Islamic university Of Technology
Department Of Computer Science and Engineering (CSE)

Enhanced Structural Health Monitoring by Payload Compression in Wireless Sensors Network

Authors:

Ngnamsie Njimbouom Soualihou (124455)

Arham Al Ahmad Siddiquee (124434)

Supervisor:

Dr. Md. Motaharul Islam
Assistant professor
Department of Computer Science and Engineering

A thesis submitted to the Department of CSE
in partial fulfilment of the requirements for the degree of
B.Sc. Engineering in CSE
Academic Year: 2015-2016

Table of contents

… To Almighty Allah

Acknowledgements

Thanks to Allah (SWT), the most gracious and most merciful. Peace and blessing of Allah be upon the last and final messenger Muhammad. The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people whose ceaseless cooperation made it whose constant guidance and encouragement crown all efforts with success.

We would like to express our heartfelt thanks to our supervisor Dr. Md. Motaharul Islam for his continuous and invaluable support throughout our thesis work. His constructive ideas helped us a lot to find right way in our research.

At last, we would like to thank our parents who have given us tremendous inspiration and support. Without their constant moral and financial support we would not have been able to accomplish this work.

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and investigation carried out by **Ngnamsie Njimbouom Soualihou** (**124455**) and **Arham Al Ahmad Siddiquee** (**124434**,) under the supervision of Dr.Md.Motaharul Islam in the Department of Computer Science and Engineering (CSE), IUT. Dhaka, Bangladesh. It is also declared that neither of this thesis or any part of this thesis has been submitted anywhere else for any degree or diploma, Information derived from the published and unpublished work of others has been acknowledged in the text and a list of reference is given.

Authors:

Ngnamsie Njimbouom Soualihou (124455)

Arham Al Ahmad Siddiquee (124434)

Supervisor:

Dr.Md.Motaharul Islam

Assistant Professor

Department of Computer Science and Engineering

Islamic University of Technology (IUT)

Abstract

Structural health monitoring is the fact of estimating the state of structural health, or detecting the changes in structure that affect its performance. The traditional approach to monitor the structural health is by using centralized data acquisition hub wired to tens or even hundreds of sensors, and the installation and maintenance of these cabled systems represent significant concerns, prompting the move toward wireless sensor network. As cost effectiveness and energy efficiency is a major concern, our main interest is to reduce the amount of overhead while keeping our health monitoring accurate. Our contribution to this structural health monitoring will be to reduce the amount of data to be transmitted that is by compressing the payload of the packets using a compression algorithm and by doing so, we will extend the lifetime of the battery, thus the capacity of the sensor node will increase since less time will be taken to transmit data between intermediate node as well as the node coordinator..

Chapter 1

Introduction

Wireless sensor network consists of a group of tiny sensor nodes, distributed in a wide geographic area, forming an ad-hoc network, collecting and conveying information regarding the area under surveillance. The data collected by these sensor nodes is aggregated and analyzed at a more capable node called as sink or gateway or base station. Wireless sensor network combine simple wireless communication, minimal computation facilities and some sort of sensing of the physical environment and this leads to a new paradigm of networks that can be deeply embedded in our physical environment, fueled by the low cost and wireless communication facilities.

Structural Health Monitoring is becoming a popular area of research as it offers opportunities in construction management and maintenance. As a result we can create post-disaster scenarios and rescue support. Thus SHM is a multidisciplinary field where a number of different skills such as seismology, electronic and civil engineering, computer science etc. and institutions can work together to increase the performance and reliability of such systems.

The traditional approach to structural health monitoring involves conventional piezoelectric accelerometers hardwired to data acquisition boards residing in a PC. The drawbacks of such a system are (1) the high cost of installation, (2) the high cost of equipment and (3) cost of maintenance [1].

The Wireless approach to SHM offers the same functionalities as the wired approach. Besides WSN approach offers various advantages over the wired approach such as a wireless system and installation and maintenance cost reduction. Despite all advantages of the wireless system, it still needs a better compression algorithm at IP payload level. Hence we propose a better payload compression for structural health monitoring in WSN.

The wireless sensor network consisting of low power devices has to communicate with other regular devices and to make this possible 6LoWPAN was introduced. In 6LoPAN only the header is compressed. To make the system more efficient in term of power consumption and the amount of data sent, we propose IP payload compression using Arithmetic algorithm.

Chapter 2

Background and Related Works

Reliable, data compression for wireless sensor networks is an ongoing area of research. Recently several reliability protocols have been proposed. WISDEN which is a reliable data transport using a hybrid of end-to-end and hop-by-hop recovery, and low-overhead data time-stamping that does not require global clock synchronization in which data compression is been made using run length encoding This paper also study the applicability of wavelet-based compression techniques to overcome the bandwidth limitations imposed by low-power wireless radios. The architecture of Wisden was simple, a base station centrally collecting data) its design was a bit more challenging than that of other sensor networks built till date. Structural response data is generated at higher data rates than most sensing applications. Furthermore, this application required loss intolerant data transmission, and time synchronization of readings from different sensors. The relatively low radio bandwidths, the high packet loss rates observed in many environments, and the resource constraints of existing sensor platforms added significant challenges to this system [2]. Wisden used a vibration card, especially designed for structural applications. In addition to describing this card, the description of Wisden focused on its three novel software components:

Reliable Data Transport Wisden used existing topology management techniques to construct a routing tree, but implemented a hybrid error recovery scheme that recovers packet losses both hop-by-hop and end-to-end.

Compression Wisden used a simple run-length encoding scheme to suppress periods of inactivity in structural response, but it also evaluated the feasibility of wavelet compression techniques to reduce Wisden's data rate requirements and to improve latency.

Data Synchronization Wisden also implemented a data synchronization scheme that requires little overhead and avoids the need to synchronize clocks network-wide [2].

An accurate data acquisition system, high-frequency sampling with low jitter and time synchronized sampling were not provided in Wisden [1].

In [1] A Wireless Sensor Network (WSN) for Structural Health Monitoring (SHM) was designed, implemented, deployed and tested on the 4200ft long main span and the south tower of the Golden Gate Bridge (GGB). Ambient structural vibrations were reliably measured at a low cost and without interfering with the operation of the bridge. In the GGB deployment, 64 nodes were distributed over the main span and the tower, collecting ambient vibrations synchronously at 1 kHz rate, with less than 10μs jitter, and with an accuracy of 30μG. The sampled data was collected reliably over a 46-hop network, with a bandwidth of 441B/s at the 46th hop. The collected data agrees with theoretical models and previous studies of the bridge. The deployment is the largest WSN for SHM [1].

In this work a small packet size was a bottleneck for network data transmission bandwidth, but increasing packet size was not a good solution for the Mica motes due to the limited amount of available RAM; a limitation resulting from an unshared buffer pool.

In [3] WSN for structural health monitoring in which Huffman coding technique was used. In this work, sensor node were collecting data, and also processing the data package. They implemented this algorithm into the sensor node to reduce the packet size to be transmitted. In this work, an on-site WSN-based structure health monitoring of Chung-Sha Bridge (Taiwan) was implemented. The implemented WSN monitoring system successfully achieved the frequency analysis of the bridge structure by monitoring with 128Hz sampling rate from end nodes. A local-data processing node with ARM Cortex M3 processor was developed. Based on this local-data-processing node, Huffman compression algorithm was implemented and examined. The experimental results showed that the wireless transmission payload was reduced by 60% and node number of the implemented network could be increased by 3 times [3].

Both these two works are concerned with the performance. Wireless sensor networks in Structural Health Monitoring based on ZigBee technology and TPSN (Timing-sync Protocol for Sensor Network), ZigBee was used because it is the most popular low-cost, low-power wireless mesh networking standard available. It is Suitable for complex networks with large spatial extension, multi-hop networks and proprietary metering and automation solutions. But one of it drawback is that the interoperation between Zigbee devices and IP based devices [4]. So our approach is to use WSN in which 6LOWPAN is used as an adaptation layer along with Payload compression using arithmetic compression.

"6LoWPAN (Montenegro et al., 2007)" refers to "IPv6 over Low-Power Wireless Personal Area Network". It defines an adaptation layer which allows transportation of IPv6 packets over IEEE 802.15.4 links. 6LoWPAN reduces the IPv6 packets to fit

within the MTU (127 bytes) of IEEE 802.15.4 frames. To do that 6LoWPAN uses header compression and fragmentation and reassembly schemes.

Chapter 3

Proposed architecture

3.1 Schematic Diagram of Our Wireless Sensor Network

Sensor Network is made up of many sensor nodes. These nodes are tiny in size. They have limited power supply, memory, and processing capability. Each sensor consists of, Transceiver, Power supply, Processor, Memory and Sensors. The overall architecture of our sensor network is shown in the diagram below.

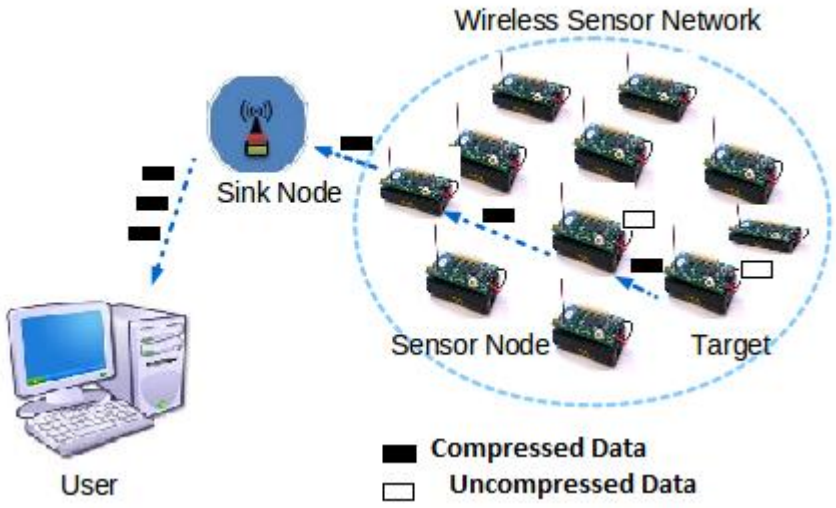


Figure3-1: Architecture of a Wireless Sensor Network

The sensor nodes are connected to one another in mesh topology fashion. Each node compresses the data using the arithmetic compression algorithm, and forward it to the sink node. The compression of data in each node i.e. local processing of data compresses the data further and thus saves power as well as increases the capacity of the nodes in the network. The target node, passes the compressed data to the sink node, and a further compression on that received data is being perform again, and then, sent to the user's terminal. So, the sink node has more computation power because of the reduce amount of transfer it has to make.

3.2 Application of WSN in Structural Health Monitoring

Structural health monitoring is an innovative method of monitoring structural safety, integrity, and performance without, otherwise, affecting the structure itself. Structural health monitoring utilizes wireless sensor networks (WSNs) to detect the presence, location, severity, and consequence of damages. In many monitoring allocations, the conventional usages of WSNs are cases with, low data rate, small data size, low duty cycle, and low power consumption. However, structural health monitoring requires high data rate, large data size, and a relatively high duty cycle.

Structural Health Monitoring (SHM) aims to give, at every moment during the life of a structure, a diagnosis of the state of the constituent materials, of the different parts, and of the full assembly of these parts constituting the structure as a whole. Structural Health Monitoring combines, a variety of sensing technologies with an embedded measurement controller to, capture, log, and analyze real-time data. SHM systems are designed to reliably monitor, test the health and performance of structures such as the following:

- Bridges and dams
- Buildings and stadiums
- Vessels and platforms
- Airframes
- Wind turbines

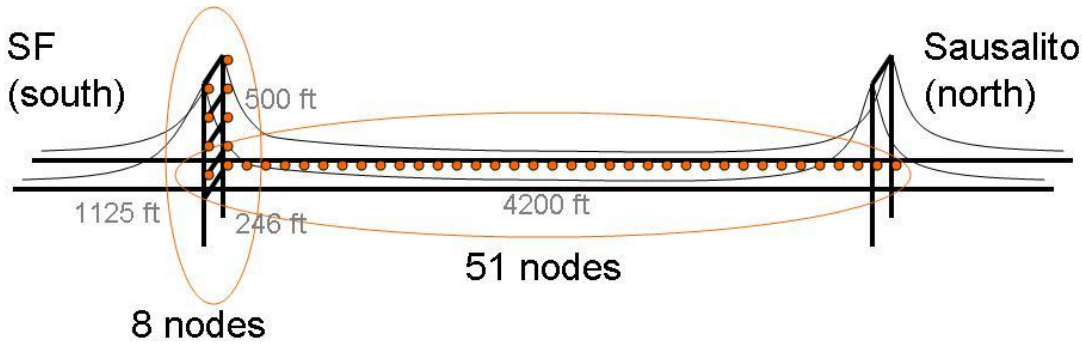


Figure 3.2: Deployment of Wireless Sensors for health monitoring of a bridge

3.3 Compression in WSN

Wireless sensor networks (WSNs) have many important applications, such as, climate change tracking, critical infrastructure monitoring, and wide area surveillance. In many of these applications, relatively simple sensors gather data that are then transmitted to centralized nodes, or *sinks*, where they are analyzed and processed. The Compressed data, as they are transmitted to the sink, can play a potentially important role in increasing the lifetime of the sensors by reducing the number of bits that need to be transmitted, thus, lowering overall power consumption.

Depending on the nature of the application, there are various criteria to measure the performance of a compression algorithm. When measuring the performance, the main concern would be the space efficiency. The time efficiency, is another factor which is considered by many other researcher leaving behind every other aspects of a compression algorithm. Since the compression behavior depends on the redundancy of symbols in the measured data, it is difficult to measure performance of a compression algorithm in general. The performance depends on the type and the structure of the input source. Additionally, the compression behavior depends on the category of the compression algorithm: lossy or lossless. If a lossy compression algorithm is used to compress a particular source file, the space efficiency and time efficiency would be higher than that of the lossless compression algorithm. Thus measuring a general performance is difficult, and there should be different measurements to evaluate the performances of those compression families. Following are some measurements used to evaluate the performances of lossless algorithms.

Compression Ratio: is the ratio between the size of the compressed file and the size of the source file.

$$compression\ ratio = \frac{size\ after\ compression}{size\ before\ compression}$$

Compression Factor: is the inverse of the compression ratio. That is the ratio between the size of the source file and the size of the compressed file.

$$compression\ factor = \frac{size\ before\ compression}{size\ after\ compression}$$

Saving Percentage: calculates the shrinkage of the source file as a percentage.

$$saving\ percentage = \frac{size\ before\ compression - size\ after\ compresion}{size\ before\ compression}\%$$

All the above methods evaluate the effectiveness of compression algorithms using file sizes. There are some other methods to evaluate the performance of compression algorithms. Compression time, computational complexity and probability distribution are also used to measure the effectiveness.

Compression Time

Time taken for the compression and decompression should be considered separately. Some applications like transferring compressed video data, the decompression time is more important, while some other applications both compression and decompression time are equally important. If the compression and decompression times of an algorithm are less or in an acceptable level, it implies that the algorithm is acceptable with respective to the time factor. With the development of high speed computer accessories, this factor may give very small values and those may depend on the performance of computers.

Entropy

Entropy is the expected value (average) of the information contained in each message. This method can be used if the compression algorithm is based on statistical information of the source file. Self-Information is the amount of one's surprise evoked by an event. In another words, there can be two events: first one is, an event which frequently happens and the other one is, an event which rarely happens. If a message says that the second event happens, then it will generate more surprise in receivers mind than the first message. Let set of event be *S = {s1, s2, s3...sn)* for an alphabet and each sj is a symbol used in this alphabet. Let the occurrence probability of each event be pj for event sj. Then the self-information I(s) is defined as follows.

$$I(S_j) = log_b \frac{1}{p_j} \qquad or \qquad I(S_j) = -log_b \frac{1}{p_j}$$

Code Efficiency

Average code length is the average number of bits required to represent a single code word. If the source and the lengths of the code words are known, the average code length can be calculated using the following equation.

$$\bar{l} = \sum_{j}^{n} p_j l_j$$

3.4 Types of Compression

In WSNs, the main objective of compression is to reduce energy consumption. Sensing/sampling, computation, and communication are the three operations mainly responsible for energy consumption in WSNs. Any technique that directly or indirectly reduces one or more of these operations, while meeting application requirements (e.g., distortion, complexity), can be considered as compression. Based on this, compression in WSNs can be classified as follows.

Sampling Compression (SC): SC is the process of reducing the number of sensing/sampling operations while keeping network coverage (for spatially or related sensors) and /or distortion loss within an acceptable margin. A number of research works exploit spatially correlated data to reduce the sensing tasks. These works primarily focus on keeping the sensors in a sleep state, while a minimal number of sensors are kept active within a group [18]. In contrast, CS approaches perform the sampling-level compression by exploiting temporal data correlations at a sensor node.

Data Compression (DC): SC is the process of reducing the number of sensing/sampling operations while keeping network coverage (for spatially or related sensors) and /or distortion loss within an acceptable margin. A number of research works exploit spatially correlated data to reduce the sensing tasks. These works primarily focus on keeping the sensors in a sleep state, while a minimal number of sensors are kept active within a group [18]. In contrast, CS approaches perform the sampling-level compression by exploiting temporal data correlations at a sensor node.

Communication Compression (CC): Typically, this is the process of reducing the number of packet transmissions and receptions, hence reducing the radio on-time of transceivers a WSN. The longer the packet to be transmitted or received, the greater the radio on-time of transceivers [18]. Hence reduced packet or data size (e.g., data compression) reduces radio on-time and reduces communication cost in WSNs. Aggregation, DCS, and predictive coding support communication-level compression.

Typically, this is the process of reducing the number of packet transmissions and receptions, hence reducing the radio on-time of transceivers a WSN. The longer the packet to be transmitted or received, the greater the radio on-time of transceivers [12].

Hence reduced packet or data size (e.g., data compression) reduces' radio on-time and reduces communication cost in WSNs. Aggregation, DCS, and predictive coding support communication-level compression.

Usually, there is a hierarchical relationship between the aforementioned types of compressions For instance, a reduced number of samples helps in reducing the data/packet length (data compression), which ultimately reduces the radio on-time of the transceivers (communication compression). It is desirable to have compression techniques, which support these three levels of compressions.

3.5 Design Issues in WSN

Wireless sensor networks are made of large number of tiny sensor nodes, which have limited power and less processing capability. The life time of the individual sensor node is not easily predictable and also the network needs to be formed autonomously as it is not possible to manually set up the sensor network for all applications. The sensor network also consists of several different kind of nodes hence heterogeneity needs to be supported. The number of sensor nodes in the network is not constant throughout the life time of the network it may vary because of addition of senor nodes or reduction of nodes due to their death. The major factors that need to consider while designing sensor network are listed below.

- **Fault Tolerance:** Possibility of node failure and change of topology of network is quite high in case of WSN. Hence the designer of network should make the network robust and reliable even in case of node failures and topology changes. The network should function smoothly and normally irrespective of node failures and topology changes.

- **Life Time:** WSN are supposed to work for a quite long time with low power consumption. They are supposed to last at-least for 6 months to 1 year. We need to keep in mind that every node in WSN may be powered using just a 3 V battery and this should be sufficient for the entire life time of the node. The design of protocols of WSN should be such that the node consumes as less energy as possible. This will help in making the WSN last longer.

- **Scalability:** The design of WSN should support addition of new nodes any time and also the design should support large number of nodes because some applications in WSN may require quite a huge number of sensor nodes.

- **Data Aggregation:** The sensor nodes in WSN are located close to each other hence the possibility of similar data being generated by the nodes next to each other is quite high. So the data needs to be aggregated and the duplicate data needs to be avoided because the transmission and reception data is the most costly affair in WSN. The data needs to be aggregated at different levels in WSN so that only the necessary data is transmitted and received and the redundant data is not communicated.

- **Cost:** The cost of each sensor node is supposed to be 1$, as WSN can have large number of sensor nodes the total cost of the network can become a quite expensive affair. So the designer of WSN needs to decide on the optimal number of nodes necessary for the application.

- **Environment:** The environment in which the WSN is deployed can be very demanding, so the design of WSN should be such that WSN should be able to survive regardless of the conditions in which WSN s deployed.

- **Heterogeneity Support:** The protocols designed for WSN should support different kinds of sensor nodes and also be able to support variety of applications.

- **Autonomous Operations:** The WSN should be able to organize, reorganize and operate autonomously because sometimes WSN deployed in places where human habitation is not possible.

- **Limited Memory and Processing Capability:** The sensor nodes have very limited memory, power and processing capabilities, so all designs of WSN should not be demanding in terms of processing requirements or memory requirements

Chapter 4:

Evaluation

4.1 Algorithm Analysis

In this algorithm the symbols are not replace by some code words. Instead, the symbol in the data are being assigned, values, based on some mathematical model; the Arithmetic algorithm can treat the whole symbols in a list, or, in a data message to be transmitted as one unit. It doesn't use a discrete number of bits or some frequency distribution for the data's symbols [6], instead each symbol is assign an interval starting with the interval [0….1); So at the beginning the probabilities of occurrence of a set of symbol together with the cumulative probabilities are taken into consideration; this cumulative probabilities are used for encoding and decoding.

4.1.1 The Encoding Process

The first step is to calculate the cumulative probabilities and then make ranges based on the obtained results. When we read a character, its range is considered as the new cumulative range on which our encoding will be done, that range is so divided into the sub parts, according to the probabilities of occurrence of the character being encoded; and the next symbol is read, and this process of sub part formation is repeated again and this goes on as long as we have a character to encode in our source data. Once the end of our source data is found, we take a fraction of our sub part range formed. Therefore using a fraction taken from our range we can represent our entire source data into binary form. Let's consider the following example [7]. From the encoded interval [0.6607, 0.66303).A sequence of bits are assigned to a number that is located in this range.

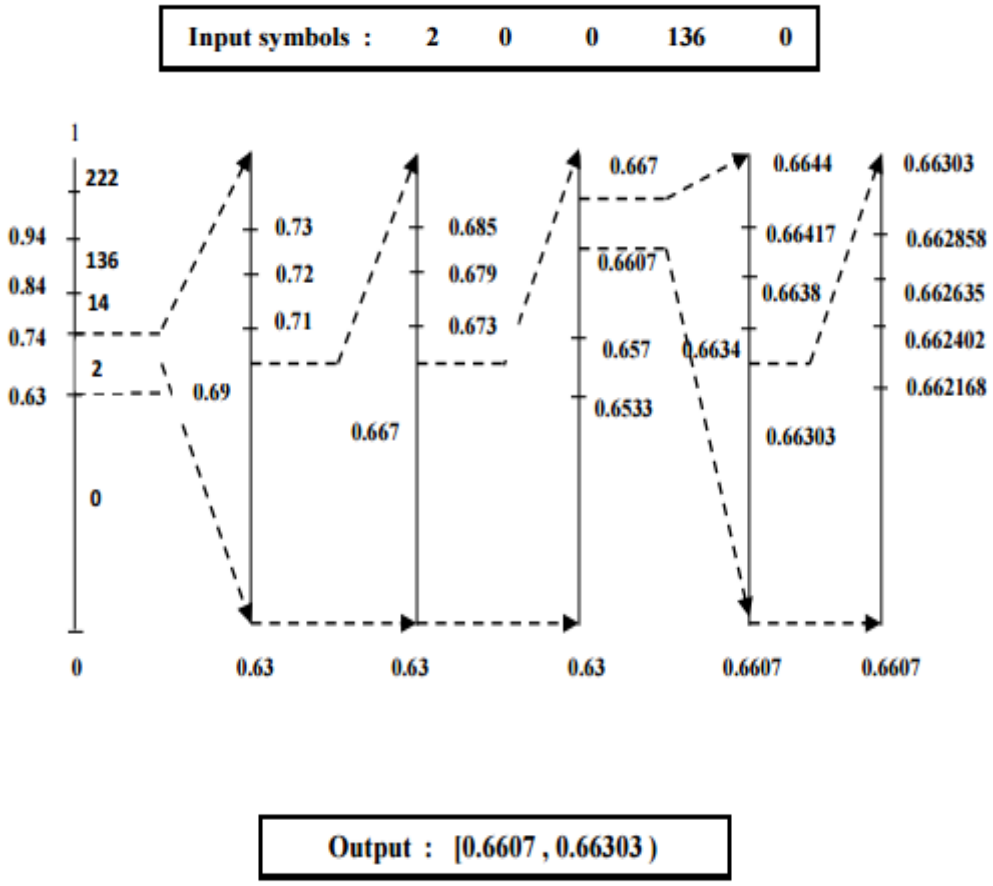


Figure 4-1: Encoding process

4.1.2 The Decoding Process

Once the compress data reach the other end, it needs to be decompressed and this is being done by following some mathematical model, and then, applying the inverse process of the encoding. But to do that, the receiver needs to know the number of symbols that were sent as well as the probability / frequency distribution.

4.2 Performance Evaluation

In performance evaluation we have used the Huffman Coding as a reference as this algorithm was used in SHM [3]. The performance is measured in terms of Compression ratio, data size, memory consumption and compressed data size.

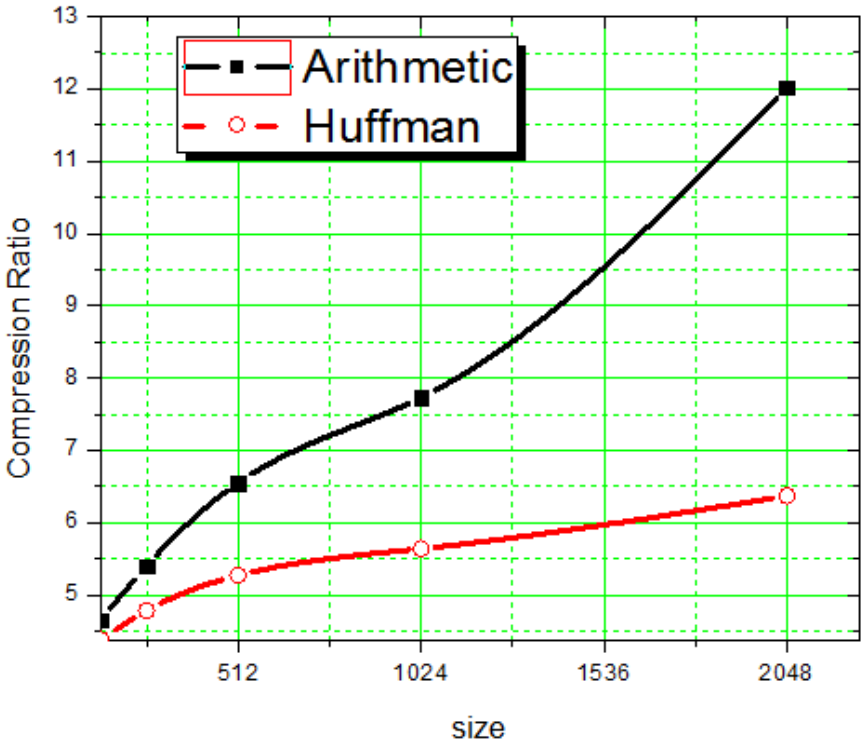


Figure 4-2: Algorithm Comparison (CR vs. Size)

In the above Figure 4-2, we compared the compression ratio (CR) of the Huffman coding with that of the arithmetic coding. It is been observed that, with the increasing in the size of the data, the compression ratio of arithmetic coding gets better than that of Huffman coding, and the compression ratio of arithmetic is almost more than the double of that of the Huffman coding as the data get higher. On the other hand Huffman Coding does not give any better compression with the increase of size. We would like to mention that the data size in SHM is large and varied. Which is suitable for our algorithm.

The following Table 4-3 show the data's result used to draw the above graph.

File Size	Compression Ratio	
	Arithmetic	Huffman
128	4.65	4.38
256	5.4	4.78
512	6.55	5.27
1024	7.73	5.64
2048	12.02	6.37

Table 4-3: Comparison between Huffman and Arithmetic

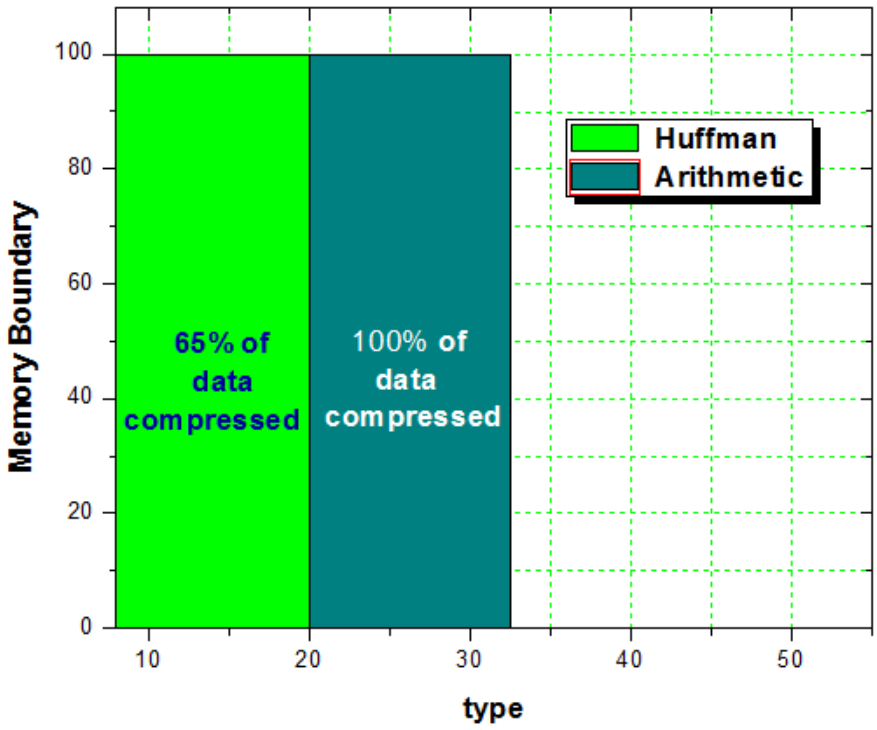


Figure 4-3: comparison of the data capable to be compressed by both algorithm

The above graph (Figure 4-3), shows, the amount of data capable to be compressed given the same memory boundary. It is seen from the graph that, the Arithmetic coding, can compress the whole data (**100%**) provided to it at a specific time, while the Huffman coding, can only compress **65%** of the data provided to it and then the node running this algorithm will run out of memory. The compression ability of arithmetic algorithm when provided with the same memory block is better than that of the Huffman.

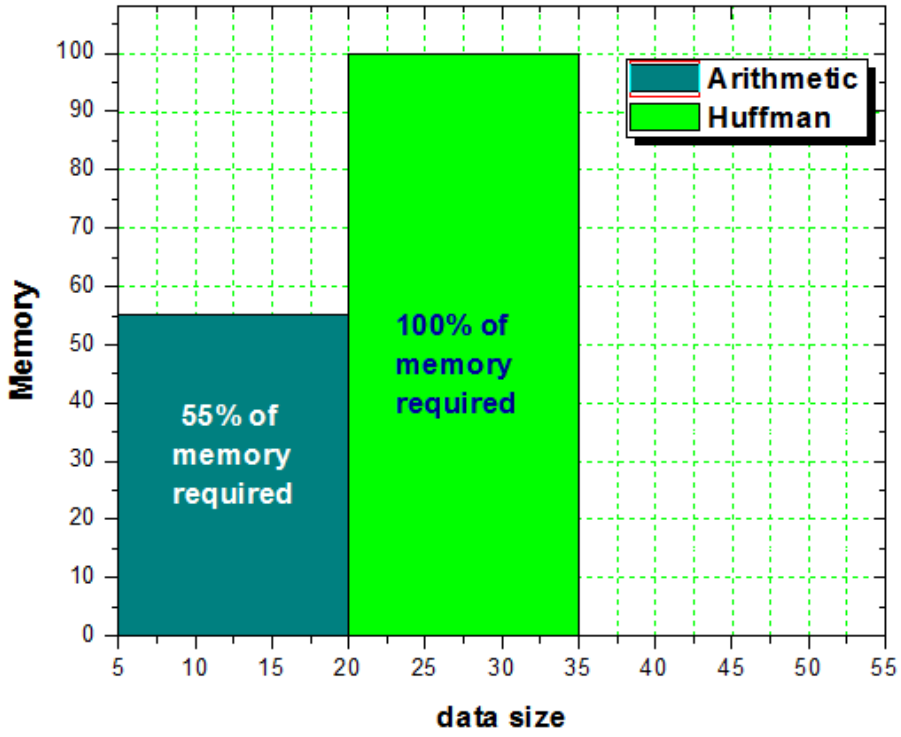


Figure 4-4: comparison of the memory need for computation

The above Figure 4-4, gives us an idea about, the memory utilization of Huffman and that of Arithmetic algorithm. From the graph, we can see that, the Arithmetic algorithm uses much less memory compared to the Huffman algorithm requires for its computation. So, in case of memory space utilization, arithmetic coding is a much better choice for compression. As the resource of sensor node is very scarce, every bit of resource utilization is critical.

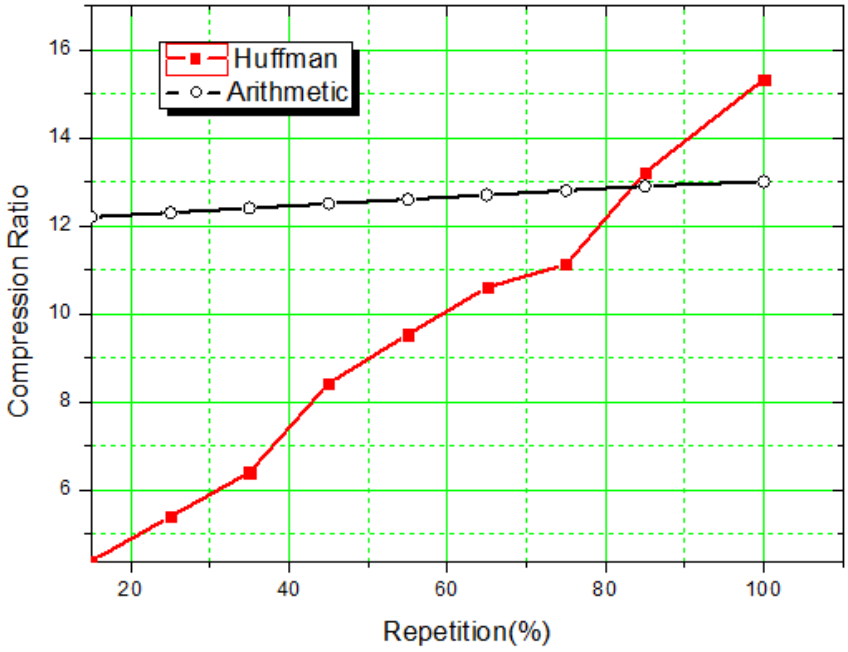


Figure 4-5: comparison of the (CR) based on the repetition of data's symbols

On the above Figure (Figure 4-5), In case of Huffman coding, the compression ratio gets better when there is more repetition of the values in the sense data. But in case of Arithmetic algorithm there's not much change due to repetitive or non-repetitive data. As in case of SHM, the data we receive from the sensors are varied, and there is very less repetition of data. So, for this approach Arithmetic algorithm gives a better compression ratio than the Huffman algorithm.

The following Table 4-5, shows the data used to generate the above graph.

Repetition	Compression Ratio	
	Huffman	Arithmetic
15	4.38	12.2
25	5.39	12.3
35	6.4	12.4
45	8.42	12.5
55	9.53	12.6
65	10.6	12.7
75	11.12	12.8
85	13.21	12.9
100	15.32	13

Table 4-5: Comparison between Huffman and Arithmetic (Repetitive Data)

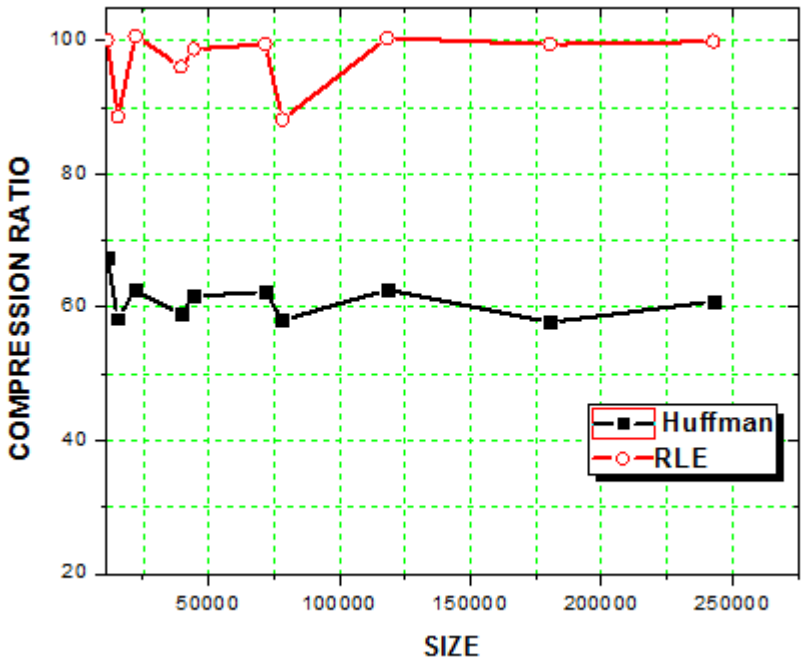


Figure 4-6: comparison of the CR between Huffman and RLE algorithm.

The above graph, Figure 4-6, shows, the comparison between Run Length Encoding (RLE) and Huffman Encoding with respect to their compression ratio. As depicted in the graph, we see that the compression ratio of RLE is much higher than that of Huffman Encoding. While Huffman coding is being considered by many other researcher. This graph shows that the RLE is much better than the Huffman.

COMPRESSION METHOD	ARITHMETIC	HUFFMAN
Compression Ratio	Very good	Poor
Compression Speed	Slow	Fast
Decompression Speed	Slow	Fast
Memory Space	Very low	Low
Compressed pattern matching	No	Yes
Permits Random access	No	Yes

Table 4-6: comparative analysis of Arithmetic and Huffman algorithm is as follows:

The above **Table 4-6**, is a comparison between the Huffman and arithmetic algorithm. It compares the two algorithm based on many factor. The most important factors that we considered while doing our thesis work, was, the compression ratio and the memory space, where we can see the compression ratio of the arithmetic algorithm being very good, while that of the Huffman is poor. For the other factor: memory space, we can see that the arithmetic algorithm required, a very low amount of space for it computation while the Huffman required more.

4.3 Pseudo code of Implementation

```
main()
{
initial_model_paramameter();
initial_ACmodel();
create_expect();
number_value=0;
for (;;)
{
read_sf_packet();
acdecoder();
number_value++;
create_observe();
if (number_value >= UPDATE_PERIOD){
sigma_change=check_sigma();
if (sigma_change == 1){
sigma_current=sigma_new;
create_expect();
update_flage = 1;
}
dist_change=check_dist();
if (dist_change == 1){
dist_current=dist_new;
update_flage = 1;
}

If (update_flage == 1) {
prepare_packet();
update_ACmodel();
write_sf_packet();
}
update_flage = 0;
number_value=0;
}
}
```

Chapter 5

Conclusion

Wireless sensor network is getting popular day by day. Its mobility and capability of real time communication has led many application use this technology. Many developers and researchers are providing their support to enhance this sector. Many new protocols, architectures are being proposed in this purpose.

Among many application areas, Structural Health Monitoring is very important. SHM has many advantages and it can be used to save both money and lives of the people of the country. By monitoring the important structures and by renovating or maintaining these structures when necessary can save a lot of money as well as keep the structures fit. Besides it can save many lives which are lost in different natural disasters.

Using WSN technology in SHM can save lives and make the system less costly and affordable. The structure's health can be easily monitored, as well as structures in the remote areas can be monitored and can be easily maintained at a very low cost which will save lives in case of emergencies.

One of the main disadvantages of the WSN node are, the power constraint (because small size implies small battery) and limited memory. It is useful to mention that WSN node can be deployed in extreme environmental conditions and no human can go over to change a damaged node. So we need to make use of the node as efficiently and as long as possible. So, the data should be transferred efficiently between sensors.

The compression algorithm and technique we proposed will increase the life time of the sensors as less data will be sent. Besides, the local processing will increase the life time as well as the number of sensors deployed in the network. We have chosen the arithmetic algorithm over other type of algorithm because of the advantages that it was offering on the compression ratio plan as well as the memory required for the processing of data. By observing the results generated by all the algorithms in our work as mention in the chapter 2. We can easily show that arithmetic algorithm is far more better that the other algorithm.

5.2 Limitation

Throughout our work we faced many problems and the one that almost stopped us from completing our thesis work was the NesC language itself. As NesC was totally new to us we couldn't totally complete the implementation of the arithmetic code.

During our implementation, we came across a network data-type error, which could not be complete by us. Even with the help of different resources that we could find, we were unable to complete it before the thesis defense day. And it has stop us from moving with our work.

5.3 Future works

As mention in our limitation the NesC language stopped us from continuing with our work, so we have a mission to go deeper in the research for this data-type error problem that, once solved will allow us to complete our work, generate more data from the algorithm, and the draw additional graph. And with the permission of our supervisor, we will submit the work in a journal, or a conference.

Appendixes

A. Tools:

- Eclipse IDE
- TinyOs plugins
- Nesc(programming language)

B. Implementation:

The implementation was done as follow.

- The make file: which consisted only of the rule that should be applied to the other file.
- A Header file: where the main structure of our code was written.
- The Module file: was the file that contains all the interfaces as well as most of the source code
- The Configuration file: this file contains the code necessary to wired all the components of our program together.

C. Source Code snipped

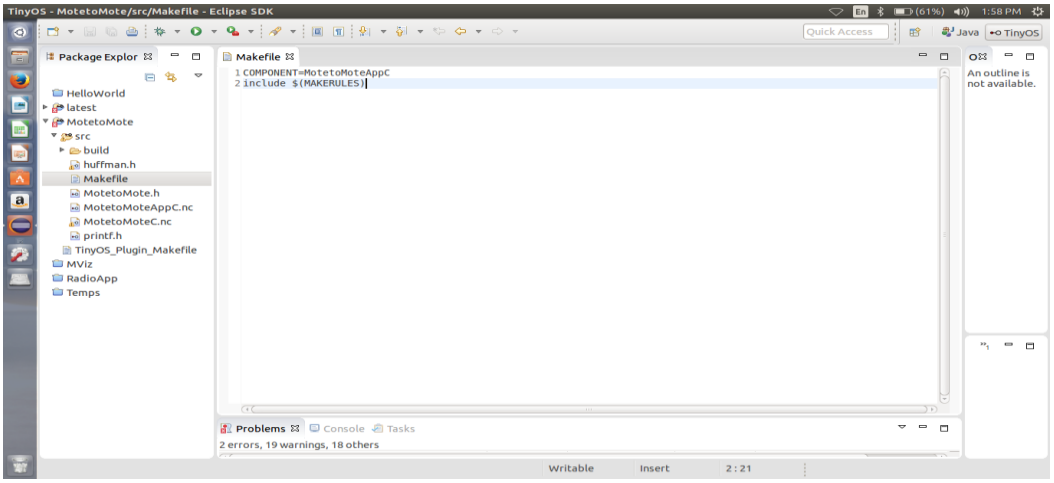


Fig1: figure of the make file source code

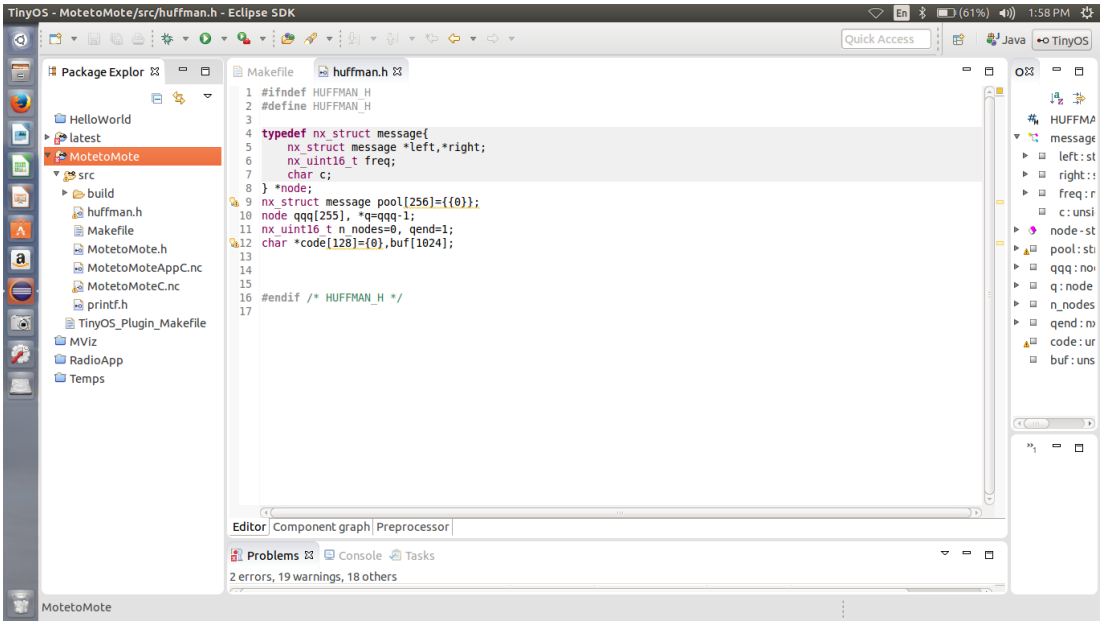


Fig2: snipped source code of the Huffman main structure

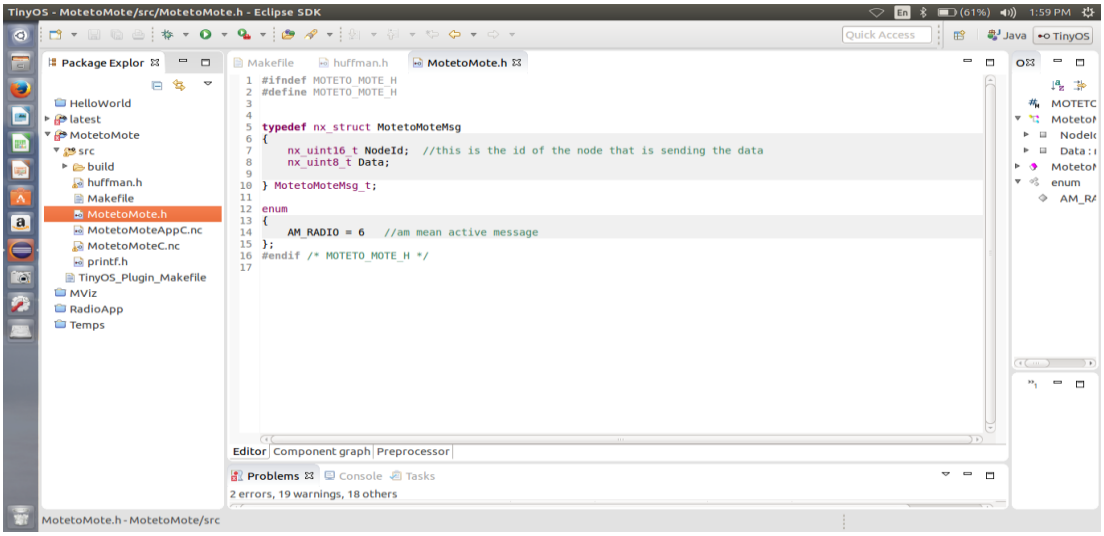


Fig3: snipped source code of the packet structure

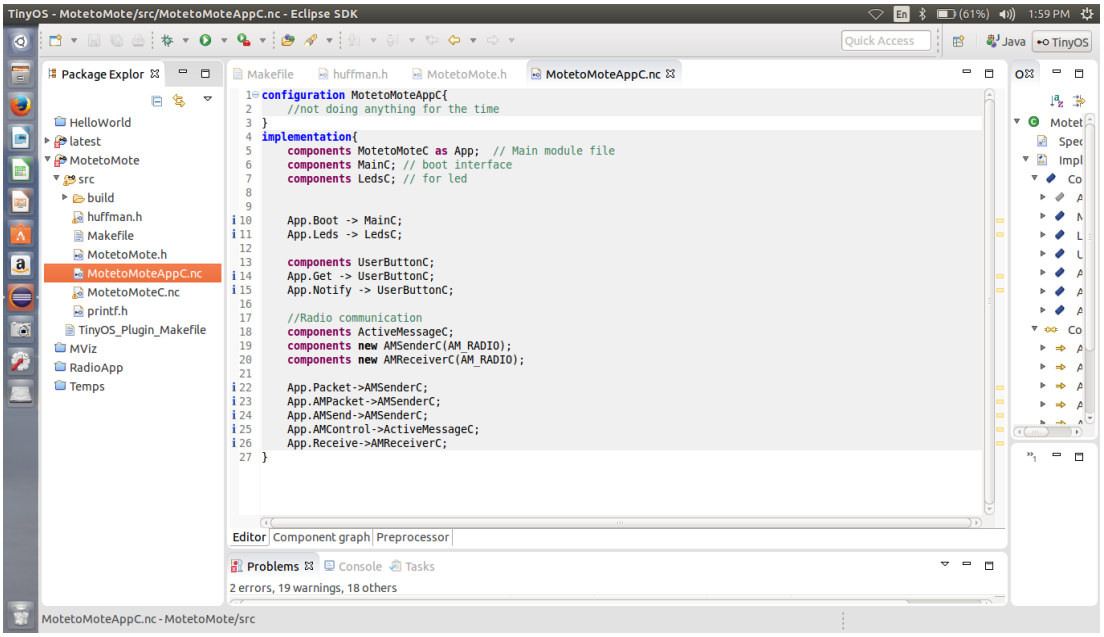
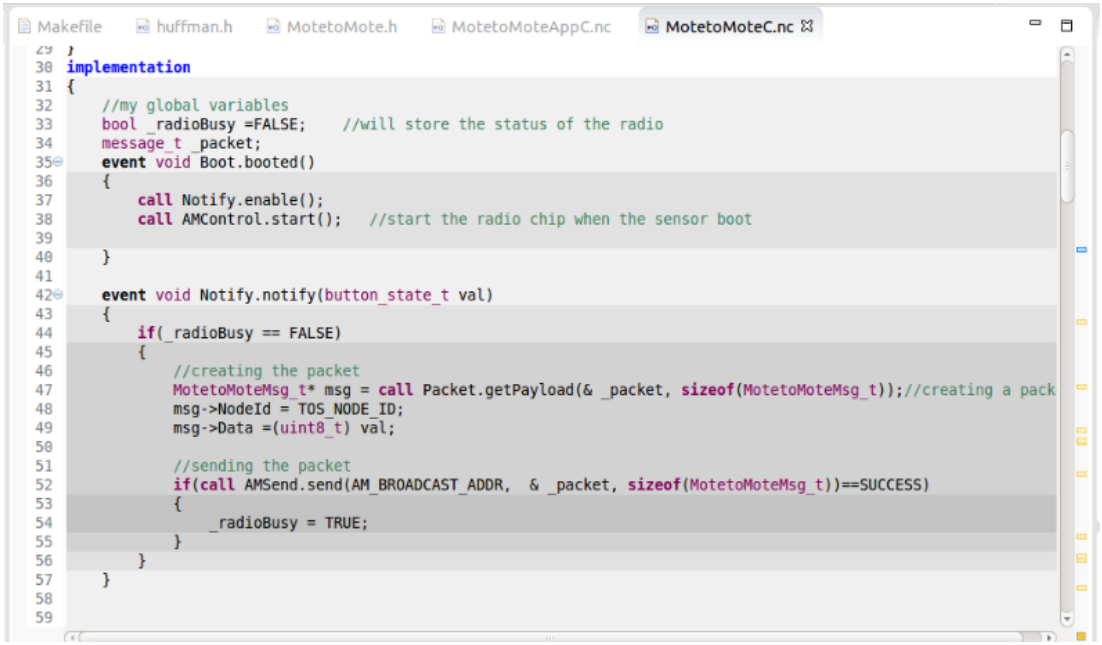


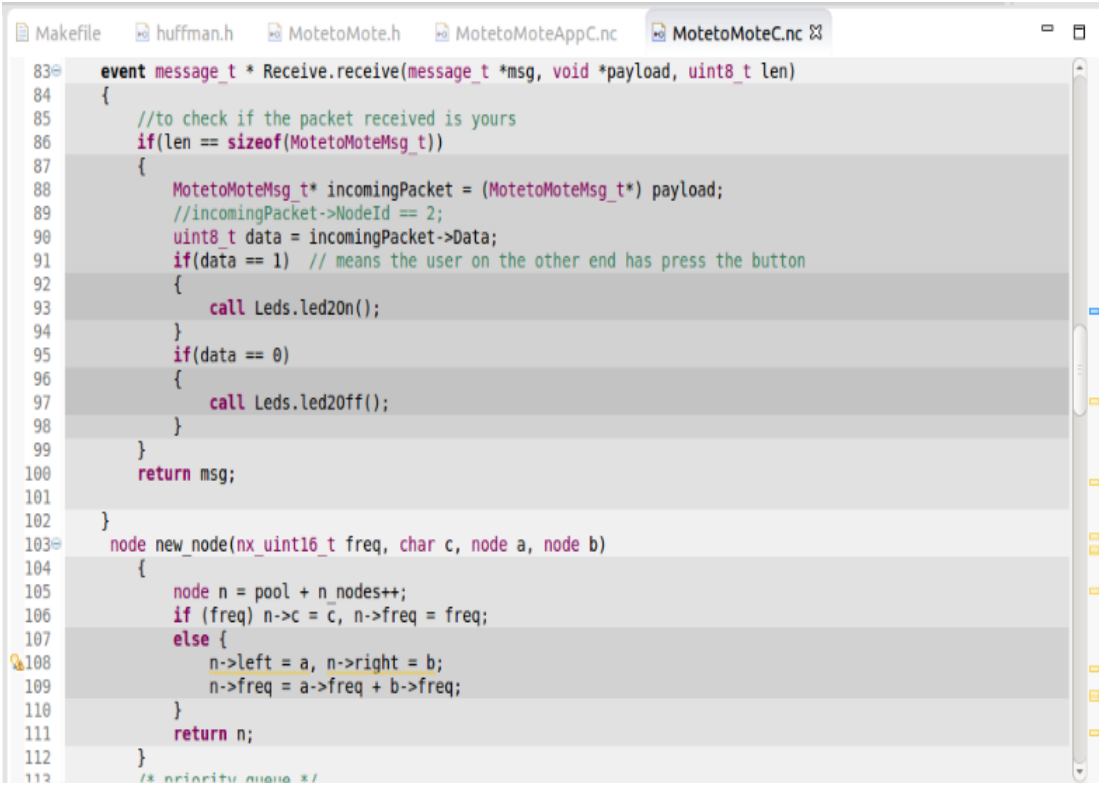
Fig3: snipped source code of the configuration file


```
#include<UserButton.h>
#include "MotetoMote.h"
#include "huffman.h"
#include "printf.h"
module MotetoMoteC
{

    uses   //general interfaces
    {
         interface Boot;
         interface Leds;
    }
    uses //UserButton related interfaces
    {
         interface Get<button_state_t>;
         interface Notify<button_state_t>;

    }
    uses   //Radio
    {
        interface Packet;  //interface that allows us to work with packets
        interface AMPacket;
        interface AMSend;   //allows you to send active message
        interface SplitControl as AMControl;   //allows you to do some basic extraction on data
        interface Receive;  // allows us to receive either in the serial port or in the radio

    }

}
```




```
}
implementation
{
    //my global variables
    bool _radioBusy =FALSE;    //will store the status of the radio
    message_t _packet;
    event void Boot.booted()
    {
        call Notify.enable();
        call AMControl.start();   //start the radio chip when the sensor boot

    }

    event void Notify.notify(button_state_t val)
    {
        if(_radioBusy == FALSE)
        {
            //creating the packet
            MotetoMoteMsg_t* msg = call Packet.getPayload(& _packet, sizeof(MotetoMoteMsg_t));//creating a pack
            msg->NodeId = TOS_NODE_ID;
            msg->Data =(uint8_t) val;

            //sending the packet
            if(call AMSend.send(AM_BROADCAST_ADDR,  & _packet, sizeof(MotetoMoteMsg_t))==SUCCESS)
            {
                _radioBusy = TRUE;
            }
        }
    }


```


```
60    event void AMSend.sendDone(message_t *msg, error_t error){
61        if(msg == & _packet)
62        {
63            _radioBusy =FALSE;
64        }
65    }
66
67    event void AMControl.startDone(error_t error){
68        if(error == SUCCESS)
69        {
70            call Leds.led0On();
71        }
72        else
73        {
74            call AMControl.start();   //will start this again
75
76        }
77    }
78
79    event void AMControl.stopDone(error_t error){
80        // TODO Auto-generated method stub
81    }
```




```
83    event message_t * Receive.receive(message_t *msg, void *payload, uint8_t len)
84    {
85        //to check if the packet received is yours
86        if(len == sizeof(MotetoMoteMsg_t))
87        {
88            MotetoMoteMsg_t* incomingPacket = (MotetoMoteMsg_t*) payload;
89            //incomingPacket->NodeId == 2;
90            uint8_t data = incomingPacket->Data;
91            if(data == 1)  // means the user on the other end has press the button
92            {
93                call Leds.led2On();
94            }
95            if(data == 0)
96            {
97                call Leds.led2Off();
98            }
99        }
100       return msg;
101
102   }
103    node new_node(nx_uint16_t freq, char c, node a, node b)
104       {
105           node n = pool + n_nodes++;
106           if (freq) n->c = c, n->freq = freq;
107           else {
108               n->left = a, n->right = b;
109               n->freq = a->freq + b->freq;
110           }
111           return n;
112       }
113       /* priority queue */
```

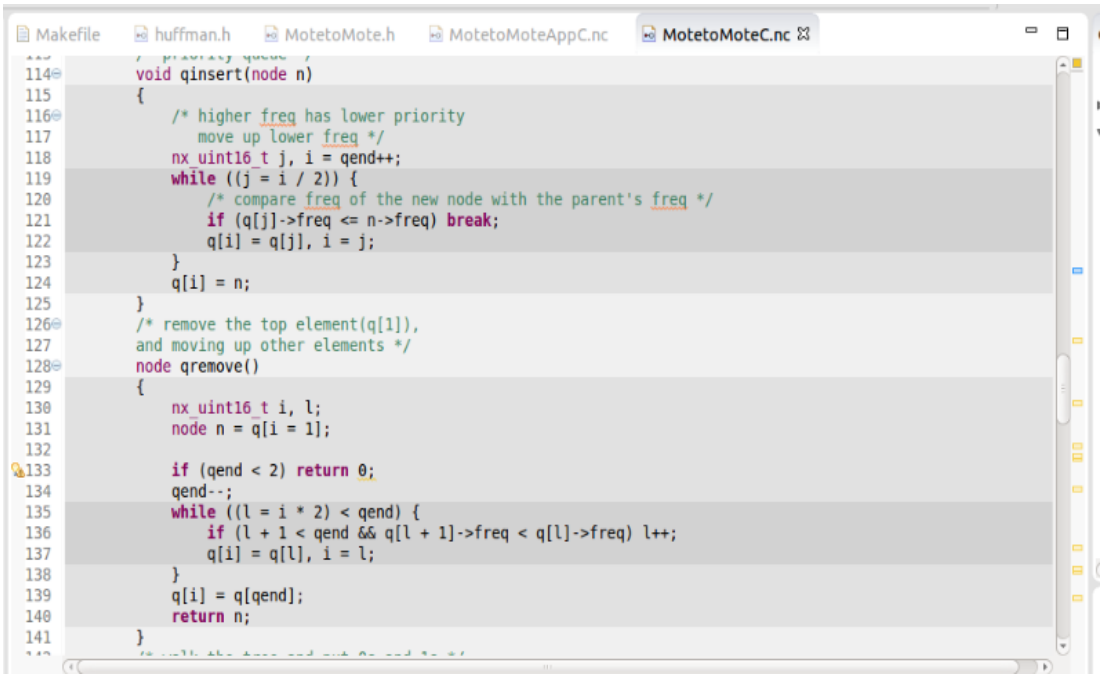


Fig4: snipped code of the module file

References

[1] Kim, Sukun, et al. "Health monitoring of civil infrastructures using wireless sensor networks." *Information processing in sensor networks, 2007. IPSN 2007. 6th international symposium on*. IEEE, 2007. [2]

[2] Xu, Ning, et al. "A wireless sensor network for structural monitoring."*Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 2004.

[3] Hsu, Chia-Hao, et al. "An implementation of light-weight compression algorithm for wireless sensor network technology in structure health monitoring." *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014.

[4] Jiang, Xiang-dong, Yu-liang Tang, and Ying Lei. "Wireless sensor networks in Structural Health Monitoring based on ZigBee technology." *Anti-counterfeiting, Security, and Identification in Communication, 2009. ASID 2009. 3rd International Conference on*. IEEE, 2009.

[5] Shahbahrami, Asadollah, et al. "Evaluation of Huffman and arithmetic algorithms for multimedia compression standards." *arXiv preprint arXiv: 1109.0216* (2011).

[6] Langdon Jr, Glen G. "An introduction to arithmetic coding." *IBM Journal of Research and Development* 28.2 (1984): 135-149.

[7] Bormann, Carsten. "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)." (2014).

[8] Kodituwakku, S. R., and U. S. Amarasinghe. "Comparison of lossless data compression algorithms for text data." *Indian journal of computer science and engineering* 1.4 (2010): 416-425.

[9] ELSON, J., AND ESTRIN, D. Time synchronization for wireless sensor networks. In Proceedings of the 15th International Parallel & Distributed Processing Symposium (2001), IEEE Computer Society, p. 186

[10] HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. System architecture directions for networked sensors. SIGPLAN Not. 35, 11 (2000), 93–104

[11] INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (Boston, MA, August 2000), ACM Press, pp. 56–67.

[12] LYNCH, J. P., SUNDARARAJAN, A., LAW, K. H., KIREMIDJIAN, A. S., AND CARRYER, E. Power-efficient data management for a wireless structural monitoring system. In Proceedings of the 4th International Workshop on Structural Health Monitoring (Stanford, CA, September 15-17 2003), vol. 1.

[13] STANN, F., AND HEIDEMANN, J. Rmst: Reliable data transport in sensor networks. In Proceedings of the First International Workshop on Sensor Net Protocols and Applications (Anchorage, Alaska, USA, April 2003), IEEE, pp. 102–112.

[14] WAN, C.-Y., CAMPBELL, A. T., AND KRISHNAMURTHY, L. Psfq: A reliable transport protocol for wireless sensor networks. In Proceeding of First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2002) (Atlanta, September 2002), pp. 1–11.

[15] Tang Yu-liang, Luo Yu, Huang Lian-fen, Guo Jian, Lei Ying, "Wireless sensor network for on-line structural health monitoring", Computer Science & Education (ICCSE) 2012 7th International Conference on, pp. 386-389, 2012.

[16] Han Zhi-gang, Cui Cai-hui, "The Application of Zigbee Based Wireless Sensor Network and GIS in the Air Pollution Monitoring", Environmental Science and Information Application Technology 2009. ESIAT 2009. International Conference on, vol. 2, pp. 546-549, 2009.

[17] Domenico Balsamo, Giacomo Paci, Luca Benini, Brunelli Davide, "Long term low cost passive environmental monitoring of heritage buildings for energy efficiency retrofitting", Environmental Energy and Structural Monitoring Systems (EESMS) 2013 IEEE Workshop on, pp. 1-6, 2013.

[18] Razzaque, Mohammad Abdur, Chris Bleakley, and Simon Dobson. "Compression in wireless sensor networks: A survey and comparative evaluation." *ACM Transactions on Sensor Networks (TOSN)* 10.1 (2013): 5.