Modified RLE with Timestamp Storage for Slow Gradient Variable Data

Chandan Maity
Embedded Systems Group
Centre for Development of Advanced Computing (C-DAC)
INDIA

Ashutosh Gupta
Embedded Systems Group
Centre for Development of Advanced Computing (C-DAC)
INDIA

Sanjat Kumar Panigrahi
Embedded Systems Group
Centre for Development of Advanced Computing (C-DAC)
INDIA

ABSTRACT

The ultra-low power embedded devices are generally battery-operated and have limited data memory. The efficient use of memory is very important as the data memory of the device is limited. Run-length encoding has attracted much attention as an important data compression technique because of its simplicity and potentially low complexity. The present paper describes a Modified Run-Length Encoding (MRLE) which has the efficient use of memory, which in turn results in minimizing energy consumption of scarce battery of embedded device and to extend the life as long as possible by maximizing the time of sleep mode. The proposed encoding method can be used in critical low power wireless sensor networks.

General Terms

Data Compression, Embedded Systems

Keywords

Run Length Encoding; Data Compression; Timestamp storage method

1. INTRODUCTION

Memory is one of the most precious components in existing electronic devices. Memory stores every data related to the concerned electronic device and therefore Memory management is one the most important tasks, especially in low size non-volatile memories such as the ones used in existing embedded devices. It is crucial for such low size memories need to store all data and commands. Over the years, processor speeds have increased at a faster rate than that of memory. Consequently, the gap between the speed of the processor and memory has widened and memory access latencies are of increasing concern. User always expects the data processing to work fast and provide results quicker in real time. Where, on one hand, this is possible by increasing memory of the embedded system, on the other hand, beyond a certain point, it is not practical to increase the memory size as with increasing size, size of the electronic device, its power consumption capacity, and the cost also increases.

One approach to more effective use of memory resources involves utilizing compression. For example, lossless data compression techniques have been used to utilize main memory resources more effectively. To be effective, however, these methods operate on relatively large data chunks stored in large size non-volatile memories and do not cater to data stored in low size non-volatile memories. For instance, taking an example of a temperature sensor that reads temperature data into an external memory device such as an EEPROM. Usually, EEPROM's have a memory capacity of 1 Mega bit or 128 kilo bytes. In most cases, 128 kilo byte of memory is not sufficient to store real time data at regular intervals of time, as a result of which EEPROM's memory exhausts after all 128 kilo byte of memory location is stored with the data. This situation can be analysed in two cases, worst case and best case. In a worst case scenario, a temperature sensor reads 1 byte of temperature data at a regular interval of 1 second. As memory capacity of EEPROM is 128 kilo byte, EEPROM gets exhausted after 128000 seconds or 128000/3600 =35.5 hours. Thus, EEPROM gets exhausted after every 35.5 hours, if 1 byte of temperature data is provided at the regular interval of 1 second. In a best case scenario, temperature sensor reads 1 byte of temperature data at a regular interval of 1 hour. As memory of EEPROM is 128 kilo byte, EEPROM gets exhausted after 128000/60 =213.33 hours or approximately 89 days.

Thus, in both the scenarios, there is a need for a system and method that allows compression of data being stored and more efficient storage of the same so that less memory is required to process tasks assigned by a user. There is also a need for a system, which processes user tasks without increasing memory, thereby not increasing size, power consumption capacity and cost of the embedded system. There is further a need for a system that performs data compression in low size non-volatile memory in real time thereby increasing memory capacity without making any changes to existing system.

The rest of the paper is organized as follows: Section II outlines related work in the area of run length encoding. Section III details the proposed solution. Section IV details the conclusion and future work.

2. RELATED WORK

Run-length encoding is a widely used simple form of data compression in which runs of data are stored as a single data value and count instead of the original run. This is most useful on data which is repeatable in nature. The major examples of varies from graphic images such as icons or animations to a data logger which is generally used for measuring and logger the environmental sensitive parameters such as temperature or humidity.

The RLE algorithm performs a lossless compression of input data based on sequences of identical values. It is a historical technique, originally used in fax machine and later adopted in image processing. The algorithm is very easy, instead of each run represented explicitly, the encoding algorithm translates the data in a pair (l, x) where l is the length of the run and x is the value of the run elements. The longer the run in the sequence to be compressed, the better is the compression ratio. [1]

3. MODIFIED RUN LENGTH ENCODING

The modified run length encoding is explained with respect to a system and a method that reads real time temperature data

and stores the data along with timestamp information in a low size non-volatile memory at a regular interval of time or on interrupt. The system comprises a temperature sensor, a microcontroller, and a real time clock (RTC). The temperature sensor monitors the temperature, whereas the microcontroller receives temperature monitored from the sensor and saves the received data in buffer defined in the data memory of the microcontroller. The real time clock (RTC) is a synchronized real time clock, which is present in the microcontroller or external to microcontroller. The real time clock (RTC) is synchronized with a reference time stamp Tref such as EPOCH time or any desired time.

In operation, temperature sensor of the proposed system reads temperature continuously. The proposed system also defines a customizable upper and lower threshold value for difference in temperature being sensed with respect to the previously sensed temperature. In an implementation and for illustration purposes, the upper threshold value and the lower threshold value of the temperature sensor is set to +/- 1, wherein in case the difference in two successive temperature readings is more than 1 or less than -1, the temperature measured later is said to be beyond defined threshold. The upper threshold value and lower threshold value of the temperature sensor can therefore be set to different values based on user needs. In an implementation, the temperature sensor generates an alarm signal through an interrupt to the microcontroller if the temperature sensor senses temperature difference between two consecutive temperature readings to be either above the upper threshold value or below the lower threshold value. Further, the temperature sensor provides temperature data to the microcontroller in real time. The microcontroller performs calculations related to time and temperature and performs data compression at a regular interval of time such that time difference and temperature difference is represented in bits in a memory efficient manner

The microcontroller remains in idle mode or sleep mode and only wakes up or comes in active mode when the microcontroller receives any interrupt signal or wakes up after a defined time interval. The microcontroller executes a method that compresses data upon reception of interrupt signal from the temperature sensor. The real time clock (RTC) is a synchronized clock present in the microcontroller or as external component to microcontroller. All the time values taken for data reading are with respect to the real time clock (RTC) and it runs 24 hours and 7 days. The device real time clock (RTC) is synchronized with a reference time stamp, wherein the reference time stamp is a base time stamp, based on which all other time stamps and difference between the time stamps are calculated. The reference time stamp is set to EPOCH time, wherein EPOCH time is a reference point for time measurement in time measurement devices. However the reference time can also be reconfigured. The current standard of EPOCH is J2000.0, which is exactly noon time of January 1, 2000, according to Gregorian calendar. Real time clock (RTC) is set to a reference time stamp Tref, wherein the reference time Tref is set to EPOCH time or configured time. In working, the microcontroller reads current time and stores it in a variable called current time Tp. The microcontroller then calculates an initial time Tinitial by calculating the difference between current time Tp and reference time stamp Tref and stores the initial time Tinitial through minimum number of bits needed to represent the difference. For instance, in case the time difference is less than 60 seconds, 6 bits are enough to represent the time difference. The microcontroller further receives an initial temperature data from the temperature sensor and stores the received data in a variable Dinitial and stores the initial temperature through minimum number of bits needed to represent the temperature. In an embodiment, instead of an interrupt being generated by the temperature sensor, the microcontroller can also continuously or periodically poll the temperature sensor and read the temperature value read thereby, and then compute the difference with the previously stored temperature reading.

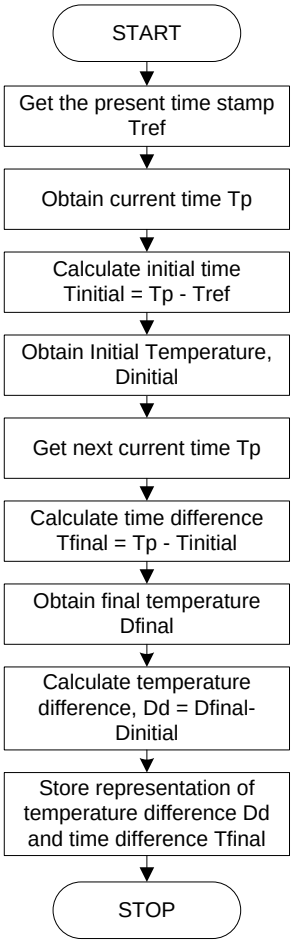

Fig 1: Algorithm flow

The microcontroller goes into sleep mode after storing the initial time Tinitial and the initial temperature Dinitial. The microcontroller remains in sleep mode and wakes up from the sleep mode when it gets an interrupt signal from the temperature sensor or at fixed time interval for calculating next time value and temperature value. The microcontroller then switches from sleep mode to wake up mode. As the process is continuous, the initial time Tinitial, stored in the memory, becomes the previous time and the next time becomes the current time Tp. The microcontroller reads the current time value Tp and calculates final time as a difference between the current time Tp and the initial time Tinitial i.e. Tfinal = Tp - Tinitial. The final time difference Tfinal is then stored in memory of the microcontroller depending on the value of the time difference Tfinal. For instance, in case time difference Tfinal is less than or equal to 60 seconds, six bits of a byte can represent the time difference Tfinal. Similarly, in case time difference Tfinal is more than 60 seconds but less than/equal to 5 minutes, 14 bits of two byte are needed represent the time difference Tfinal.

During the step of calculating the time difference, the microcontroller also calculates difference between the new temperature, also referred to final temperature Dfinal, and the initial temperature Dinitial. The difference is illustrated as Dd = Dfinal – Dinitial. In an implementation, this temperature difference can be stored in bit sequence in the memory of the microcontroller as a representation of the extent of difference. For illustration, the representation can depict, whether there is no temperature difference, whether temperature difference is more than 0 but below upper threshold, whether temperature difference is less than 0 but above lower threshold, or whether the temperature difference is above upper threshold/below lower threshold.

Four exemplary cases would now be described to illustrate the manner in which the bit sequence stores the time difference and the temperature difference representation

3.1 Time difference Representation

3.1.1 Case I: if the time difference Tfinal is less than one minute or 60 seconds.

If the time difference Tfinal is less than 60 seconds, 1 byte length can be allocated, out of which 6 bits can be allocated for storing the time difference. The time difference can be stored in Least Significant Bits (LSB) of a byte allocated for the data to be stored in the memory and remaining 2 bits can be allocated for representation of temperature difference. Memory storage format for the time difference is shown as:

XXYYYYYY

Wherein X represents temperature difference and Y represents the time difference Tfinal. For instance, the representation of temperature difference can be 00, 01, 10, and 11 depending on whether the measured temperature difference is equal to 0 (00), more than 0 and less than equal to 1 (10), less than 0 and more than equal to -1 (01), or is beyond the defined upper and lower thresholds (11). Further, in case the time difference is 50 seconds, the same can be represented as 100110 as the 6-bit sequence.

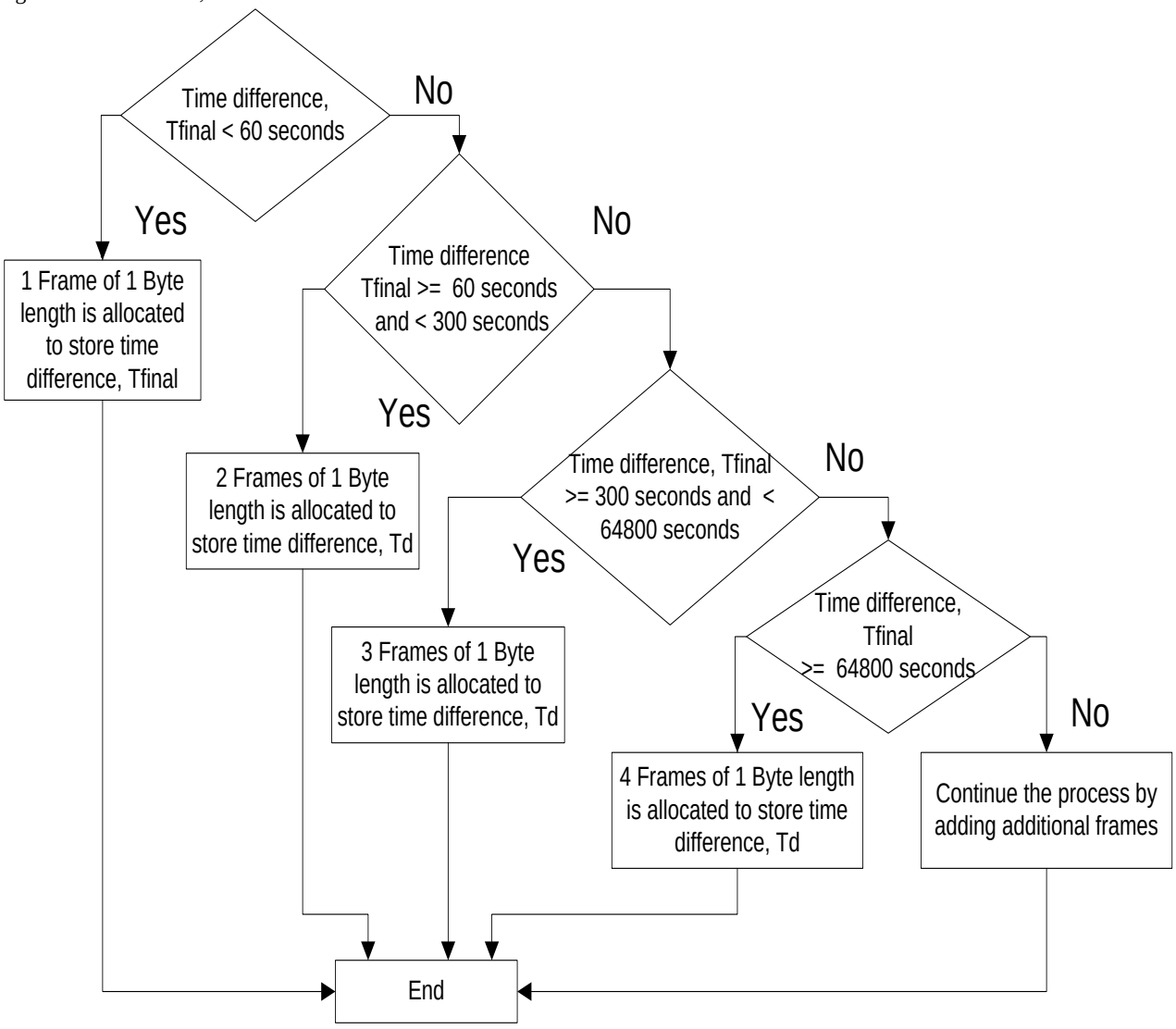

Fig 2: Time difference representation flow

3.1.2 Case II: if the time difference Tfinal is greater than or equal to 1 minute or 60 seconds and less than 5 minutes or 300 seconds.

In this case the first frame can store 60 seconds of time difference. Thus, a second frame of 1 byte length can be allocated for storing remaining time difference, td of 240 seconds. Therefore, 6 bits are allocated from frame 1 to store 60 seconds of time difference, Tfinal and 8 bits are allocated from frame 2 to store 240 seconds of time difference. Hence, upto 5 minutes or 300 seconds of time difference can be stored in 2 frames. Frame format for storing 5 minutes of time difference data is shown as:

XXYYYYYY YYYYYYYY

wherein X represents temperature difference and Y represents the time difference, Tfinal.

3.1.3 Case III: if the time difference Tfinal is greater than or equal to 5 minutes or 300 seconds and less than 18 hours or 64800 seconds.

First frame can store up to 60 seconds of time difference and second frame can store up to 240 seconds of time difference Tfinal. Thus first and seconds frame can store only up to 5 minutes or 300 seconds of time difference Tfinal. For storing remaining 64500 seconds of time difference Tfinal, a third frame of 1 byte can be allocated. Hence, first frame and second frame can store up to 5 minutes or 300 seconds of time difference Tfinal and third frame can store 64500 seconds of time difference Tfinal. Frame format for storing time difference Tfinal greater than or equal to 5 minutes and less than 18 hours or 64800 seconds is shown as:

XXYYYYYY YYYYYYYY YYYY

wherein X represents temperature difference and Y represents the time difference Tfinal.

3.1.4 Case IV: if the time difference Tfinal is greater than or equal to 18 hours or 64800 seconds.

First frame, second frame and third frame can store up to 18 hours or 64800 seconds of time difference Tfinal. For storing time difference Tfinal of more than 18 hours, another frame is allocated. Fourth frame can store time difference Tfinal of more than 18 hours or 64800 seconds. Frame format for storing time difference Tfinal for more than 18 hours or 64800 seconds is shown as:

XXYYYYYY YYYYYYYY YYYYYYYY YYYYYYYY

wherein X represents temperature difference and Y represents the time difference, Td.

3.2 Temperature difference Representation

The microcontroller can calculate temperature difference value Dd and store the temperature difference in memory in a representation that illustrates the change in temperature between initial and final readings. In an embodiment, the microcontroller checks whether the temperature difference Dd has reached an upper threshold value and lower threshold value, i.e., +/-1 degree. In case the upper and lower thresholds have not been reached, the temperature difference can be represented through two bits only, which is 00 for no change, 01 for change less than 0 but more than -1 i.e. a negative change, and 10 for change less than 1 but more than 0 i.e. a positive change. However, in case the change is beyond defined thresholds bit representation 11 can be used and 1 additional byte can be allocated to represent the final temperature. Based on the change in temperature observed by the microcontroller, four cases therefore arise for temperature difference, Dd.

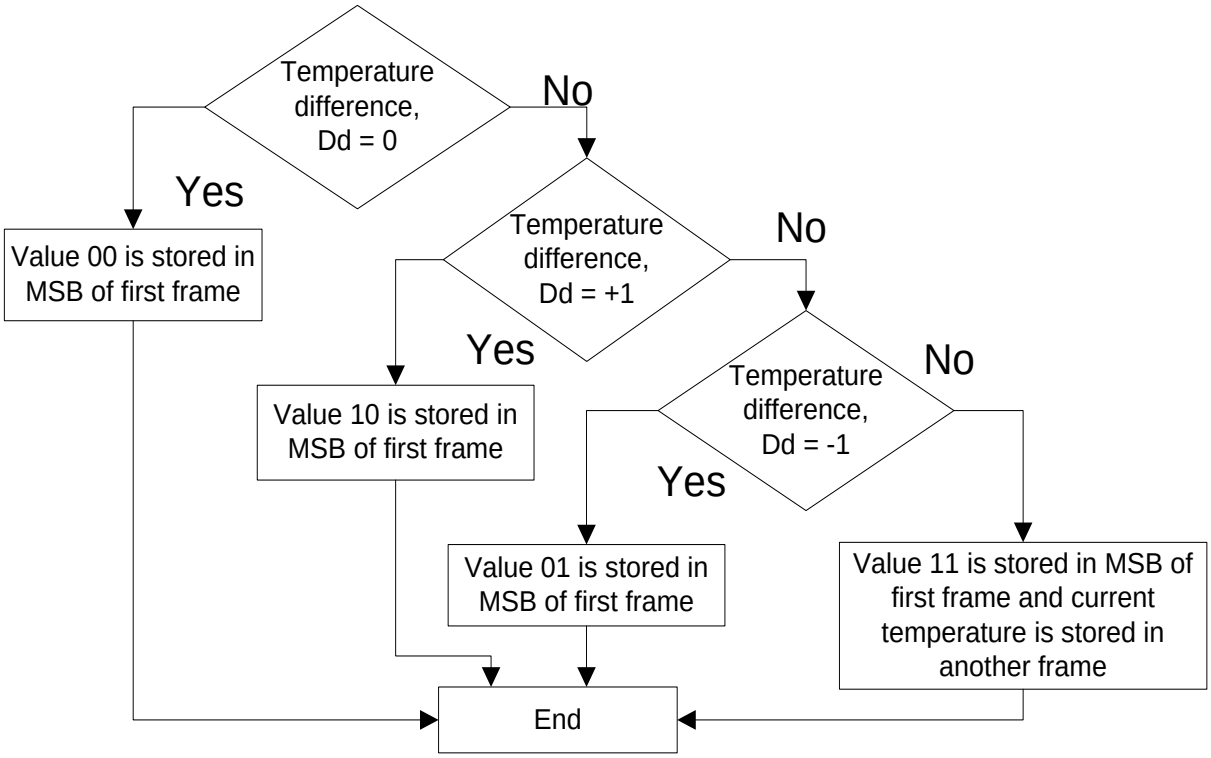

Fig 3: Temperature difference representation flow

3.2.1 Case I: if the microcontroller does not observe any temperature difference.

Dd, value 00 is stored in 2 bits allocated in the memory. In an implementation, the 2 bits can be the Most Significant Bits (MSB) of the frame allocated by the microcontroller and the allocation of memory can be shown as:

00YYYYYY

wherein 00 represents no change in the temperature difference Dd calculated by the microcontroller and Y represents time difference.

3.2.2 Case II: if the temperature difference Dd calculated by the microcontroller is equal to -1 degree.

Value 01 is stored in the MSBs of the frame allocated in the memory for storing temperature difference, Dd. Frame allocation is shown as:

01YYYYYY

wherein 01 represents change in the temperature difference, Dd observed by the microcontroller is equal to -1 degree and Y represents time difference.

3.2.3 Case III: if the temperature difference Dd calculated by the microcontroller is +1 degree.

Value 10 is stored in the MSBs of the frame allocated in the memory for storing temperature difference, Dd. Frame allocation is shown as:

10YYYYYY

wherein 10 represents change in the temperature difference, Dd observed by the microcontroller is equal to +1 degree and Y represents time difference.

3.2.4 Case IV: if the temperature difference Dd calculated by the microcontroller is +1 degree.

In this case, value 11 is stored in the MSBs and another frame is allocated for storing current temperature. Frame allocation is shown as:

XXYYYYYY ZZZZZZZZ

wherein X represents temperature difference as 11, Y represents time difference, and Z represents current temperature read by the temperature sensor.

Four cases for storing representation of the temperature difference Dd is shown with respect to the temperature difference calculated by the microcontroller at duration less than 1 minute or 60 seconds. Thus only one frame is allocated. If calculation for temperature difference, Dd is performed after 60 seconds and within 5 minutes or 300 seconds, 2 frames are allocated. If calculation for temperature difference, Dd is performed after 5 minutes or 300 seconds and within 18 hours or 64800 seconds, 3 frames are allocated. If calculation for temperature difference, Dd is performed after 18 hours or 64800 seconds, 4 frames are allocated. For instance, if temperature difference, Dd is calculated after 4 minutes and the difference observed by the microcontroller is less than 0 and more than -1 degree, value 01 is stored in the MSBs of the first frame and 2 frames are allocated for storing the time difference. If temperature difference, Dd is calculated after 20 hours and the difference observed by the microcontroller is more than 0 but less than 1, value 10 is stored in the MSBs of the first frame and 4 frames are allocated for storing the time difference. Different combination of cases can be observed based on the calculated time difference and the representation of temperature difference. The microcontroller, thus stores only a representation that depicts the temperature difference along with time difference with respect to a reference time stamp, so that the time at which the temperature was read can also be inferred, thereby eliminating allocating memory to unwanted data and compressing the data to fixed number of frames. The system efficiently utilizes the limited memory available and can store more data.

In an embodiment, based on the time difference intended to be stored, the number of bytes can dynamically be increase and allocated. In case more amount of information is desired to be stored in a compressed mode, the time that can be stored in the 6 LSBs of the first frame can be reduced. For instance, instead of 64 seconds that can be stored in the 6 bits, 50 seconds can be configured to be stored such that for subsequent data one or more new frames can be added. This allows more data to be stored in the compressed space and also allows dynamically allocating the number of bytes to store as much time difference as desired

The microcontroller utilizes less temperature data and less memory of EEPROM to store the calculated data for further use. As the process is iterative, previously calculated data is stored in RAM during calculation and moved to the EEPROM memory and stored after calculation. Thus, the memory allocated for storing temperature data can be compressed in real time, thereby increasing the efficiency of the system.

4. RESULTS

The performance of the compression algorithm is calculated by using compression ratio, which is defined as

$$Comp\_ratio = 100. (1- Comp\_size / Orig\_size)$$

Where comp_size and orig_size are the size of the compressed and uncompressed bit stream.

Table 1. Comparison Result

T_{final} (in seconds)	Conventional Method	Modified RLE	Compression Ratio
$0 < T_{final} < 60$	16 bits	8 bits	50 %
$60 <= T_{final} < 300$	24 bits	16 bits	33.3 %
$300 < T_{final} <= 64800$	24 bits	20 bits	16.67 %
$64800 < T_{final}$	40 bits	32 bits	20 %

The results show the proposed modified run length encoding is lossless compression algorithm and has on average 30 % better compression ratio as compared to the conventional method.

5. CONCLUSION

This research paper provides a new and better performance technique for data compression which can be used for slow gradient variable Data with timestamp storage. The proposed encoding technique has shown good results because temperature sensing data values slowly change in the environment. The authors have tested and used the optimized algorithm in the e-safeT data logger developed by C-DAC. In

the next few months, we hope to gain significant experience with the algorithm overall accuracy and performance, by deploying the e-safeT data logger in cold supply chain.

6. ACKNOWLEDGMENTS

This work was done as a part of project titled "Design and Development of Object Tracking system for environmental sensitive object in transit" funded by Department of Electronics and Information Technology (DeitY), Ministry of Communications and Information Technology, Government of India. Authors are thankful to Dr. G.V Ramaraju, (Scientist G & Group Coordinator) and Smt. Geeta Kathpalia (Scientist F & Head of Division) for the support. The authors are indebted to Dr. B K Murthy, Executive Director, C-DAC & Shri V. B. Taneja Director (A & R) to give enough space and freedom to cultivate and nurture the research areas in embedded systems.

7. REFERENCES

[1] Stefano Ferilli, "Automatic Digital Document Processing and Management: Problems, Algorithms and techniques," ISBN: 0857291971

[2] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression*. New York: Van Norstrand Reinhold, 1992.

[3] Z. Xiong, O. Guleryuz, and M. T. Orchard, ―A DCT-based embedded image coder,‖ *IEEE Signal Processing Lett.*, vol. 3, pp. 289–290, Nov. 1996.

[4] A. Said and W. A. Pearlman, ―New, fast, and efficient image codec based on set partitioning in hierarchical trees,‖ *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–249, June 1996.

[5] C. Tu and T. D. Tran, ―Context based entropy coding of block transform coefficients for image compression,‖ in *Proc. SPIE Applications of Digital Image Processing XXIV*, San Diego, CA, Aug. 2001, pp. 377–389.

[6] T. D.Tran and T. Q. Nguyen, ―A progressive transmission image coder using linear phase uniform filterbanks as block transforms,‖ *IEEE Trans. Image Processing*, vol. 8, pp. 1493–1507, Nov. 1999. JPEG-2000 VM3,1A software,‖ ISO, ISO/IEC JTC1/SC29/WG1 N1142, Jan. 1999.

[7] J. Liang, C. Tu, and T. D. Tran, ―Fast lapped transforms via time-domain pre- and post-filtering,‖ in *Proc. ICICS*, Singapore, Oct. 2001. H26L Test model long term number 8 (TML-8) draft0, ITU-T Study Group 16 (VCEG) June 2001.