LoRa-Based Sensor Node Energy Consumption with Data Compression

Olli Väänänen
School of Technology
JAMK University of Applied Sciences
Jyväskylä, Finland
0000-0002-7211-7668

Timo Hämäläinen
Faculty of Information Technology
University of Jyväskylä
Jyväskylä, Finland
0000-0002-4168-9102

***Abstract*—In this paper simple temporal compression algorithms' efficiency to reduce LoRa-based sensor node energy consumption has been evaluated and measured. It is known that radio transmission is the most energy consuming operation in a wireless sensor node. In this paper three lightweight compression algorithms are implemented in an embedded LoRa platform to compress sensor data in on-line mode and the overall energy consumption is measured. Energy consumption is compared to the situation without implementing any compression algorithm. The results show that a simple compression algorithm is an effective method to improve the battery powered sensor node lifetime. Despite the radio transmission's high energy consumption, the sleep current consumption is a significant factor for the device overall lifetime if the measurement interval is long.**

Keywords—Internet of Things, edge computing, sensor data, compression, energy efficiency

I. INTRODUCTION

It is known that the most significant single energy consumer of wireless sensor node is the radio transmitter. A well-known method to reduce wireless sensor node energy consumption is to compress the data and thus reduce the radio transmitting periods. In this paper, three simple temporal compression algorithms are implemented in a LoRa-based sensor node, and the overall energy consumption is measured. The effect of the compression algorithm in battery lifetime has been evaluated. In this paper it has been demonstrated that if the measurement interval is rather long, then the most effective method to lengthen the battery lifetime is to minimize the sensor node sleep current consumption between the measurement and transmission periods. Minimizing the sleep current also improves the effectivity of the powerful compression algorithm to lengthen battery lifetime. Thus, the combination of very low sleep current consumption together with light temporal compression algorithm has been demonstrated to be an effective method to lengthen the battery lifetime of the LoRa sensor node. This kind of application could be useful for example in agricultural applications for measuring some environmental magnitudes. In different smart farming applications typical measured magnitudes are for example temperature and humidity. These magnitudes behave rather linearly, and thus the measurement interval can be rather long. In agricultural applications the sensor nodes are often located in the fields, thus requiring wireless transmission and a battery powered supply. Suitable applications could be also different smart city applications. Temporal compression algorithms used in this research are very effective for linearly behaving data.

II. RELATED WORK

There are some research papers about the energy consumption of wireless sensor nodes. In [1], many myths related to energy models of wireless sensor networks have been busted. For example, it is generally known that the radio transmission (sending data packets) is the most energy consuming operation, but actually keeping the radio in idle mode listening is in total a more consuming operation. This is because the transmitting and receiving last a very short time compared to the time that the radio is listening for incoming packets. It is also demonstrated that generic energy models do not fit for all devices even if the devices use a similar technology [1]. Thus, measurements need to be made to gain reliable results for certain devices used. The energy consumption models and battery lifetime experiments of different wireless sensor node technologies are published for example in [2][3].

LoRa technology has been developed to be used in the Internet of Things applications. LoRa is a low-energy wide area network especially suitable for transmitting sensor data. There are several published research papers regarding energy consumption modelling for LoRa based sensor nodes, such as [4][5][6][7]. In [8] the energy consumption of different wireless sensor network technologies is compared. Papers have demonstrated that LoRa and SIGFOX offer the best lifetime for low intensity traffic. It is also demonstrated that the sleep power consumption is a significant factor for the device lifetime if the transmission period frequency is low [8].

In [6] the effect of LoRa parameters on the LoRa node energy consumption has been researched and measured. Higher spreading factor (SF) increases the time on air and thus increases the power consumption of transmission. But to achieve long range, high SF value needs to be used. In fact, the overall LoRa transmission energy consumption is dependent on different LoRa/LoRaWAN parameters such as spreading factor, coding rate, payload size, and bandwidth [6].

One well known method to reduce wireless sensor node energy consumption is to use data compression. Temporal lossy compression algorithms are often light and simple. If the wireless transmission periods could be reduced using data compression, it would reduce the overall energy consumption as the radio could be a longer time period on sleep mode.

For example, with LoRaWAN Class A the device is not listening to downlink messages except after uplink transmission for two short downlink windows. Thus, it can remain in sleep mode until the data needs to be transmitted [9].

Thus, there are many research papers dealing with LoRa device energy consumption and other papers dealing with compression algorithms efficiency to reduce energy consumption. But not many experiments have been published on how the simple compression algorithm can reduce LoRa-sensor node energy consumption and thus lengthen the battery lifetime. In this paper the effectiveness of simple temporal compression algorithms to reduce overall energy consumption of the LoRa sensor node is evaluated and measured.

III. Temporal Compression Methods for Sensor Data

Compression algorithms implemented and tested in this paper are simple temporal compression methods based on data linearity. Many environmental magnitudes behave rather linearly, and simple linearity-based algorithms utilize that behavior. The methods are very simple, easy to understand and easy to implement in an embedded platform. IoT wireless sensor nodes are often computationally constrained, and thus simple and light compression algorithms suit these devices well. Wireless sensor nodes are also often battery powered or even harvesting the energy from the environment. Thus, minimizing the energy consumption is important for lengthening the device lifetime.

The implemented algorithms are Lightweight Temporal Compression (LTC) and Real-Time Linear Regression based Temporal Compression (RT-LRbTC). LTC is originally presented in [10]. RT-LRbTC is a modification of other linear regression-based compression algorithms, and it is originally presented by the authors of this paper in [11]. These algorithms are lossy methods meaning that reconstructed data after compression is not exactly the same as the original data. The maximum reconstruction error is determined by the error bound (ε) used.

A. Lightweight Temporal Compression

Lightweight Temporal Compression (LTC) is a well-known temporal compression method. It was first presented in 2004 in [10], and several modifications on that algorithm have been presented since then. It has proved to be a very effective compression algorithm, especially for environmental magnitudes [12]. As a disadvantage, the LTC has unpredictable latency, and thus it is not well suited for real-time or near real-time applications [13]. In this paper the LTC is used as it is in its original version and transformed and implemented for embedded Arduino board from MATLAB version used in [11] and [12]. The LTC algorithm itself is computationally very light. With every new measured value, comparisons need to be made between the new value with error bound extremes to the previously calculated upper and lower limit lines. As a result of the comparisons in maximum two new lines are calculated. Calculating line parameters (slope and y-intercept) requires one division, one multiplication, one summation and subtractions. Thus, the compression algorithm is computationally very light.

B. Real-Time Linear Regression based Temporal Compression

Real-Time Linear Regression based Temporal Compression (RT-LRbTC) is also a very simple temporal compression algorithm. It is a modification from other linear regression-based algorithms, and it is developed especially for compressing the on-line sensor data stream. The algorithm's suitability for compressing on-line data stream is based on minimizing the algorithm's inherent latency. In general, this algorithm's inherent latency is one measurement interval (Δt) long. The basic form of the algorithm uses 3 previous data values ($N = 3$) to calculate the regression line and uses it to predict future values with certain error bound. The algorithm is explained in detail in [11]. For every new measured value its value needs to be compared to the previously calculated regression line value in that time stamp. If the difference between the new value and regression line is more than error bound, then a new regression line is calculated. Calculating the regression line requires the sum of N times x_k, x_k^2, y_k, $x_k y_k$, and the square of the sum of x_k. x_k is the time stamp and y_k is the measured value. In this paper also $N = 4$ version is tested to see if calculating regression line with 4 values has any effect on the complexity and energy consumption of the compression algorithm calculation. The implemented version is derived to Arduino from MATLAB version used in [11].

IV. Energy Consumption of Implemented Compression Methods with LoRa Connection

Embedded LoRa-based sensor node used in this experiment was Arduino MKR WAN 1310 board. It has a Microchip SAM D21 32-bit Arm Cortex-M0+ based low power microcontroller and Murata CMWX1ZZABZ LoRa module. It also has a crypto chip ATECC508 by Microchip [14]. Arduino boards are widely used by hobbyists but also in research and in the industry due to their simplicity and ease of use.

DHT22 sensor was used to measure temperature. DHT22 is a low-cost temperature and humidity sensor with a digital output [15]. It is widely used by hobbyists for its low cost, and there are many project examples available on the internet where DHT22 sensor is used. There are also support libraries available for most used embedded development boards. The sensor has measurement resolution 0.1 Celsius degrees for temperature, and its accuracy is <±0.5 Celsius degrees for temperature measurement [15].

Arduino MKR WAN 1310 was powered by Lithium-Polymer (Li-Po) battery. The Li-Po battery used had 2 000 mAh capacity with 3.7 V nominal voltage. Thus, its energy capacity was 7.4 Wh which is 26 640 Ws. The Arduino board supports Li-Po battery with JST-PH connector, and it has a built-in battery charger. Arduino board can also be powered from the USB port, but the energy consumption is lower if powered from the battery.

A. Measurement Setup

The compression algorithms implemented were set to measure the temperature with DHT22 sensor at regular intervals. The embedded board was set to go to deep sleep mode between the measurements. After every measurement the algorithm compressed the data, and the compressed data was stored and sent via LoRa network if needed. The LoRa network used was a commercial network in Finland.

The board's current consumption was measured with oscilloscope current probe from the battery plus-wire, except that the deep sleep current consumption was measured with a digital multimeter (DMM), which was set in series with the battery. The other possibility would be to use a so called shunt resistor in series with the battery and measure the voltage over the resistor with oscilloscope voltage probe. The current probe was chosen to be used because of its simplicity and ease of use even though its accuracy is not very good when measuring very low-level current values.

The measurement devices used were Tektronix MSO 4104 Mixed Signal Oscilloscope with 1 GHz measurement bandwidth, and the probes used were Tektronix TCP0030 Current Probe and TAP1500 Active Voltage Probe. The current probe has 1 mA sensitivity and 1 % DC accuracy [16]. The DMM used was Tenma RS232C TrueRMS model.

B. *Deep Sleep Current Consumption*

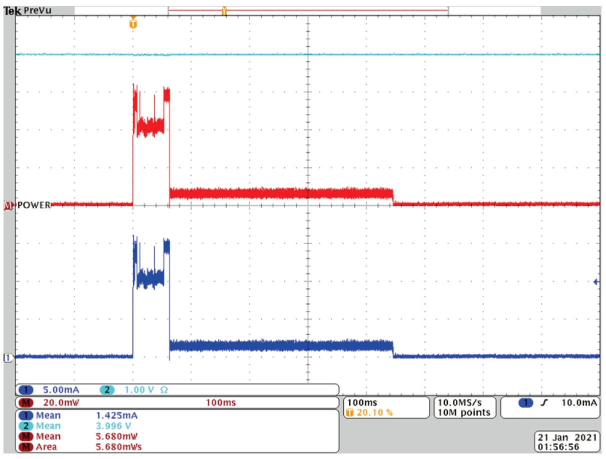
Deep Sleep current consumption was measured with the algorithms implemented on the board. The board was set to deep sleep mode between measurements. The DMM was set in series with battery wire. The measured deep sleep current was not dependent on the algorithm used, as the measurement result was the same with every algorithm tested and also without compression algorithm implemented. The measured deep sleep current was 116 µA – 117 µA. After the measurement instant and possibly LoRa-transmission, when the Arduino board went to deep sleep mode, the current dropped down immediately to 150 µA - 160 µA level, and continued going down quickly to 120 – 130 µA level and then more slowly to stabilize to 116 µA – 117 µA level. It took about 20 seconds to gain the 117 µA level. The battery voltage was measured at the same time with the oscilloscope's voltage probe connected to the board's battery connection. The battery voltage was 3.99 V while measuring the deep sleep current. Thus, the power consumption in deep sleep mode was: $P_{ds} = U_{batt} \times I_{ds} = 3.99 \text{ V} \times 117 \cdot 10^{-6} \text{ A} = 4.6683 \cdot 10^{-4} \text{ W} = 0.46683$ mW.

The 117 µA current consumption in deep sleep mode is not very low for a modern microcontroller development board, and with some other boards it could be possible to achieve lower levels.

C. *Sensor Measurement and Algorithm Current Consumption*

On a regular basis the embedded board wakes up from the deep sleep mode and makes the measurement. The real-time clock (RTC) is running even when the board is in deep sleep mode, and it wakes up the board. After the wake up, the board measures the temperature from DHT22 sensor and applies the compression algorithm implemented. As a result of algorithm calculations, the board either sends the compressed value via LoRa-connection or falls into deep sleep mode to wait for the next measurement period.

The current/energy consumption during sensor measurement and compression algorithm period was measured with oscilloscope current probe, and oscilloscope automatic measurement functions were used to show the mean values. The battery voltage was measured at the same time with voltage probe, and the oscilloscope MATH-function was used to multiply the measured values to get the power value. The oscilloscope measurement function "area" was used to calculate the power graph area to get the energy consumption. In Fig. 1. one measurement result can be seen. The deep sleep energy consumption with the same oscilloscope settings was also measured to be subtracted from the measurement period result. Thus, only sensor measurement and algorithm calculation period consumption were achieved. The sensor measurement, data compression and board shut down to deep sleep takes about 60 ms time (the high pulse in Fig. 1.). After the measurement and algorithm calculations, the current consumption did not go directly to the deep sleep level but remained slightly higher for 390 ms due to DHT22 sensor.



```
1  5.00mA      2  1.00 V Ω
M  20.0mW                100ms       100ms        10.0MS/s     1  ʃ  10.0mA
                                     T 20.10 %    10M points
1  Mean      1.425mA
2  Mean      3.996 V
M  Mean      5.680mW                                           21 Jan  2021
M  Area      5.680mWs                                          01:56:56
```

Fig. 1. RT-LRbTC: sensor measurement and algorithm period.

The oscilloscope measurement settings were: current probe 5 mA/div, voltage probe 1 V/div and horizontal scale 100 ms/div. Each measurement was repeated ten times, and the average values of the measurements were used for better measurement reliability. The measurement results can be seen in TABLE I. Each value in TABLE I is an average value of the ten separated measurements in one second measurement window (oscilloscope screen scale 100ms/div = total 1 second, as seen in Fig. 1.).

TABLE I. SENSOR MEASUREMENT AND ALGORITHM MEASUREMENT RESULTS

	No compression	**LTC**	**RT-LRbTC, *N*=3**	**RT-LRbTC, *N*=4**
Current (mA)	1.3977	1.4247	1.4016	1.4500
Battery voltage (V)	3.9949	3.9925	3.9906	3.9963
Power (mW)	5.5802	5.6836	5.6208	5.7814
Energy (mWs)	5.5775	5.6836	5.6208	5.7814
Deep sleep energy (mWs)	1.0003	0.9013	0.8428	0.9492
Measurement + algorithm (mWs)	4.5772	4.7823	4.7780	4.8322

The last line of TABLE I shows the results for the energy consumed for sensor measurement and algorithm calculations for each algorithm and also without any compression algorithm for comparison. The result is achieved by subtracting the deep sleep energy value (second last row) from energy value (third last row). It can be seen from the results that the values are on the same level for each algorithm and even without compression algorithm implemented, this means that the algorithm calculations do not affect at all the board's energy consumption, or the effect is so small that it is not possible to recognize it with this measurement setup. As can be seen from the results, there is no significant difference between the algorithms. The results between the algorithms and without an algorithm are within measurement uncertainty. The measured differences are negligible.

D. LoRa Transmission Energy Consumption

LoRa node used the spreading factor SF10 for transmitting, and the network sent the confirmation with SF9 or SF12. During the whole testing period the network confirmation was sent with SF9 for 49.2 % of all occasions and with SF12 for 50.8 % for all occasions. TABLE II shows the energy consumption measurement results when LoRa node is sending the data packets with SF10, and the network sends the confirmation with SF9. In TABLE III are the results when the data up is with SF10 and data down with SF12. Every measurement is repeated ten times, and the results are the average values of the measurements. In Fig. 2. one result for the SF10 uplink and SF9 downlink situation can be seen. It seems that the LoRa sensor node is not receiving the confirmation from the network even though the network has sent the confirmation. Two 10 mA pulses, the first approximately one second after the transmit, and the second one two seconds after transmit are the two receive windows that follow the uplink. Because the downlink is not received during the first window, then the second receive window opens two seconds after the transmit. In Fig. 3. there is the result of SF10 uplink, SF12 downlink situation. In this case it seems that the LoRa node has received the downlink confirmation because the receive window is rather long and not followed by the second receive window.

In TABLE II and TABLE III, the last row transmission energy is calculated by subtracting the deep sleep energy and previously measured sensor measurement and algorithm energy consumptions (TABLE I) from the measured transmission energy value (third last row in TABLE II and III). The deep sleep energy was measured with the same setup and measurement equipment settings while the LoRa-sensor node was in deep sleep mode. In Fig. 2. and Fig 3. the current measurement with the current probe is the blue line (probe 1), battery voltage is the light blue line (probe 2), and power is the red M-line which is voltage multiplied with current. The results in Fig. 3. include the sensor measurement and algorithm calculations, but Fig. 2. includes only the sensor measurement as the compression algorithm was not implemented in this case. The transmission period is approximately 400 ms long, and it is followed by the network confirmation time window one second after the transmission.

As can be seen in the transmission energy values (last row in TABLE II and TABLE III) there is not a big difference between the algorithms and taking into account the measurement uncertainty of low current values measured with current probe, the differences are negligible. The results with no compression algorithm implemented are even higher than with LTC algorithm. The results of LTC algorithm compression are the temperature value (float number) and time stamp (which is only a sequence number, integer), thus the transmitted data is 8 bytes. With RT-LRbTC algorithms the regression line parameters (slope and base, both float numbers) with time stamp (sequence number, integer) are sent in total 12 bytes. Thus, the amount of data sent is bigger with RT-LRbTC algorithms, and thus this could explain a slightly higher energy consumption in the transmission. In general, these results can be regarded to be approximately the same for each algorithm. The differences are negligible and can be explained by measurement uncertainty.

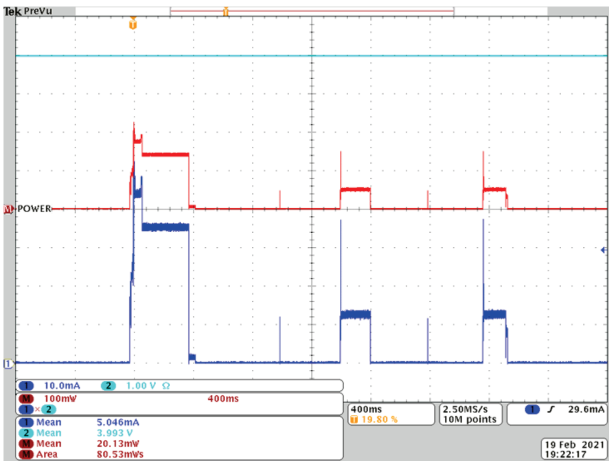
TABLE II. UPLINK SF10, DOWNLINK SF9

	No Compression	**LTC**	**RT-LRbTC, *N*=3**	**RT-LRbTC, *N*=4**
Current (mA)	4.9736	4.7976	5.1049	5.2293
Battery voltage (V)	3.9928	3.9924	3.9887	3.9897
Power (mW)	19.842	19.143	20.323	20.652
Energy (mWs)	79.372	76.565	81.289	82.605
Deep sleep energy (mWs)	3.5636	4.0993	4.971	4.632
Transmitting (mWs)	71.2312	67.6834	71.54	73.1408

TABLE III. UPLINK SF10, DOWNLINK SF12

	No Compression	**LTC**	**RT-LRbTC, *N*=3**	**RT-LRbTC, *N*=4**
Current (mA)	7.4714	7.3759	7.7266	7.598
Battery voltage (V)	3.9919	3.9918	3.9879	3.9851
Power (mW)	29.807	29.425	30.791	30.061
Energy (mWs)	119.21	117.68	123.15	120.26
Deep sleep energy (mWs)	3.5636	4.0993	4.971	4.632
Transmitting (mWs)	111.0692	108.7984	113.401	110.7958

The oscilloscope settings were 10 mA/div for current probe, 1 V/div for voltage probe, and the horizontal scale was 400 ms/div. The values were measured using oscilloscope mean value measurement function from the oscilloscope screen area which was 4 seconds in total. The settings can be seen in Fig. 2. and Fig. 3.



1 10.0mA 2 1.00 V Ω
M 100mW 400ms
1 × 2
1 Mean 5.046mA
2 Mean 3.993 V
M Mean 20.13mW
M Area 80.53mWs
400ms
19.80 %
2.50MS/s
10M points
1 29.6mA
19 Feb 2021
19:22:17

Fig. 2. Measurement and LoRa transmission period with SF10 uplink and SF9 downlink. No compression algorithm implemented

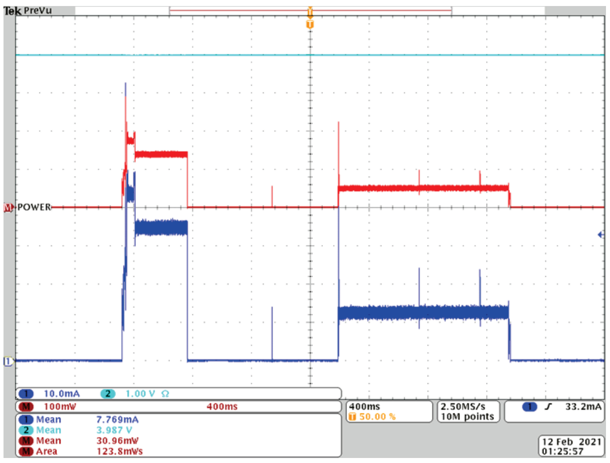


Fig. 3. RT-LRbTC (N=3). Measurement, algorithm and LoRa transmission period with SF10 uplink and SF12 downlink

V. Overall Energy Consumption and Battery Lifetime

The overall energy consumption of the LoRa sensor node with the compression algorithm implemented is combined from the measurement event and algorithm consumption (W_M), which happens on a regular basis (measurement interval Δt), the LoRa transmission event consumption (W_S) which frequency of occurrence depends on the measurement interval (Δt), and the compression ratio (CR). The overall time is t_x. Between the measurement events the device is in deep sleep mode and its power consumption is P_{ds}. The total energy consumption (W_{TOT}) can be calculated approximately by (1):

$$W_{tot} = P_{ds}t_x + \frac{t_x}{\Delta t}W_M + \frac{t_x}{CR \times \Delta t}W_s \tag{1}$$

Where $P_{ds}t_x$ is the energy that the device uses in deep sleep mode during the whole time t_x. This value includes the measurement and transmission periods as well as the deep sleep values were subtracted from the other energy values in TABLEs I-III. $t_x/\Delta t$ is the number of the measurement periods. $t_x/(CR \text{ x } \Delta t)$ is the number of transmission periods.

As the DHT22 temperature sensor has the accuracy of ±0.5 Celsius degrees, it is reasonable to use error bound value ε = 0.5 Celsius degrees for compression algorithms as an example. In [11] the LTC and RT-LRbTC (N = 3) have achieved the compression ratios CR = 9.5-10.2 (LTC) and CR = 5.5-6.0 (RT-LRbTC) for real temperature data with a 10-minute measurement interval (Δt), when the error bound used has been ±0.5 Celsius degrees. RT-LRbTC with N = 4 has not been tested with available temperature data set. The temperature data sets used in [11] were real temperature data sets achieved from Finnish Meteorological Institute's open data service. The data sets were Salla Naruska measurement station data from whole years 2018 and 2019. The temperature data used was measured and presented with 0.1 Celsius degrees resolution.

As an example, the 2 000 mAh battery lifetime can be calculated with 10-minute measurement intervals and with measurement results from TABLEs I, II and III. The equation (1) solved for overall time t_x is (2):

$$t_x = \frac{W_{TOT}}{P_{ds} + \frac{W_M}{\Delta t} + \frac{W_S}{CR \cdot \Delta t}} \tag{2}$$

With a 10-minute measurement interval, the Δt = 600 s. The 2 000 mAh Li-Po battery total energy is W_{TOT} = 7.4 Wh = 26 640 Ws. This amount of available energy was used even though it is a rather optimistic estimation. For example, if the battery powered sensor node is located outside, the temperature can be as low as -30 Celsius degrees in Finland. It is well known that the capacity of lithium-based batteries can drop significantly in cold conditions.

The battery lifetime without any compression algorithm can be estimated by using CR = 1 value. For other parameters the no compression measurement values from TABLEs I-III can be used. The values used (no compression): P_{ds} = 0.46683 mW, W_M = 4.5772 mWs, W_S = 91.4689 mWs. The W_S value is calculated from TABLE II and TABLE III values by taking into account that 49.2 % of transmitting periods were SF10, SF9 events and 50.8 % were SF10, SF12 events. The estimated battery lifetime without any compression algorithm used by equation (2) is: 42 494 353 s = 491.8 days

The battery lifetime with LTC algorithm implemented was calculated with compression ratio CR = 10. Thus, overall battery lifetime with LTC algorithm if the whole battery capacity can be used is: 54 415 973 s = 629.8 days. The battery lifetime is 28.1 % longer than without compression.

In TABLE IV are the results for all tested algorithms with their measured energy consumption values. The compression ratio used for RT-LRbTC algorithms was CR = 6.

TABLE IV. Battery Lifetime with Different Algorithms

	No Compression	**LTC**	**RT-LRbTC, N=3**	**RT-LRbTC, N=4**
CR	1	10	6	6
Battery lifetime (days)	491.8	629.8	615.9	616.0

With RT-LRbTC algorithms the battery lifetime is lengthened by 124.2 days, which is 25.2 % longer lifetime than without any compression.

The deep sleep energy consumption and measurement intervals are very significant parameters for the LoRa-node lifetime together with the compression ratio. In this example the measurement interval is rather long as is often the case in agricultural applications. Thus, the node's deep sleep energy consumption determines mostly the battery lifetime. The Arduino MKR WAN 1310 deep sleep current (measured 117 µA) is not an especially low level for a modern embedded sensor node. If the deep sleep power consumption could be reduced by 50 %, then the battery lifetime without any compression would be 783.6 days, with LTC algorithm 1203.7 days and with RT-LRbTC algorithms 1154 days. Thus, the LTC algorithm would lengthen the lifetime by 420 days, which is 53.6 % longer lifetime than without any compression. The RT-LRbTC algorithms would lengthen the lifetime by 370 days, which is 47.2 % longer than without any compression.

If a bigger reconstruction error is allowed, then the error bound can be higher than 0.5 degrees. With bigger error bound the compression ratio is better thus reducing the overall energy consumption. In [11] the authors have simulated the same compression algorithms with real temperature data with 10-minute measurement interval. For those data sets the compression ratio has been up to 20 for LTC with 1.0 degrees error bound and up to 10 for RT-LRbTC algorithm with 1.0 degrees error bound. Fig. 4. presents the 2000 mAh battery lifetime for LTC and RT-LRbTC (N=3) algorithms with different error bound values. The CR values for different error bounds are the same as in [11]. The battery lifetime lengthening is rather limited if the error bound is increased from 0.5 degrees to 1.0 degrees. The effect is only about 10 days. Thus, it should be considered if using higher error bound is worth of having a significantly bigger reconstruction error.

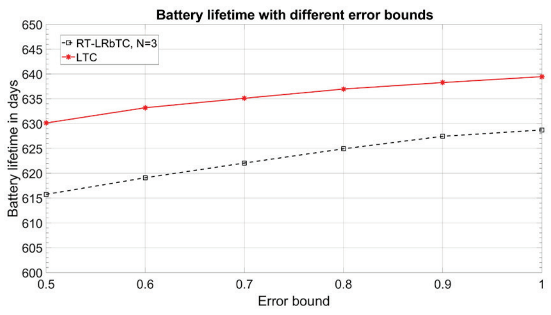


Fig. 4. The effect of error bound on battery lifetime

In general, it is not possible to affect the energy consumption that the measurement itself consumes and the LoRa-transmission consumes a lot. LoRa radio can be set to use certain spreading factor for lower energy consumption, but it can shorter the range. Thus, the most effective ways to reduce overall energy consumption in this kind of sensor node is to choose the low energy consumption embedded platform, which has very low sleep current consumption and to use an effective compression algorithm suitable for low power sensor node. If the measurement interval was shorter, then the data would behave more linearly, and the linearity-based compression algorithms would behave more effectively and result in higher compression ratio. The effect in energy saving achieved with data compression would be bigger because there would be more data transmission periods if no compression algorithm is used, and a higher compression ratio would result in higher reduction in transmission periods. It would underline the compression algorithm's effect on reducing the sensor node's overall energy consumption. In [11] the compression ratio values achieved with LTC and RT-LRbTC (N=3) for air pressure data with 10-minute measurement interval were almost CR = 30 (for LTC) and CR = 15 (for RT-LRbTC) when the error bound was 0.5 hPa. Thus, it can be estimated that the effect of these compression methods on overall energy consumption would be bigger if the air pressure was the measured magnitude.

VI. Conclusions

In this paper we implemented simple sensor data compression algorithms on LoRa-based sensor node. The overall energy consumption was measured with and without implemented compression algorithm. The measurement results demonstrated that the tested compression algorithms are computationally so light that due to calculations they do not have any effect on embedded device energy consumption. The overall saving in energy consumption is due to the reduced amount of radio transmission periods thanks to data compression. The measurement interval was ten minutes, which is rather long but can be typical in agricultural applications measuring some environmental magnitudes. Due to the long measurement interval, the device's sleep energy consumption was proved to be the most significant factor in the device's lifetime.

References

[1] D. Harrison, D. Burmester, W. Seah, and R. Rayudu, "Busting myths of energy models for wireless sensor networks," Electron. Lett., 52: 1412-1414, 2016. https://doi.org/10.1049/el.2016.1591

[2] M. Srbinovska, V. Dimcev and C. Gavrovski, "Energy consumption estimation of wireless sensor networks in greenhouse crop production," *IEEE EUROCON 2017 -17th International Conference on Smart Technologies*, Ohrid, 2017, pp. 870-875, doi: 10.1109/EUROCON.2017.8011235.

[3] J. Rahmé, N. Fourty, K. Al Agha and A. Van den Bossche, "A Recursive Battery Model for Nodes Lifetime Estimation in Wireless Sensor Networks," *2010 IEEE Wireless Communication and Networking Conference*, Sydney, NSW, Australia, 2010, pp. 1-6, doi: 10.1109/WCNC.2010.5506424.

[4] N. Madiyar, and K. Nurzhigit, "Prediction of energy consumption for LoRa based wireless sensors network," Wireless Networks, 26(5), pp. 3507-3520, 2020. doi:10.1007/s11276-020-02276-5

[5] T. Bouguera, J. Diouris, J. Chaillout and G. Andrieux, "Energy consumption modeling for communicating sensors using LoRa technology," *2018 IEEE Conference on Antenna Measurements & Applications (CAMA)*, Vasteras, 2018, pp. 1-4, doi: 10.1109/CAMA.2018.8530593.

[6] T. Bouguera, J. Diouris, J. Chaillout, R. Jaouadi, and G. Andrieux, "Energy Consumption Model for Sensor Nodes Based on LoRa and LoRaWAN," Sensors (Basel, Switzerland), 18(7), p. 2104, 2018, doi:10.3390/s18072104

[7] L. Casals, R. Vidal, and C. Gomez, "Modeling the Energy Performance of LoRaWAN," Sensors, 17(10), p. 2364. 2017, doi:10.3390/s17102364

[8] É. Morin, M. Maman, R. Guizzetti and A. Duda, "Comparison of the Device Lifetime in Wireless Networks for the Internet of Things," in *IEEE Access*, vol. 5, pp. 7097-7114, 2017, doi: 10.1109/ACCESS.2017.2688279.

[9] About LoRaWAN. https://lora-alliance.org/about-lorawan/

[10] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow and D. Estrin, "Lightweight temporal compression of microclimate datasets [wireless sensor networks]," *29th Annual IEEE International Conference on Local Computer Networks*, Tampa, FL, USA, 2004, pp. 516-524, doi: 10.1109/LCN.2004.72.

[11] O. Väänänen, and T. Hämäläinen, "Sensor Data Stream on-line Compression with Linearity-based Methods," *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, Bologna, Italy, 2020, pp. 220-225, doi: 10.1109/SMARTCOMP50058.2020.00049.

[12] O. Väänänen and T. Hämäläinen, "Compression Methods for Microclimate Data Based on Linear Approximation of Sensor Data" The 19th International Conference on Next Generation Wired/Wireless Advanced Networks and Systems NEW2AN 2019, August 26 - 28, 2019, St.Petersburg, Russia.

[13] G. Giorgi, "A Combined Approach for Real-Time Data Compression in Wireless Body Sensor Networks," in *IEEE Sensors Journal*, vol. 17, no. 18, pp. 6129-6135, 15 Sept.15, 2017, doi: 10.1109/JSEN.2017.2736249.

[14] Arduino MKR WAN 1310. https://store.arduino.cc/mkr-wan-1310

[15] DHT22 temperature and humidity sensor. https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[16] Tektronix TCP0030 Current Probe Instruction Manual. https://www.tek.com/current-probe-manual/tcp0030