017, pages 161–170. IEEE, 2017.
[20] Brockway McMillan. Two inequalities implied by unique decipherability. Information Theory, IRE Transactions on, 2(4):115–116, 1956.
[21] Mark R Nelson. Lzw data compression. Dr. Dobb's Journal, 14(10):29–36, 1989.
[22] Bodhi Priyantha, Dimitrios Lymberopoulos, and Jie Liu. Enabling energy efficient continuous sensing on mobile phones with littlerock. In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, pages 420–421. ACM, 2010.
[23] Rupak Samanta and Rabi N Mahapatra. An enhanced cam architecture to accelerate lzw compression algorithm. In VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on, pages 824–829. IEEE, 2007.
[24] C Saravanan and M Surender. Enhancing efficiency of huffman coding using lempel ziv coding for image compression. International Journal of Soft Computing, 2, 2013.
[25] Amit Setia and Priyanka Ahlawat. Enhanced lzw algorithm with less compression ratio. In Proceedings of International Conference on Advances in Computing, pages 347–351. Springer, 2012.
[26] W. Stallings. Data and computer communications. Electronic Mail¡ªSMTP and Mime. MacMillan, pages 701–724, 1997.
[27] Li Sun, Pinyi Ren, Qinghe Du, and Yichen Wang. Fountain-coding aided strategy for secure cooperative transmission in industrial wireless sensor networks. IEEE Transactions on Industrial Informatics, 12(1):291–300, 2016.
[28] P Tischer. A modified Lempel-Ziv-Welch data compression scheme. 1987.
[29] Sonal Verma, Andrew Robinson, and Prabal Dutta. Audiodaq: turning the mobile phone's ubiquitous headset port into a universal data acquisition interface. In Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, pages 197–210. ACM, 2012.
[30] Haidong Yi, Wei-Han Yu, Pui-In Mak, Jun Yin, and Rui P. Martins. A 0.18-v 382- μ w bluetooth low-energy receiver front-end with 1.33-nw sleep power for energy-harvesting applications in 28-nm cmos. IEEE Journal of Solid-state Circuits, 53(6):1618–1627, 2018.