2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation

Development of Huffman Code for Lora Technology

Pramoth Pongpunpurt
Department of Computer and Information Science
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
s5804062857158@email.kmutnb.ac.th

Weerawat Khawsuk
Department of Electrical Engineering Academic Division
Chulachomkloa Royal Military Academy
Nakhon Nayok, Thailand
khawsuk@gmail.com

Nikorn Sutthisangiam
Department of Computer and Information Science
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
nikorn.s@sci.kmutnb.ac.th

***Abstract*—This research aims to implement the LoRa development boards to transmit measured sensor data. Two data transmission schemes are investigated, one using a traditional ASCII format and another exploiting the Huffman codes. Transmitted character sizes between these two coding representations are recorded and compared in terms of compression rate. To evaluate the error performance, various transmission distances up to $1,000$ meters are investigated. At each receiving location, 250 sets of data are collected for both encoding formats. Experimental results show that the Huffman codes can reduce the transmitted character size on average. Furthermore, using compressed data via the Huffman algorithm can decrease error percentages of incorrect transmitted data.**

***Index Terms*—LoRaWAN, data encryption, Huffman coding**

I. Introduction

Wireless communication becomes an irrefutable technology which has influenced on numerous aspects of every daily life. As more population from rural area moving into cities or suburb area, the need to use smart city technology is gaining more attention. Many types of devices and sensors are deployed in various locations such as electricity poles, buildings and offices, and transportation infrastructures in order to provide useful and manageable data. These gathering information are crucial for monitoring system status or making intelligent decisions on city's offered services.

The concept of Long Range Wide-Area Network (LoRaWAN) gains more interests in developing smart city applications among information technology (IT) researchers as the use of small communication devices increases. The LoRaWAN is designed in such a way that low powered devices can be communicated with Internet-connected applications over longer range wireless connection. Several attractive and practical advantages include a longer lifetime of batteries, a lower price of devices, and a wider range of transmission coverage. In particular, the critical issue of this technology relies on reliable and efficient transmission of data over the wider coverage range.

The LoRa devices from Semtech are among many prevailed technologies for building the Internet of Thing (IoT) network. Their wireless products are an ultimate solution because they eliminate the use of repeaters, reduce the product cost, extend the battery life, and improve the network capacity. Moreover, the Semtech's wireless product lines consist of gateways, transceivers, receivers, and transmitters that cover the industrial, scientific, and medical (ISM) band of radio frequency (RF) spectrum ranging from a low kilohertz up to 2.4 GHz [1].

When thousands of connected devices are simultaneously communicating through a bandwidth-limited channel, network becomes a bottle neck and data communication is less effective. Reducing data size from each sensor would lead to a smaller amount of data traffic and therefore causes a more efficient wireless communication [2]. Data compression is a useful technique in reducing resource usage since it can save memory space and uses network with less bandwidth. There are many algorithms that result in a lossless data compression such as Huffman coding, Lempel-Ziv coding, and Shannon-fano coding. However, the Huffman scheme yields a better performance in compressing data than other algorithms [3].

This research primarily utilizes the Lopy and Pysense microcontroller boards as data transmitting and receiving units. They are LoRa certified products that offer a perfect combination of power, friendliness, and flexibility. Furthermore, the Huffman algorithm is implemented onto our development boards to achieve a better data compression rate. In stead of transmitting in a typical 8-bit ASCII code format, we encode each character of collected sensor data with the Huffman codes and send them to the receiver units. Several communication distances are investigated to analyze their compression rate and error performance.

This paper is structured as follows. Section II describes the LoRa architecture, its available frequency bands, and specification of development boards, while Section III summarizes the Huffman compression algorithm. In Section IV, the experimental scenarios, parameter settings, and the Huffman codes generated from our collected data are presented. Section V provides data evaluation and result discussion. Finally, the concluding remarks along with its future research are given in Section VI.

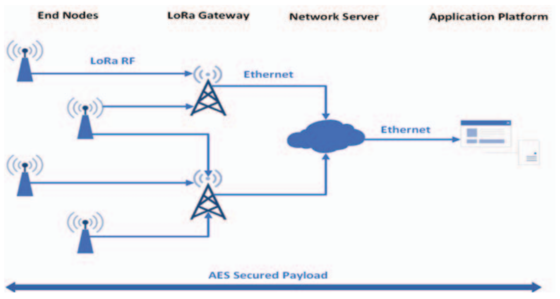

Fig. 1. LoRa Network Architecture

II. LoRa Architecture and Hardware

A. LoRa Architecture

The LoRa network architecture illustrated in Fig. 1 consists of 4 main components as follows.

- End Nodes: They are sensors or application where sensing and control take place. These nodes are often employed remotely. Due to the relatively low frequency and rate of transferred data, it is possible to put LoRa end nodes into hibernation or standby. This leads to the considerably lesser power consumption of the nodes. In turn, the battery lifetime can last longer without recharging to maintain devices online continuously.
- LoRa Gateway: In LoRaWAN, nodes are connected with a specific gateway. Data transmitted by the node are sent to all gateways and each gateway transfers it to a cloud based network server. The gateway is also accountable for putting the transmitted or received data in the right place.
- Network Server: It is responsible for handling of uplink data received by the gateways and scheduling of downlink data transmissions. It duplicates data packets from different gateways, performs securing check, and sends the acknowledge (ACK) signal to the gateways. The network server also sends the packet to the specific application server.
- Application Platform: This is the end user application. The cloud platform for IoT provides support for the end-user processes. Multiple application server can exist with the same LoRaWAN network.

B. LoRa Frequency Bands

The available frequency range and transmitted power of LoRaWAN technology are assigned differently in each region. For example, the United States (US) uses a $902-928$ MHz frequency range with a 30 dBm transmitted power, while the European Union (EU) operates at an $863-870$ MHz frequency range with a 20 dBm transmitted power. The Asia Pacific (APAC) region is however offering for both frequency ranges, but with a 14 dBm transmitted power. Table I summarizes frequency ranges, transmitted power, as well as coverage distances in various areas for the US, EU, and APAC regions. For Thailand, the office of National Broadcasting and Telecommunications Commission (NBTC) designates a $920-925$ MHz frequency band for public use of LoRa application [4].

TABLE I
Frequency Band in Each Region

Region		Frequency	
		$902-928$ MHz	$863-870$ MHz
Regional ISM Band		US, APAC	EU, APAC
Device Output Power (UL Power)		US: 30 dBm APAC: 14 dBm	EU: 20 dBm APAC: 14 dBm
Coverage	Dens-Urban	US: 0.88 km APAC: 0.32 km	EU: 0.48 km APAC: 0.32 km
	Urban	US: 1.84 km APAC: 0.64 km	EU: 1.0 km APAC: 0.68 km
	Sub-Urban	US: 5.2 km APAC: 1.75 km	EU: 2.8 km APAC: 1.9 km
	Rural	US: 15.0 km APAC: 5.0 km	EU: 8.0 km APAC: 6.0 km

C. Development Boards and Devices

The Lopy hardware, as shown in Fig. 2 (left), is a MicroPython enabled WiFi, Bluetooth, and LoRa development board designed for IoT applications [5]. It supports 2 operating frequency bands. The first frequency range is between $863-870$ MHz for the EU and APAC regions. It can transmit a power ranging from $2-14$ dBm. Similarly, the second frequency range is between $902-928$ MHz for the US and APAC regions. It also can transmit a power spanning from $5-20$ dBm, spreading factors between $7-12$, channel bandwidth of $125, 250$, and 500 kHz, and coding rates of $4/5, 4/6, 4/7$ and $4/8$.

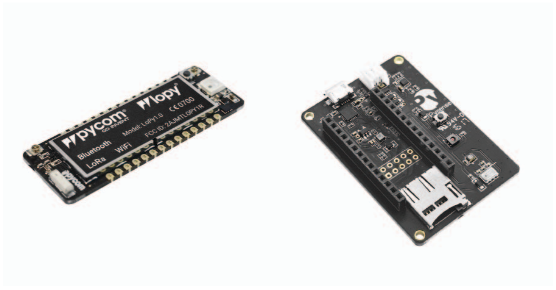

Fig. 2. LoPy hardware (left) and Pysense shield (right)

The Pysense shield, as depicted in Fig. 2 (right), is an add-on daughter-board for environmental sensor projects. It is integrated with a wide range of sensors such as an ambient light sensor, a barometric pressure sensor, a humidity sensor, a 3-axis 12-bit accelerometer, and a temperture sensor. It also has a micro SD card holder for easily keeping a log of measured sensor data and a battery connector with charger for an outdoor operation. According to specification as mentioned in [6], its dimension is small with $55 \times 35 \times 10$ mm^3, and its weight is light at 11 g in a standalone mode. It also operates with an ultra low power at about 1 μA in deep sleep.

III. Huffman Compression Algorithm

The Huffman coding is considered as an optimal compression algorithm in the sense that no other uniquely decodable set codewords has a smaller average codewords length for a given discrete memoryless source [7]. It maps an individual symbol (e.g. alphabets, characters, or numbers) into a variable length codeword (e.g. word or bit sequences) based on prior statistical source knowledge. The length of each Huffman code is directly related to frequency of each character appeared in the text to be compressed. The symbol having a more frequent appearance rate is assigned with a shorter length codeword than those having a lesser appearance rate. The Huffman algorithm is categorized in the same class of the Greedy algorithm since it chooses the best solution (codewords) at any mapping step. Due to the non-ambiguity of generated codewords, it is considered as a lossless algorithm as well [8].

The coding generation begins with constructing a binary tree in a bottom-up manner, rather than a typical top-down style. This tree is then traversed in a pre-order fashion in order to gain codes and mapped them with the original symbols. The Huffman algorithm can be summarized into 4 steps as follows.

1) The alphabet containing unique characters is organized in a minimum heap formed of leaf nodes, where a leaf node corresponds to a character and its frequency. The minimum heap acts as a priority queue in which the frequency field value is used for comparing every two nodes. The least frequent character is a root for the minimum heap.
2) Two nodes are determined and extracted with the minimum frequency in the heap.
3) A new node is formed where the frequency field value is a sum of the two nodes frequencies. The left child is assigned with a 0, while the right child is assigned with a 1. The combined node is then inserted into a new minimum heap.
4) Steps in 2) and 3) are repeated until the heap is formed with a single node. It is the root (100%) node of the complete Huffman tree. Finally, the code for each alphabet is found by working backward and tracing the sequence of 0s and 1s.

IV. Experimental Testing and Results

A. Scenarios and Parameter Settings

The experiment for data transmission takes place in a subub of the Nonthaburi Province (Bangkok metropolitan area) of Thailand. We use the 2nd floor of a residential building as a base for our LoPy transmitter. It is deployed at about 4 meters above the ground level. A surrounding environment consists of a 4-story school building and a tall tree standing in front of the transmitter to obstruct a clear line of sight, as shown in Fig. 3 (right).

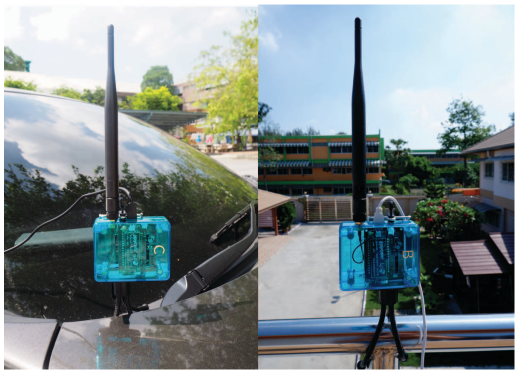

Fig. 3. LoPy receiver (left) and transmitter (right)

The Lopy receiver, as shown in Fig. 3 (left), is placed at about 1.5 meters high from the ground on 4 different locations. The distances are chosen at 315, 500, 750 and 1,000 meters away from the transmitter base. The transmitter site and four receiving locations are displayed in Fig. 4 using Google Map. At each loaction, the compressed data via Huffman algorithm are sent from the transmitter to the receiver for 250 data sets.

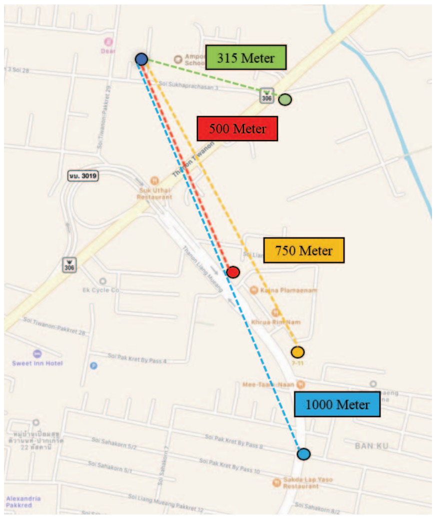

Fig. 4. Experiment locations on Google Map

TABLE II
LoRa Parameter Settings

Frequency	923 MHz
Bandwidth	125 kHz
Spreading Factor	12
Coding Rate	4/8
Transmitted Power	20 dBm

The LoPy uses Semtech's patented LoRa modulation technique SX1272/73 which can achieve a sensitivity of over −137 dBm. The high sensitivity combined with the integrated +20 dBm power amplifier yields industry leading link budget making it optimal. The message in packet format has a maximum size of 255 bytes [9]. In this experiment, the Lopy board is operated at a 923 MHz frequency to transmit data with a channel bandwidth of 125 kHz. To achieve a longer transmission range, we use a 12 spreading factor and a 4/8 coding rate. The transmitted power is set to 20 dBm due to the maximum capability of our Lopy device. The parameter settings for this experiment are summarized in Table II.

B. Data Collection and Huffman Codes

Data collected from our Lopy and Pysense boards are a device identification number, an altitude in meters, a pitch in degrees, 3 axis accelerations for X, Y, and Z directions in relative with the gravitational acceleration (G), a roll angle in degrees, an air pressure in Pascal, a humidity in percent, a temperature in °C, and an ambient light in lux. An example of these measured sensor data sets is illustrated in Table III. For each data set, an individual character, e.g. alphabet, number, and punctuation, is recorded. A total of 100 data sets are gathered and tabulated for further statistical analysis.

TABLE III
Example of Measured Sensor Data Set

Data Type	Value
'id':	'DeviceA'
'altitude':	'25211.23'
'pitch':	'-0.9372684'
'acceleration':	'(-0.08447266, 0.01635742, 0.9957275)'
'roll':	'5.204562'
'pressure':	'100801.2'
'humidity':	'37.23952'
'temperature':	'35.0'
'light':	'(38, 16)'

We use a MicroPython programming language to obtain an average frequency appearance of each character from our 100 recorded data sets. Based on these empirical statistics, the Huffman codes for all recorded characters are generated. Table IV shows an appearance percentage, a Huffman code, and a code length of each character. These Huffman codes are utilized during data transmission of our experiment, instead of a traditional 8-bit ASCII code transmission.

C. Experimental Results

From the experiment, a total of 1,000 data are transmitted for all 4 assigned locations. Fig. 5 shows the frequencies of data size being transmitted without using the Huffman code, e.g. it only transmits with a typical 8-bit ASCII character format. It appears that each transmission sends measured sensor data with the size between 126 – 136 characters (bytes). The maximum frequency of sent data size is 135 characters, while on average the data are transmitted with 133.68 characters.

Similarly, the experiment is repeated with the implementation of Huffman codes onto transmitted data. Fig. 6 illustrates the number of compressed data being send at all 4 locations. There are 1,000 measured sensor data in total. We see that the Huffman compressed data are transmitted with the size between 76 – 84 characters (bytes). The maximum occurrence of send data size is 82 characters, while the average data size is 81.53 characters.

TABLE IV
Character Frequencies and Huffman Codes

Character	Frequency (%)	Huffman Code	Length (bits)
a	1.72	011101	6
A	0.43	10010001	8
c	1.72	101011	6
d	1.29	011100	6
D	0.43	10010000	8
e	4.29	0000	4
g	0.43	111010111	9
i	3.43	10001	5
l	2.15	111011	6
n	0.43	10010111	8
m	0.86	1110100	7
o	0.86	1001111	7
p	1.29	010101	6
r	2.58	01001	5
s	0.86	1001001	7
t	3.43	10100	5
u	1.72	101010	6
v	0.43	11101010	8
h	1.29	010100	6
y	0.43	10010101	8
0	4.38	0001	4
1	2.62	01011	5
2	4.85	0011	4
3	3.05	01111	5
4	2.36	01000	5
5	3.26	10000	5
6	2.75	01101	5
7	2.62	01100	5
8	1.76	101100	6
9	1.97	101101	6
{	0.43	10010110	8
'	15.45	110	3
:	3.86	11100	5
space	8.58	111	3
(	0.86	1001100	7
-	0.86	1001101	7
.	3.86	10111	5
,	4.72	0010	4
)	0.86	1001110	7
}	0.43	10010100	8
\n	0.42	111010110	9

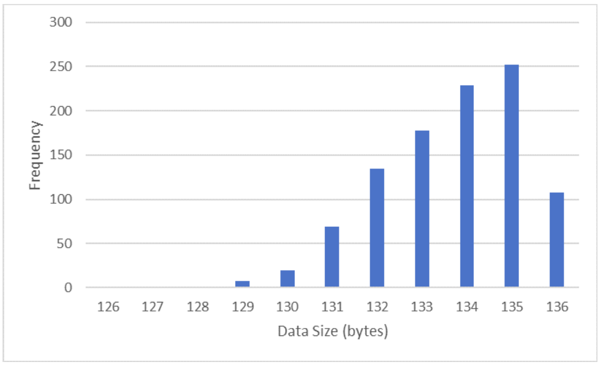

Fig. 5. Size of transmitted data using ASCII format (total $1,000$)

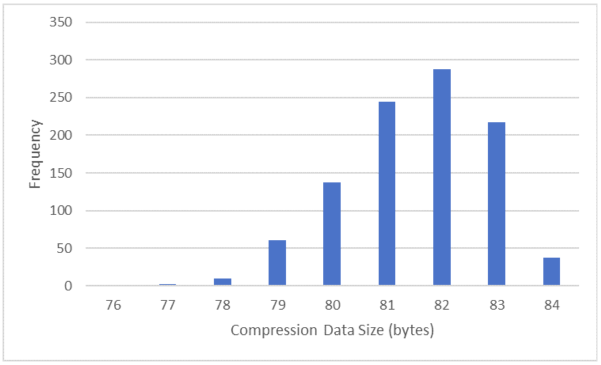

Fig. 6. Size of transmitted data using Huffman codes (total $1,000$)

V. Evaluation and Discussion

The performance evaluation on data compression and transmission error are presented. It compares results when sending data in a traditional ASCII format with those in the generated Huffman codes as follows.

A. Compression Performance

We can calculate the expected length of Huffman codes generated from our data transmission by

$$\text{Expected Length} = \frac{\sum \text{Frequency} \times \text{Code Length}}{\text{Total Frequency}} \qquad (1)$$

From Table IV, the expected Huffman code length is 4.71 bits. Based on the average data size for an 8-bit ASCII format, the implementation of Huffman codes yields the expected number of $\frac{(4.71)(133.68)}{8} = 78.07$ characters for each transmission. This expected value can be viewed as a lower bound for transmission size using Huffman codes. To illustrate the coding performance, we obtain the compression percent (P_c) by

$$P_c = \frac{(\text{Orignal size} - \text{Compressed size})}{\text{Orignal size}} \times 100\% \qquad (2)$$

In comparison, we see that using Huffman codes in this LoRa application can reduce data transmission size. The theoretical or expected number of data size is 78.07 characters, while our experiment yields an average of 81.53 transmitted characters. The compression percentages are 41.60% and 39.01%, respectively for theoretical and experimental data. Table V shows a comparison of the average number of data size when transmitting with the ASCII format and the Huffman codes.

TABLE V
Comparison of Average Transmitted Characters

Data Scheme		Average Size	P_c
ASCII Format		133.68	–
Huffman	Expected	78.07	41.60%
	Experiment	81.53	39.01%

B. Error Performance

Recall that data transmission takes place at 4 different distances. Each location sends data using both a typical ASCII format and the Huffman code. To evaluate data transmission performance, the numbers of error or invalid data are observed and tabulated. Table VI shows the average number of transmission error at all 4 ranges where 250 data are transmitted. We see that there are no error occurred when transmitting data at distances of 315 and 500 meters, regardless of coding schemes. At 750 meters distance, the average numbers of error are 22.46 (or 16.80%) and 18.51 (or 13.84%) characters for an ASCII format and the Huffman codes, respectively. Similarly at $1,000$ meters distance, the average numbers of error increase to 33.12 (or 24.77%) characters for an ASCII format and 27.63 (or 20.66%) characters for the Huffman codes.

TABLE VI
Average Number of Transmission Error

Distance (m)	Average Data Error (bytes)	
	ASCII Format	Huffman Code
315	0	0
500	0	0
750	22.46	18.51
$1,000$	33.12	27.63

C. Discussion

In this paper, only 100 sets of measured data are pre-collected and used as a representation of the transmission data source. The occurrence of send characters is an approximation of a true statistics and may not contribute to the shortest Huffman codes. However, the experimental data results in the Huffman codes with an average code length of 81.53 characters closer to the expected value of 78.07 characters, in comparison with the average 133.68 character from the ASCII format. These average values suggest that using the Huffman codes can reduce a transmission data size. In particular, this experiment indicates that the data is compressed down to an approximately 40% from the original size.

In terms of transmission errors, we see that the Lopy and Pysense development boards can send data without errors in the shorter distances below 500 meters. A number of errors are increased as the transmission distances becomes longer. The reliability of data transmission is improved about 4% (resulting in average of 5 error characters less) when the Huffman codes are implemented. These errors are perhaps originated from noises and interference due to building and tree obstruction and a signal multi-path.

Furthermore, we notice from experiment that the overall transmission time when using the Huffman codes is relatively equal to that from the ASCII format. The processing time to encode and decode Huffman algorithm does not affect the LoRa transmission time.

VI. Conclusion

In this research, we successfully implement the Huffman codes onto measured sensor data set and transmitted them via the LoRa devices. Data transmission using the Huffman codes result in shorter data sizes than those with a traditional ASCII format. The size of compressed data uses about 40% less characters on average. This results implies that communication bottle neck of data transmission from many devices simultaneously can be improved with the Huffman code implementation.

As we transmit data in longer ranges, the errors are generally increasing. However, we see that using the Huffman codes can reduce the average number of transmission errors. At $1,000$ meter range, the errors decrease from 24.77% down to 20.66%.

For future work, we can obtain a shorter average transmission data size by removing some redundancies. Since the measured sensor data are ordered in a pre-defined format, we can eliminate the uses of characters to identify each data type. Therefore, only measured data values are utilized in order to generate the Huffman codes.

References

[1] LoRa Alliance, “What is the LoRaWan,” available at https://lora-alliance.org/about-lorawan.

[2] K. Tzortzakis, K. Papafotis, and P.P. Sotiriadis, “Wireless self powered environmental monitoring system for smart cities based on LoRa,” Panhellenic Conference on Electronics and Telecommunications (PACET), Nov 2017, Greece.

[3] K. Sharma, and K. Gupta, “Lossless data compression techniques and their performance,” International Conference on Computing, Communication and Automation (ICCCA), May 2017, India.

[4] NBTC, “Office of The National Broadcasting and Telecommunications Commission,” available at https://www.nbtc.go.th.

[5] Lopy, “Lopy Datasheet,” available at https://docs.pycom.io/datasheets/development/lopy.html

[6] Pysense, “Pysense Datasheet,” available at https://docs.pycom.io/datasheets/boards/pysense.html

[7] S. Haykin, “Digital Communications,” Wiley, 2006.

[8] A.C. Petrini and V.M. Ionescu, “Implementation of the Huffman coding algorithm in Windows 10 IoT core,” 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), July 2016, Romania.

[9] Semtech, “Semtech SX1272” available at https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1272