Efficiency of temporal sensor data compression methods to reduce LoRa-based sensor node energy consumption

Olli Väänänen
School of Technology, JAMK University of Applied Sciences, Jyväskylä, Finland, and
Timo Hämäläinen
Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland

Abstract

Purpose – Minimizing the energy consumption in a wireless sensor node is important for lengthening the lifetime of a battery. Radio transmission is the most energy-consuming task in a wireless sensor node, and by compressing the sensor data in the online mode, it is possible to reduce the number of transmission periods. This study aims to demonstrate that temporal compression methods present an effective method for lengthening the lifetime of a battery-powered wireless sensor node.

Design/methodology/approach – In this study, the energy consumption of LoRa-based sensor node was evaluated and measured. The experiments were conducted with different LoRaWAN data rate parameters, with and without compression algorithms implemented to compress sensor data in the online mode. The effect of temporal compression algorithms on the overall energy consumption was measured.

Findings – Energy consumption was measured with different LoRaWAN spreading factors. The LoRaWAN transmission energy consumption significantly depends on the spreading factor used. The other significant factors affecting the LoRa-based sensor node energy consumption are the measurement interval and sleep mode current consumption. The results show that temporal compression algorithms are an effective method for reducing the energy consumption of a LoRa sensor node by reducing the number of LoRa transmission periods.

Originality/value – This paper presents with a practical case that it is possible to reduce the overall energy consumption of a wireless sensor node by compressing sensor data in online mode with simple temporal compression algorithms.

Keywords Internet of things, Energy efficiency, Compression, Sensor data, Edge computing

Paper type Research paper

1. Introduction

Sensors are fundamental components of internet of things (IoT) design. According to a report published in 2021, 40% of IoT engineers use environmental sensing in their IoT design, and only 14% do not use sensor technology at all in their IoT design (Farnell, 2021). Typical applications using environmental sensing are, for example, different home control-related applications where the sensors are measuring, for instance, air quality, temperature, humidity and air pressure. Environmental sensors are also widely used in industrial applications and particularly in agricultural applications. In agriculture, precision agriculture (PA) requires up-to-date information from the environment and from the field for decision-making to improve quality and production (Jawad *et al.*, 2017).

Most IoT solutions use wireless connections between the edge device, gateway and cloud. According to the Farnell report, 77% of the engineers who responded to a survey used a wireless connection in their IoT design. Only 23% use wired connectivity (Farnell, 2021). Sensors and wireless sensor networks (WSNs) are fundamental technologies, for example, in smart environment monitoring systems. Smart environment monitoring systems can be used in agriculture for smart farming and for monitoring air quality, water pollution and radiation pollution (Ullo and Sinha, 2020).

Globally, there are already more IoT devices than people. It is estimated that the number of IoT devices will triple from 8.74 billion in 2020 to more than 25.4 billion in 2030 (Farnell, 2021). IoT devices are often battery-powered and will be used in every area of our lives. Therefore, the energy consumption of IoT devices is an important issue. By minimizing the energy consumption, it is possible to lengthen the device or battery lifetime, thus reducing the overall costs. At the same time, it is a well-known fact that the wireless connectivity is the single most energy-consuming task in a wireless IoT sensor node. The most

The current issue and full text archive of this journal is available on Emerald Insight at: **https://www.emerald.com/insight/0260-2288.htm**

Sensor Review
Emerald Publishing Limited [ISSN 0260-2288]
[DOI 10.1108/SR-10-2021-0360]

Received 14 October 2021
Revised 19 March 2022
Accepted 7 June 2022

energy-consuming modes in wireless sensor nodes are when the radio is transmitting, receiving or idle mode (Harrison *et al.*, 2016; Lin *et al.*, 2021). According to Farnell report (Farnell, 2021), Wi-Fi is the most popular type of wireless connectivity followed by cellular (4 G/LTE/5G), Bluetooth low-energy and LoRa. LoRa is a low-power wide-area network technology that is well suited for energy-constrained sensor devices located far from the base station.

In IoT home applications, it is rather easy to power the devices from the mains power supply, but even these are often battery-powered for installation simplicity. In many other sectors, it is not even possible to use the mains power supply to power the devices. Typical applications where the devices need to be battery-powered or energy-harvesting powered are, for example, different agricultural and environmental monitoring applications where IoT devices can be spread to a geographically wide area in the field (Prauzek *et al.*, 2018).

There are many methods for reducing the overall energy consumption of a wireless sensor node. Wireless sensor nodes consume energy mainly in sensing, processing and data communication. At other times, it can be in the sleep mode (Lin *et al.*, 2021). The effective use of sleep modes between sensor measurements is a significant method for reducing energy consumption (Väänänen and Hämäläinen, 2021). Another method to reduce energy consumption in IoT sensor nodes is to lengthen the sampling interval, thus keeping the device in sleep mode for longer periods (Lin *et al.*, 2021). This also reduces the number of transmitting periods with a wireless connection; thus, it is a very effective way to reduce energy consumption. Lengthening the sampling interval results in lower precision in the measured data, as it is not possible to detect sudden and rapid changes when they occur between measurements. It is impossible to obtain any information regarding the measured magnitude changes and the direction of change between measurements.

Different sensor data compression methods are one solution to improve this situation. It is possible to compress the raw sensor data in the sensor node, thus reducing the amount of data required to transmit via wireless connection (Săcăleanu *et al.*, 2018). It is also possible to reduce the number of transmitting periods by using a temporal compression algorithm, and at the same time, obtain the information if the measured values change rapidly. Many temporal compression methods are computationally light and simple. These methods can be used for IoT devices that are computationally constrained and have limited energy resources.

In this study, the energy consumption of LoRa sensor node with different LoRa modulation parameters and temporal compression algorithms was evaluated with practical measurements. This study demonstrates that the sleep mode energy consumption, LoRa modulation parameters and compression algorithm used have a significant effect on the overall energy consumption of an IoT sensor node. The remainder of this paper is organized as follows. The LoRa and LoRaWAN wireless IoT protocols and their basic parameters are presented in Section 2. Section 3 presents the basics of temporal compression algorithms, and the implemented compression algorithms are presented in more detail. The test and measurement setup and measurement challenges are presented in Section 4. The measurement results with different LoRa parameters and implemented algorithms are discussed in Section 5. The combined results and overall energy consumption with different LoRa parameters and algorithm combinations are presented in detail in Section 6. Finally, Section 7 concludes the paper.

2. LoRa and LoRaWAN

LoRaWAN is a low-power wide-area (LPWA or LPWAN) networking protocol developed to be an energy-efficient wireless protocol to connect battery-powered IoT devices to the internet (LoRa Alliance, What is LoRaWAN Specification). LoRaWAN is optimized to extend the battery lifetime, capacity and range of IoT devices as well as to minimize costs.

Several other wireless technologies are available for use in IoT devices. Wi-Fi and Bluetooth low energy are widely used for communication in personal devices, especially for short distances. Cellular technology is suitable for applications in which a large amount of data must be transmitted over a long range. LoRa offers very low power consumption and a long range for transmitting sensor data a few times per hour (LoRa Alliance, What is LoRaWAN Specification). Another low-power and long-range wireless technology is the SIGFOX. Both LoRa and SIGFOX are asynchronous technologies; therefore, nodes can be in sleep mode and wake up only when there is a need to transmit data (Morin *et al.*, 2017). Each wireless technology has its own characteristics, advantages and disadvantages. Thus, there is not one single technology suitable for every application. LoRa is a very potential technology for sensor devices when the transmitted amount of data is rather limited and low-energy consumption is required. Commercial LoRa networks have good geographical coverage. For example, in Finland, the commercial network is operated by Digita, and its network covers almost the entire country if a terminal device is located outdoors (Digita, LoRaWAN network coverage in Finland). Even for indoor devices, the network covers most of the country.

LoRa is a physical layer that includes wireless modulation, enabling long-range connectivity. LoRa uses chirp spread spectrum (CSS) modulation, which enables low power consumption and long range in wireless connectivity at the same time (LoRa Alliance, What is LoRaWAN Specification).

LoRaWAN is a communication protocol and system architecture that uses the LoRa physical layer to achieve a low-power operation and a long communication range. LoRaWAN network uses a star topology in which the nodes are not associated with a specific gateway/base station. The transmitted data can be received by several base stations, and the network side removes redundant packets (Lora Alliance, What is LoRaWAN Specification).

Three different device classes are described in the LoRaWAN protocol. The most energy efficient of the three classes is Class A. The Class A device does not listen to the downlink messages from the network, except for two short time windows after every uplink transmission (LoRa Alliance, What is LoRaWAN Specification). Thus, between the transmitting periods, the device and LoRa radio can be in sleep mode. In addition to effective modulation, operation at the sub-GHz level enables a long communication range. LoRa communication uses an unlicensed industrial, scientific and medical band (Lavric and

Popa, 2018). Sub-GHz frequency range helps signal penetration through the obstacles between the device and base station. Sub-GHz frequency and LoRa modulation enable the long range as well as the good network coverage for devices located indoors.

The CSS modulation spreads the narrowband signal over a large frequency band, thus enabling the signal to be very resistant to noise and immune for interference (Lavric and Popa, 2018). In LoRa CSS modulation, the spread spectrum is achieved with a chirp signal that continuously varies its frequency (Semtech, What are LoRa and LoRaWAN). LoRa supports spreading factors (SFs) from 6 to 12. The higher SF allows a longer range, but it also results in higher time on air (ToA) values and lower data rates (DRs) (Lavric and Popa, 2018; Semtech, What are LoRa and LoRaWAN). For example, according to Semtech (Semtech, What are LoRa and LoRaWAN), the range with the upper link SF10 is 8 km, but with SF7, the range is only 2 km. These values are examples and vary greatly depending on the circumstances. The ToA is correspondingly 371 ms with SF10 and 61 ms with SF7. The range depends significantly on the terrain, but this provides an idea of the SF effect on the range.

The LoRa nodes have the possibility of setting the DR from 0 to 6. The DR represents a predefined set of LoRa settings such as the SF (Lavric and Popa, 2018). The LoRa specification describes the DR settings as presented in Table 1 (LoRa Alliance, RP002-1.0.0 LoRaWAN Regional parameters). The configuration column presents the SF and channel width.

The LoRaWAN protocol defines an adaptive DR (ADR) mechanism, which optimizes the DR used. LoRa devices located close to the base station do not require a high link budget that is with SF12. ADR optimizes the SF used and minimizes the ToA. The ADR has simple rules for changing the DR used. If the link budget is high, then the SF can be increased and vice versa (Semtech, What is an ADR). It is also possible to set the LoRa node to use a certain DR, but the ADR is designed to optimize the SF and ToA, and thus, it should be the recommended setting. The ADR scheme maximizes the battery lifetime. LoRaWAN also supports optional acknowledgments (ACK) and message retransmissions. The LoRaWAN node can indicate whether an ACK is requested in each transmission. If ACK is required, the node is expecting to receive ACK (confirmation) in one of the two receive windows after message transmission. If the ACK is not received, the LoRaWAN node retransmits the message with the same DR as originally, and then DR decreases every two attempts to lower DR until DR = 0 if the ACK is not received. (Casals *et al.*, 2021). If the ACK is not requested, the LoRaWAN node listens to the possible downlink message from the network in any case but does not retransmit the message if the confirmation is not received. The LoRa network server can send a downlink message even if it is not requested.

Table 1 LoRaWAN DR settings

DR	Configuration	Physical bit rate (bit/s)
0	SF12/125 kHz	250
1	SF11/125 kHz	440
2	SF10/125 kHz	980
3	SF9/125 kHz	1,760
4	SF8/125 kHz	3,125
5	SF7/125 kHz	5,470
6	SF7/250 kHz	11,000

3. Temporal compression methods

Many temporal compression methods are well suited for sensor-based 1D data. Data compression is a common method for reducing data size. Compression methods can be divided into lossless and lossy methods. The compression ratios achieved are not very high with lossless methods (Lin *et al.*, 2019). Lossy algorithms can achieve a compression ratio that is several times higher than that of lossless algorithms, but with the cost of reconstruction error (Lu *et al.*, 2021). There is often a temporal correlation in sensor data if the observation window is short. Temporal compression methods use this temporal correlation (Lin *et al.*, 2019).

The temporal compression methods used in this study are simple and computationally lightweight compression algorithms. The methods used in this study are based on data linearity. The environmental magnitudes behave rather linearly if the observation window is short. For example, air temperature in a shadow does not change significantly in seconds. It normally requires minutes to observe the temperature change, even when it is changing at its extreme speed. If the temperature is rising, it changes quite linearly as it behaves similarly also if it is going down.

The compression methods used in this study either find linear segments from the sensor data stream with certain error bound or use linear regression from previous values to predict future values with allowing certain error bound. Thus, the compressed data set loses some information. These methods are computationally light and thus suitable for constrained battery-powered IoT sensor nodes. These methods are also easy to understand and implement.

The compression algorithms used in this paper were lightweight temporal compression (LTC) and two versions of the real-time linear regression-based temporal compression (RT-LRbTC). The two versions of the RT-LRbTC vary from each other by the number of sensor values used to calculate the regression line. Three and four values (N values) versions were tested.

3.1 Lightweight temporal compression

The LTC is a well-known compression algorithm that is particularly suitable for environmental data. It was first presented by Schoellhammer *et al.* (2004). It has also been used to compress sensor data in wireless body sensor networks (WBSN) (Giorgi, 2017). Several modifications of the original LTC have been presented (Parker *et al.*, 2013; Azar *et al.*, 2018; Sarbishei, 2019; Li *et al.*, 2018; Klus *et al.*, 2021).

LTC has proven to be a very effective compression algorithm, particularly for linearly behaving environmental sensor data (Väänänen and Hämäläinen, 2019). One major disadvantage of LTC is its unpredictable latency. As the LTC compresses the data in the online mode by finding the best and longest linear segment from the incoming sensor data, it sends the linear segment endpoint to the sink only when the algorithm finds it as the new value falls off from the linear segment. If the

method compresses the data very efficiently, then the transmitting intervals become long, and the receiving side does not even know in which direction the values are changing. Thus, LTC is not well suited for compressing sensor data in real-time or near-real-time applications (Giorgi, 2017).

In this study, the LTC was used as in its original version presented by Schoellhammer *et al.* (2004). The same version was used by Väänänen and Hämäläinen (2019, 2020) as MATLAB version. In this study, the LTC was programmed for the Arduino and implemented on the Arduino MKR WAN 1310 LoRa board.

The LTC itself is computationally light because with every new value, it is only necessary to make a comparison between the new value with error bound extremes and the previously calculated upper and lower limit lines. As a result of the comparison, a maximum of two new lines must be calculated to create new upper and lower limit lines that will be used with the next value. Calculating the new limit line parameters (slope and y-intercept) requires one division, one multiplication, one summation and subtractions, thus resulting in a computationally simple algorithm.

3.2 Real-time linear regression-based temporal compression

RT-LRbTC uses linear regression calculated from previous sensor values to predict future sensor values. This type of compression algorithm works well if the measured data behaves rather linearly. This is the case for many environmental magnitudes, such as temperature, humidity and air pressure.

RT-LRbTC is based on several other simple linear regression-based algorithms. Other simple linear regression-based compression algorithms have also been developed (Väänänen and Hämäläinen, 2019; Hung *et al.*, 2013; Duvignau *et al.*, 2019). RT-LRbTC was originally presented by Väänänen and Hämäläinen (2020). RT-LRbTC was developed especially for compressing sensor data in online mode. It has a shorter inherent latency than other linearity-based compression methods, which is its most significant benefit compared to other methods (Väänänen and Hämäläinen, 2020).

The inherent latency of the RT-LRbTC algorithm is one measurement interval Δt in the linear section. The algorithm uses N previously measured values to calculate the regression line, which predicts future values with a certain error bound (ε) allowed from the line. New measured sensor value is compared to the previously calculated regression line. If the difference from the line is smaller than ε, then the algorithm waits for the next measured value. When the new value falls off from the linear section (distance greater that the error bound ε from the line), then the new line is calculated from the values already available. From the calculated line, the line parameters and time stamp are sent/stored. On the network side, if new parameters are not received, then the values follow the previous regression line with the error bound allowed. N is a minimum of three, and in this study, N was three and four (two versions). Calculating the new regression line requires some calculation: 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 (sample number) and y_k is the measured value. $N = 3$ is the basic form of RT-LRbTC, and $N = 4$ version was also tested in this study to see whether the required calculations had any effect on the overall energy consumption.

The RT-LRbTC algorithm was tested with real measured temperature data by Väänänen and Hämäläinen (2020). In the data sets used, the measurement interval was 10 min. The data sets were obtained from the Finnish Meteorological Institute's open data service. The data sets used were two full-year data sets with a 10-min measurement interval. As a result, with 0.5° C error bound, the RT-LRbTC algorithm has achieved compression ratio CR = 5.5–6.0 (Väänänen and Hämäläinen, 2020). The error bound ε = 0.5°C represents the maximum difference between the original measured values and reconstructed values from the compressed data. The average reconstruction error is smaller than the error bound used. The compression ratio achieved with a certain linearity-based compression algorithm depends on the measured magnitude characteristics and error bound used. A higher error bound results in a better compression ratio, but with the cost of a larger reconstruction error.

4. LoRa device energy consumption with compression algorithm implemented

The LoRa SF used determines the ToA and thus it also determines how much energy a transmission consumes. This is because if the ToA is longer, then the LoRa radio is transmitting for a longer time and consumes energy for a longer time as well.

Väänänen and Hämäläinen (2021) measured the LoRa device energy consumption with ADR set on, and thus, the LoRa device was transmitting with a DR of 2 (SF10). The device was in a stable place, and thus, the conditions did not change, and the DR remained constant. The only difference between the energy consumption of the transmitting periods is the difference between whether the downlink is received or not. If the downlink is received, the energy consumption increases as the device receives the data. If the downlink is not received, then the LoRa device only listens shortly during the two receive windows, and the overall energy consumption is lower. The downlink message is used by the network server for acknowledge messages (ACK) (Maudet *et al.*, 2021).

In this study, a setup similar to that of Väänänen and Hämäläinen (2021) was used for the practical experiments and energy consumption measurements. In this study, the ADR was not on, and the DR was set (fixed) for every transmitting period. The DRs tested ranged from DR0 to DR5. All of these DRs have a channel width of 125 kHz. The downlink SF was automatically set from the network and was not controlled in this study. Normally, a downlink message has the same SF as an uplink message if the first receive window is used. For the second receive window, the SF12 is used as the default (Casals *et al.*, 2021).

The LoRa device used was Arduino MKR WAN 1310 board. The Arduino MKR WAN 1310 has a Microchip SAM D21 32-bit Arm Cortex-M0+ based microcontroller and a Murata CMWX1ZZABZ LoRa module (Arduino MKR WAN 1310). The Arduino MKR WAN, 1310 was chosen because of its simplicity and ease of use. It is also very popular among hobbyists but is also widely used in industry for piloting and experiments.

The temperature was measured using a DHT22 sensor. The DHT22 also measures humidity, but for this experiment, only

temperature was used. DHT22 is a low-cost sensor with a digital output. DHT22 sensor has measurement resolution for temperature 0.1°C and its accuracy is $< \pm 0.5$°C (DHT22 temperature and humidity sensor). Another popular and rather similar temperature and humidity sensor is DHT11, which is often used in agricultural applications (Rehman *et al.*, 2022). In this study, DHT22 was chosen because it is slightly more accurate and has wider measurement ranges; however, it is only slightly more expensive. It is well suited for this type of experiment, where the accuracy requirement is not high.

The Arduino MKR WAN 1310 was powered by a 2,000 mAh lithium-polymer (LiPo) battery. The LiPo battery has a 3.7 V nominal voltage, resulting theoretically 7.4 Wh total energy capacity, which equals 26,640 Ws. The battery has a JST-PH connector that can be directly used with the Arduino MKR WAN 1310 board. The energy consumption of the Arduino board is lower when powered by a battery than when powered by the USB port. If the board is powered by the USB port, then one LED is always ON, and also USB IC-chip consumes extra energy. The device setup is illustrated in Figure 1. The setup was built for this experiment only, but it could be used in real practical case when enclosed properly against environmental conditions.

4.1 Measurement setup

For the experimental energy consumption measurements, the Arduino board was set to measure the temperature with the DHT22 sensor at regular intervals. After each measurement, the compression algorithm was applied by microcontroller. If the result of the algorithm required data transmission, then the data were sent via the LoRa connection. After every measurement event and possible data transmission, the device was set to go to deep-sleep mode. The device woke up only for the measurement, compression algorithm and possible transmission periods. The LoRa network was a commercial network operating in Finland.

The current consumption in the active mode was measured using a shunt resistor. Two oscilloscope channels were used to measure the voltage across a 10 Ω shunt resistor, which was in series at the battery plus wire. Both oscilloscope channels used battery negative terminal as the reference level. The measurement setup is illustrated in Figure 2(a). Current

Figure 1 Arduino MKR WAN 1310 in test setup

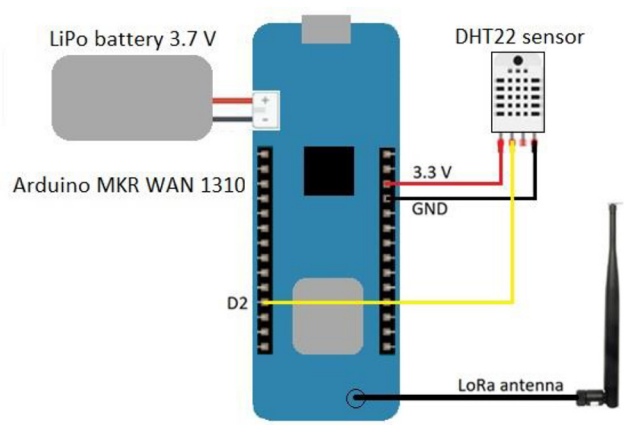
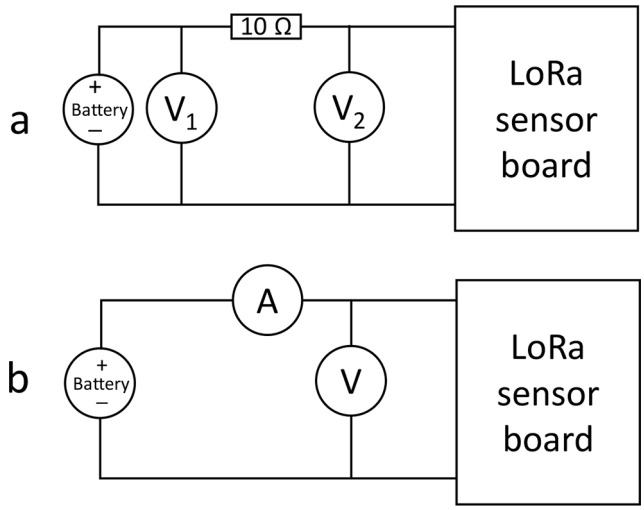


Figure 2 Current consumption measurement circuits



consumption was calculated from the measured voltages $I = (V_1 - V_2)/R$, where R is the shunt resistor (10 Ω). V_2 can also be used to measure the supply voltage in the battery connection of the board. Thus, the power consumption of the device is:

$$P = V \cdot I = V_2 \cdot \frac{(V_1 - V_2)}{R} \quad (1)$$

In a previous study by Väänänen and Hämäläinen (2021), current was measured using an oscilloscope current probe. That kind of setup is shown in Figure 2(b), where the ammeter is the current probe. The current probe is very easy to use, but its accuracy is poor when measuring low-level values. The shunt resistor measurement of the current is more accurate and repeatable. The results obtained by Väänänen and Hämäläinen (2021) were used in this paper for comparison when available.

In this study, the current consumption in the deep-sleep mode was measured using a digital multimeter (DMM) in series with a battery wire. The DMM is more accurate for measuring low-level deep-sleep current than the shunt resistor method used for active periods. A high-quality DMM can reliably measure μA level current value if it remains stable. The DMM cannot be used to measure current consumption for active periods because the current level is changing and does not remain static.

The device used for the measurements was a Tektronix MSO 4104 mixed signal oscilloscope with a 1 GHz measurement bandwidth, and the probes used were TAP1500 active voltage probes. The voltage in the battery connector of the board was also measured separately using a Tektronix P6139A passive voltage probe. The DMM used was a Tenma RS232C TrueRMS model.

5. Measurement results

The measurements were carried out without implementing a compression algorithm and with compression algorithms to see whether the algorithm calculations have any effect on energy consumption. The energy consumption for sensor measurement and algorithm calculations was measured with all

algorithm combinations and without an algorithm. Transmission energy consumption was measured using two different payloads. If the measured raw or compressed value is sent with a time stamp, only 8 bytes are needed to transmit. This is the case without a compression algorithm or with the LTC algorithm. With the RT-LRbTC algorithm, a total of 12 bytes are transmitted because two line parameters (slope and base) and a time stamp are needed to transmit.

Figure 3 presents the overall LoRa sensor node energy consumption scenario over time. Number 1 in Figure 3 represents deep-sleep energy consumption (base consumption). Number 2 represents the extra energy consumption of the sensor measurement and data processing (with or without the compression algorithm implemented). It occurs at regular intervals that are determined by the measurement interval. Number 3 represents the energy consumption of the LoRa transmission. LoRa transmission is required after every measurement event if no compression algorithm is implemented. If a compression algorithm is implemented, LoRa transmission does not occur after every measurement event. The deep-sleep energy consumption is presented here as existing all the time, and the measurement events and LoRa transmissions were presented and measured as extra energy consumption on top of deep-sleep base energy consumption. When the device is measuring or transmitting, it is not in deep-sleep mode, but for measurement purposes, this type of presentation is easier. In any case, the deep-sleep power consumption is a fraction of the measurement event and/or transmitting event power consumption.

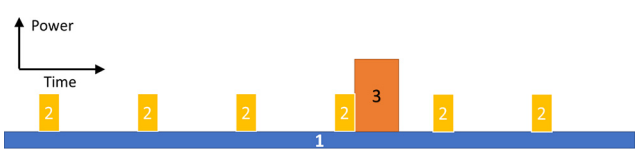
5.1 Deep-sleep current consumption

The deep-sleep current consumption was measured without implementing an algorithm, and the result was confirmed with algorithms implemented. As the device was in deep-sleep mode, there was no difference if the algorithm was implemented or not. The device was set to go into deep-sleep mode between the measurement periods. A similar setup was used by Väänänen and Hämäläinen (2021), and the results obtained in this study were in same level. The measured deep-sleep current was 106–107 μA. When the device goes into deep-sleep mode, the current drops immediately to 150–160 μA, and after 10–20 s, it reaches 107 μA level. The voltage in the battery connector of the board was measured with an oscilloscope at the same time. The battery voltage was 3.99 V in this case, thus resulting in deep-sleep power consumption $P_{ds} = V_{battery} \cdot I_{ds} = 3.99\,\text{V} \times 117 \cdot 10^{-6}\,\text{A} = 4.6683 \cdot 10^{-4}\,\text{W} = 0.46683\,\text{mW}$.

5.2 Sensor measurement and algorithm energy consumption

As the measurement interval is typically minutes, the sensor node remains most of the time in deep-sleep mode, but it wakes up with a regular basis to perform the measurement and algorithm calculations. If the result from the algorithm calculation is that there is no need to transmit any data, then the device returns to the deep-sleep mode. The real-time clock (RTC) runs even in deep-sleep mode and wakes the device up on a regular basis, which is determined by the measurement interval.

Figure 3 LoRa sensor node overall energy consumption scenario



The energy consumed by the sensor measurement, and possible algorithm was measured with the oscilloscope using a shunt resistor, as explained in the measurement setup section. The oscilloscope measurement results are shown in Figure 4. Channels 2 and 3 (blue and purple lines on top of one another) were used to measure the voltage difference across the shunt resistor. Channel 4 (green line) measured the voltage in the battery connector. The power line (red MATH-line in mW) was calculated using the oscilloscope MATH-function from Channels 2, 3 and 4 data: $P = U \times I = \text{CH4} \times ((\text{CH2} - \text{CH3})/10)$. CH2-CH3 denotes the voltage across the shunt resistor. 10 is the resistor size in ohms. The oscilloscope measurement function was used to calculate the MATH line area (integral), which is the total energy consumed in the oscilloscope window timescale (5.022 mWs during 2 s in Figure 4). Then, the average value of the red MATH-line was measured before the measurement event (device wake up) when the device was in deep-sleep mode. The average deep-sleep value (average power) was multiplied by the timescale used in the oscilloscope screen (2 s in Figure 4) to obtain the base energy consumed, which was subtracted from the total energy measured (5.022 mWs in Figure 4). Thus, additional energy consumption from the sensor measurement and data processing was measured.

The same measurement was repeated a minimum of ten times for each algorithm implemented as well as without the algorithm implemented. The measurement results are listed in Table 2. The average value was calculated from all measurements using a certain algorithm (a minimum of ten measurements with each algorithm). Max and Min values show the maximum and minimum measured values, and Std Dev is the standard deviation calculated from all the measured values with the certain algorithm implemented. Last row presents measurement results with current probe (Väänänen and Hämäläinen, 2021). Figure 5 shows the average results with the maximum, minimum and standard deviation values for each algorithm.

It can be seen from the results presented in Table 2 and Figure 5 that the effect of the algorithm on the measurement and data acquisition event energy consumption is negligible. The algorithms implemented and evaluated were computationally so light that the possible effect on the energy consumption was smaller than the measurement inaccuracy.

5.3 LoRa transmission energy consumption

The scenario for the LoRa transmission energy consumption is shown in Figure 6 (as a function of time). Number 1 in the figure is the base energy consumption, which is the deep-sleep energy consumption. Number 2 represents the sensor measurement, data acquisition and algorithm energy consumption in addition to base energy consumption. Number 3 is the LoRa transmission uplink, and number 4 is the LoRa transmission downlink if received (in Figure 3, the uplink and downlink energy consumptions are combined and presented by number 3).

Figure 4 Sensor measurement and data processing energy consumption

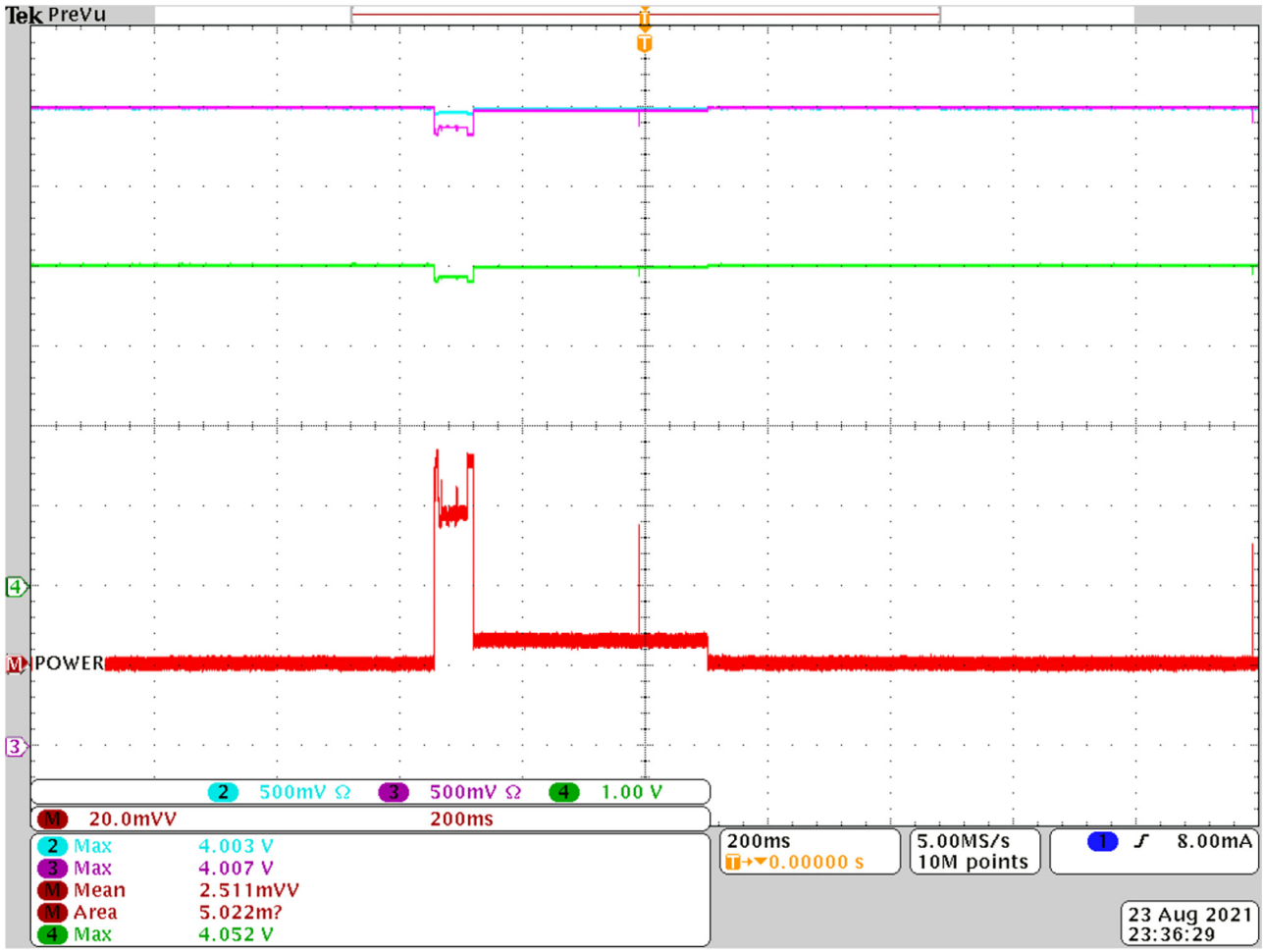


Table 2 Sensor measurement and algorithm energy consumption with and without algorithms implemented. Results are in mWs

	No compression (mWs)	RT-LRbTC, $N = 3$ (mWs)	RT-LRbTC, $N = 4$ (mWs)	LTC (mWs)
Max	5.05	5.24	5.22	5.30
Average	**4.88**	**4.96**	**4.98**	**4.95**
Std dev	0.11	0.18	0.16	0.16
Min	4.70	4.64	4.68	4.75
From current probe measurement	4.57	4.78	4.83	4.78

In this measurement setup, the overall energy consumption was measured using the oscilloscope from the timescale that was visible on the oscilloscope screen. The oscilloscope MATH-function was used to calculate the overall power (red MATH-line in Figures 7, 8 and 9) line in mW, and its area (integral) was calculated using the oscilloscope measurement function (in mWs). Then, the average base power level was measured using the oscilloscope ZOOM function from the time before the device wakes up (in deep-sleep mode). The average power was used to calculate the average base energy consumption at that timescale (oscilloscope screen, 10 s in Figures 7, 8 and 9). This average base energy (number 1 in Figure 6) was subtracted from the overall measured energy consumption. This results

Figure 5 Sensor measurement and possible algorithm overall energy consumption

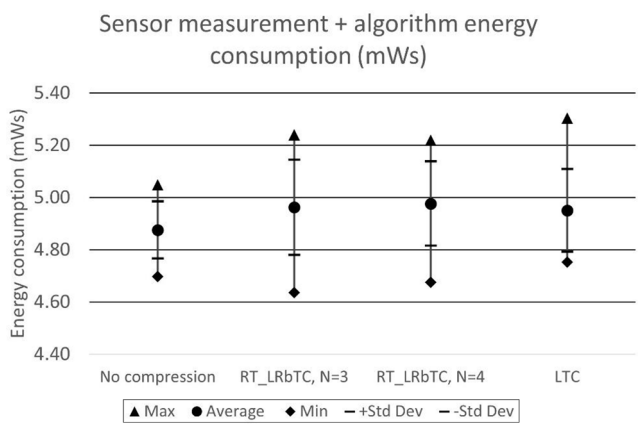
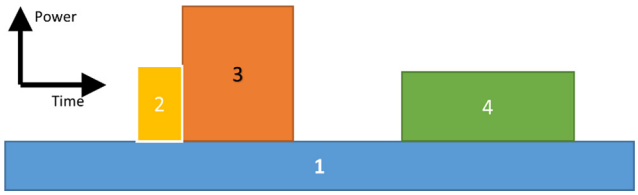


Figure 6 LoRa node energy consumption scenario



in extra energy that the sensor, algorithm and LoRa transmission add to the base energy consumption.

The sensor measurement and algorithm effect (Table 2, which is number 2 in Figure 6) was then subtracted from the measurement results, resulting in transmission-only energy consumption. LoRa transmission energy consumption was measured for 8 bytes (for LTC and no compression cases) and 12 bytes (for RT-LRbTC cases) payload situations with every SF (SF7-SF12). The DR can be adjusted in Arduino MKR WAN 1310 by enabling ADR and setting a certain DR value:

- modem.setADR(true); and
- modem.dataRate (0);//set data rate to be 0-5.

This needs to be done before every transmission period when the radio wakes up. The total transmission energy consumption for every SF case was measured a minimum of ten times, and the average values are listed in Table 3.

The significant differences in energy consumption depending on the SF used can be seen in Table 3. The downlink was sent from the network side every time, even though the ACK was not required, but quite often, it was not received. If the downlink is not received, then the transmission period energy consumption is lower, but that situation should not be the normal case. If these values are used to predict the device lifetime, the values with the downlink received should be used as the worst case for the energy consumption.

The difference between the received and unreceived downlinks can be seen in Figures 7 and 8. In Figure 8, the LoRa radio opens two short receive windows after transmission. The first receive window is approximately 1 s after the transmission, and the second window is 1 s after the first window. In Figure 7, the LoRa radio opens the first receive window 1 s after transmitting, and in this case, the LoRa radio receives the downlink message. The SF effect on the ToA is shown in Figures 7, 8 and 9. In Figures 7 and 8 with SF12, the transmission takes approximately 1,500 ms, while with SF8 (in

Figure 7 Lora transmission (8 bytes) with uplink SF12 and downlink SF12

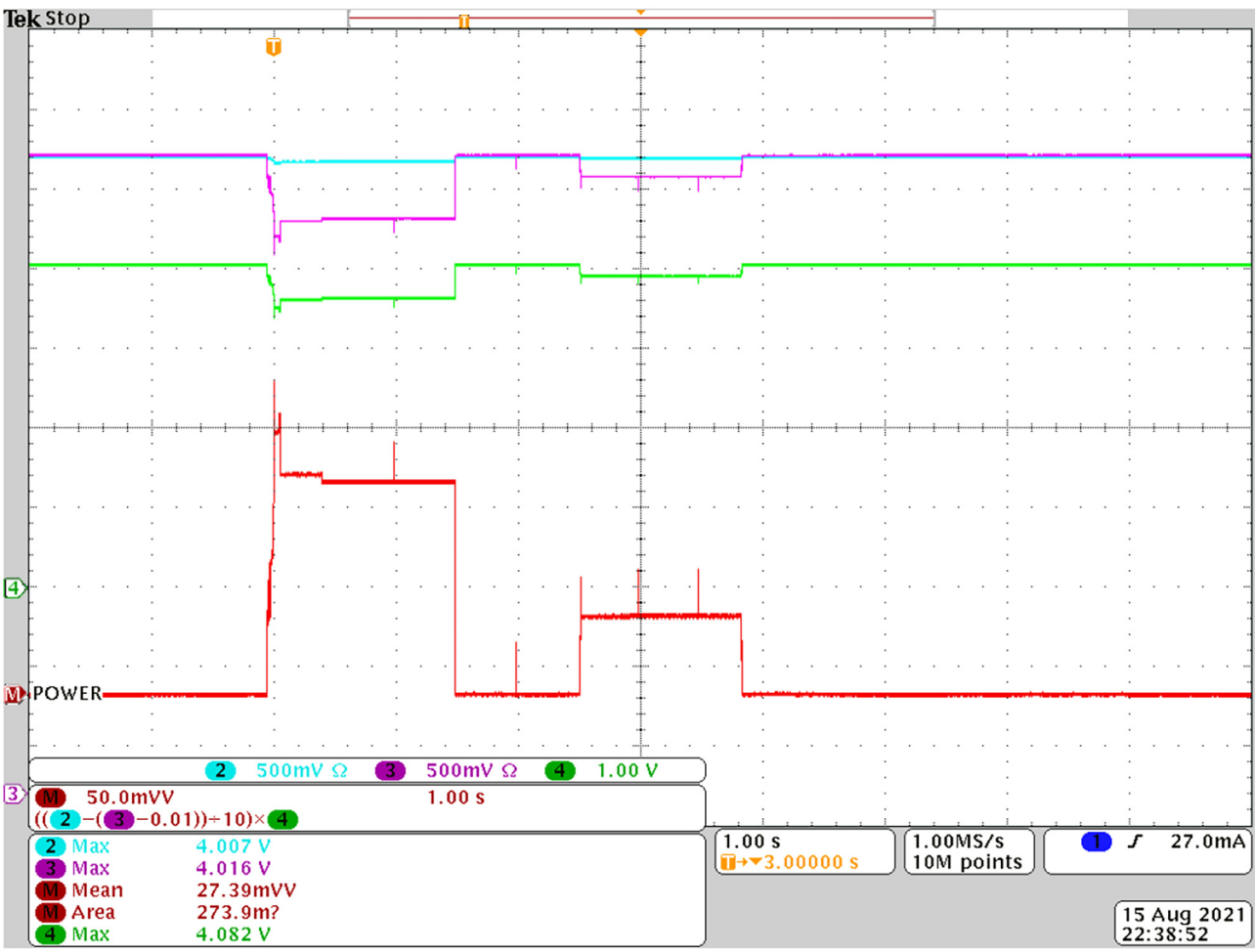


Figure 8 LoRa transmission (8 bytes) with uplink SF12 and downlink not received

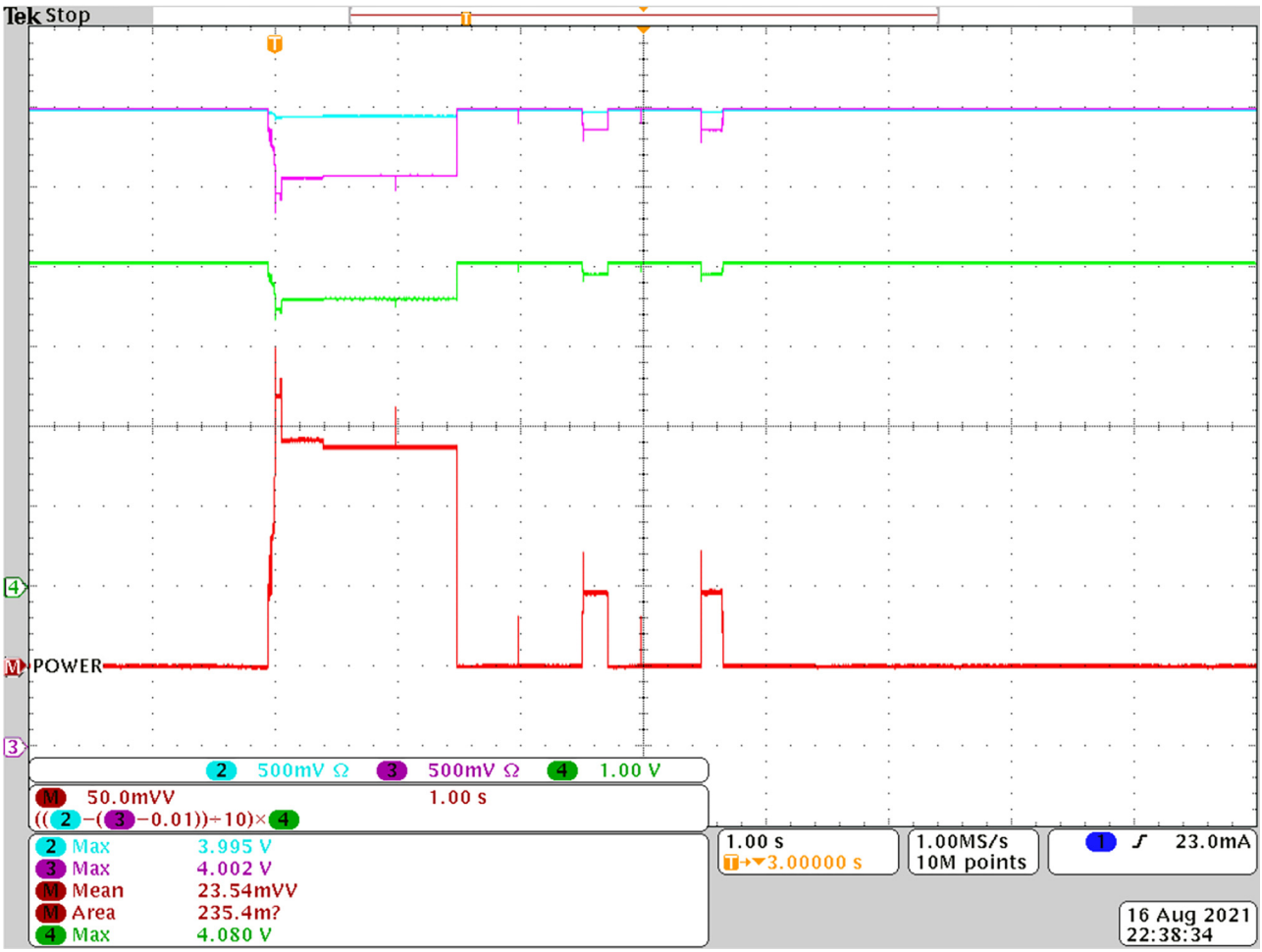


Figure 9), it takes approximately 100–200 ms. That explains the significant difference in energy consumption.

6. Overall energy consumption and average power consumption

The overall energy consumption of the LoRa sensor node is a combination of deep-sleep consumption, measurement event consumption, algorithm execution and possible LoRa transmission. In this study, the sensor measurement event and the algorithm execution were combined. The sensor measurement and algorithm calculations (W_M) take place on a regular basis and are determined by the measurement interval (Δt). The LoRa transmission event (energy consumed W_S) is determined by the measurement interval but also by the compression ratio (CR) that the algorithm achieves. The overall energy consumed during time t_x can be estimated using equation (2):

$$W_{tot} = P_{ds}t_x + \frac{t_x}{\Delta t}W_M + \frac{t_x}{CR \times \Delta t}W_s \tag{2}$$

where $P_{ds}t_x$ is the energy consumed by the device in the deep-sleep mode during time t_x. $t_x/\Delta t$ is the number of measurement periods. $t_x/(CR \times \Delta t)$ is the number of transmission periods. It can be seen from the equation that it is possible to minimize the overall energy consumption either by lengthening the measurement interval or using a compression algorithm, which results in a high compression ratio for the measured data stream. Other possibilities would require different hardware solutions.

If the total available energy is known (battery capacity for example), then the overall lifetime can be solved from equation (2):

$$t_x = \frac{W_{ToT}}{P_{ds} + \frac{W_M}{\Delta t} + \frac{W_S}{CR \cdot \Delta t}} \tag{3}$$

The average power consumption can be derived from equation (2) by dividing by time t_x as $P = W/t$. Resulting in equation (4):

$$P_{avg} = P_{ds} + \frac{W_M}{\Delta t} + \frac{W_S}{CR \cdot \Delta t} \tag{4}$$

The DHT22 temperature sensor has an accuracy of ±0.5°C. Thus, it was reasonable to use the error bound value $\varepsilon = 0.5$°C for the compression algorithms. Väänänen and Hämäläinen (2020)

Figure 9 LoRa transmission (8 bytes) with uplink SF8 and downlink SF10

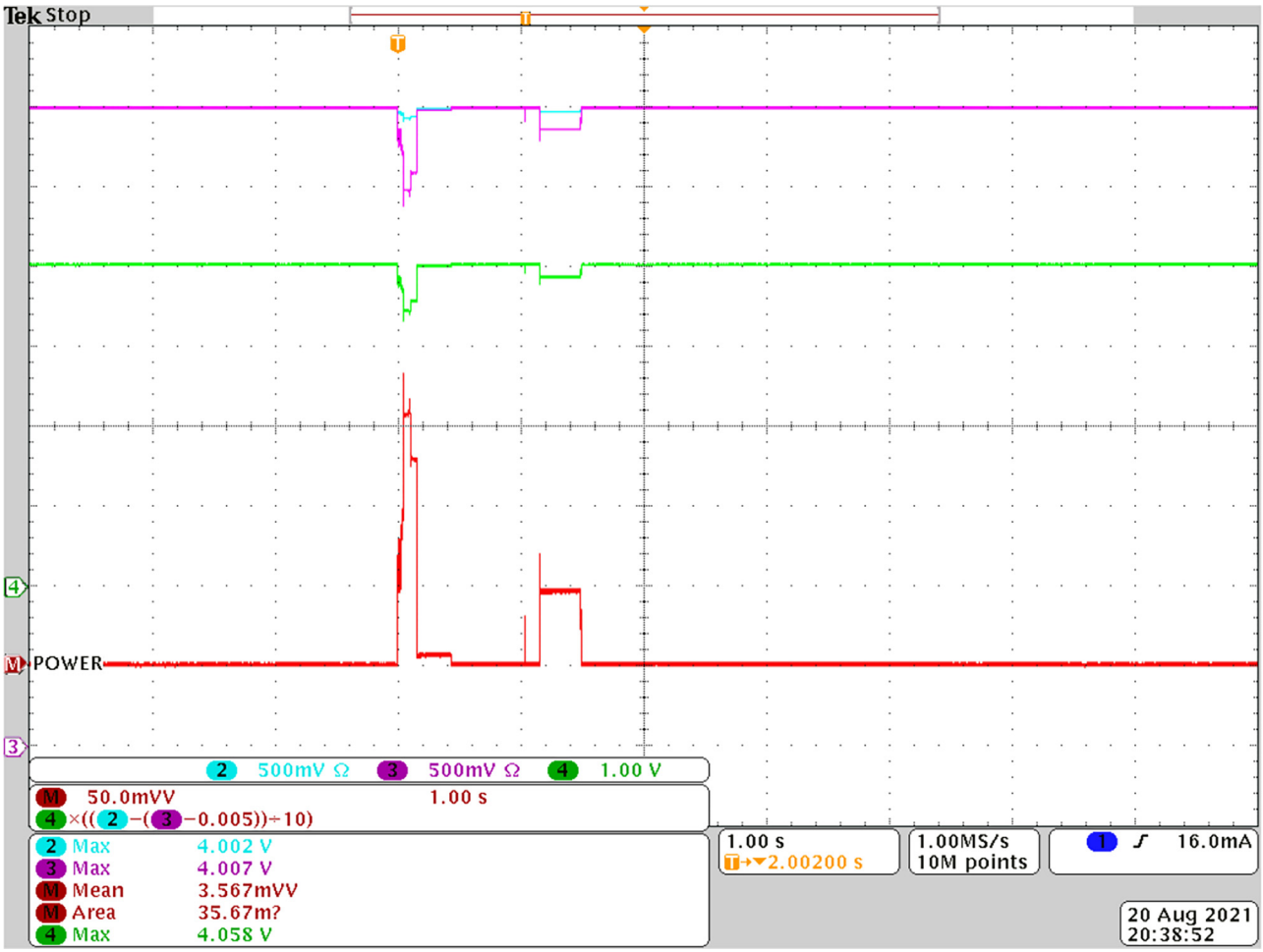


Table 3 LoRa Transmission energy consumption with different SFs and two different payloads. Results are in mWs

	8 bytes (mWs)	12 bytes (mWs)
Uplink SF12, downlink SF12	248.89	257.82
Uplink SF12, downlink SF9 (downlink not received)	201.02	212.32
Uplink SF11, downlink SF12	163.93	162.73
Uplink SF11, downlink SF9 (downlink not received)	119.63	120.12
Uplink SF10, downlink SF12	104.41	107.60
Uplink SF10, downlink SF9 (downlink not received)	65.28	68.03
Uplink SF9, downlink SF11	59.55	58.60
Uplink SF9, downlink SF9 (downlink not received)	41.01	42.50
Uplink SF8, downlink SF10	32.54	33.26
Uplink SF8, downlink SF9 (downlink not received)	27.89	28.78
Uplink SF7, downlink SF9	18.11	19.09
Uplink SF7, downlink SF9 (downlink not received)	13.99	No data
Uplink SF7, downlink SF9 (downlink received in second window)	20.32	No data

tested the LTC and RT-LRbTC algorithms for real temperature data sets with a 10-min measurement interval. Temperature data sets were obtained from the Finnish Meteorological Institute's open data service. The data sets used were 2018 and 2019 full-year data from the Salla Naruska measurement station. The temperature data were measured in degrees Celsius, with a resolution of 0.1°. With ε = 0.5°C, the compression algorithms achieved compression ratios of CR = 9.5–10.2 with LTC and CR = 5.5–6.0 with RT-LRbTC (N = 3).

Table 4 lists the average power consumption for different compression algorithms with different SF scenarios. The measurement interval was 10 min (Δt = 600 s). The values in

Table 4 Average power consumption with different algorithms implemented and with certain compression ratios. Results are in mW

	No compression (mW)	RT_LRbTC, $N = 3$ (mW)	RT_LRbTC, $N = 4$ (mW)	LTC (mW)
Uplink SF12, downlink SF12	0.898	0.548	0.548	0.517
Uplink SF12, downlink SF9 (downlink not received)	0.818	0.535	0.535	0.509
Uplink SF11, downlink SF12	0.756	0.522	0.522	0.503
Uplink SF11, downlink SF9 (downlink not received)	0.682	0.510	0.510	0.496
Uplink SF10, downlink SF12	0.657	0.506	0.506	0.493
Uplink SF10, downlink SF9 (downlink not received)	0.592	0.495	0.495	0.487
Uplink SF9, downlink SF11	0.582	0.493	0.493	0.486
Uplink SF9, downlink SF9 (downlink not received)	0.551	0.488	0.488	0.483
Uplink SF8, downlink SF10	0.537	0.486	0.486	0.481
Uplink SF8, downlink SF9 (downlink not received)	0.530	0.484	0.484	0.481
Uplink SF7, downlink SF9	0.513	0.482	0.482	0.479
Uplink SF7, downlink SF9 (downlink not received)	0.506	–	–	0.478
Uplink SF7, downlink SF9 (downlink received in second window)	0.517	–	–	0.479

Table 4 were calculated using equation (4) from the measured values from Tables 2 and 3 (8 bytes results for LTC and no compression, 12 bytes results for RT-LRbTC algorithms). The compression ratios were $CR = 10$ for LTC and $CR = 6$ for both RT-LRbTC algorithms, which are realistic values and were achieved by Väänänen and Hämäläinen (2020).

There were no significant differences in the power consumption values between the algorithms tested. The differences were larger for high SF values when the effective compression algorithm can achieve energy savings by reducing the number of LoRa transmission periods. LoRa transmission periods are significant energy consumers, particularly if the used SF is high. With high SF values, the compression algorithms used were very effective for reducing energy consumption. This is clearly shown in Figure 10 (from Table 4).

The battery used in this experiment was a 2,000 mAh LiPo battery with 3.7 V nominal voltage. Its overall capacity is 7.4 Wh, which is 26,640 Ws. This capacity is the nominal capacity in the optimal situation. For example, in cold weather, the capacity of lithium-based batteries significantly collapses (Li *et al.*, 2017). Aging also affects the battery capacity.

Figure 10 LoRa node power consumption with different SF values

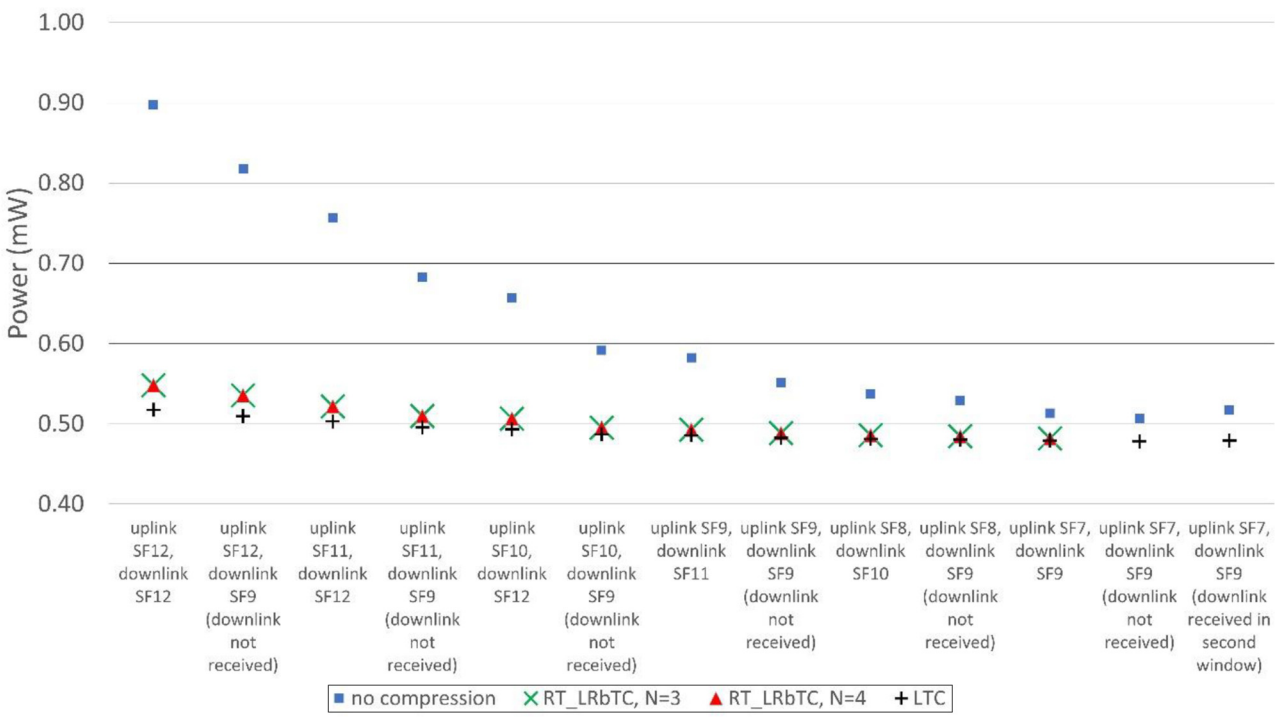


Table 5 Battery lifetime in days with different scenarios

	No compression (days)	RT-LRbTC, N = 3 (days)	LTC (days)
Uplink SF12, downlink SF12	343.4	562.6	596.0
Uplink SF12, downlink SF9 (downlink not received)	376.9	575.8	605.3
Uplink SF11, downlink SF12	407.7	591.0	612.7
Uplink SF11, downlink SF9 (downlink not received)	451.8	604.8	621.9
Uplink SF10, downlink SF12	469.2	608.9	625.0
Uplink SF10, downlink SF9 (downlink not received)	520.9	622.4	633.4
Uplink SF9, downlink SF11	529.5	625.7	634.7
Uplink SF9, downlink SF9 (downlink not received)	559.1	631.5	638.7
Uplink SF8, downlink SF10	574.2	634.8	640.6
Uplink SF8, downlink SF9 (downlink not received)	582.2	636.4	641.6
Uplink SF7, downlink SF9	600.7	640.0	643.8
Uplink SF7, downlink SF9 (downlink not received)	608.9	No results	644.7
Uplink SF7, downlink SF9 (downlink received in second window)	596.5	No results	643.3

The battery lifetime was calculated using equation (3) in the case where full capacity was available. The values used were W_{TOT} = 26,640 Ws and P_{ds} = 4.6683·10^{-4} W. W_M and W_S values were from the Tables 2 and 3. *CR* values of 10 for LTC and 6 for RT-LRbTC. The battery lifetimes in days for different algorithms and SF scenarios are listed in Table 5. RT-LRbTC with $N = 4$ is not presented here because its results are very close to RT-LRbTC with $N = 3$.

As can be seen from Table 5, it is easy to achieve over an 18-month lifetime if the RT-LRbTC algorithm is used (with a 0.5° error bound). Without implementing a compression algorithm, it is possible to have less than a 12-month lifetime if the device is located at a long distance from the base station, or if there are obstacles between the device and base station (thus using a high SF value). In a good network coverage situation, the difference is small between the compression algorithm used or without compression algorithm implemented. Generally, the deep-sleep current consumption of 107 μA is rather high for a modern microcontroller-based LoRa node, and it determines the overall lifetime.

In research by Väänänen and Hämäläinen (2021), the DR was not fixed, and instead, the ADR was used. Thus, in that case, the LoRa node was always transmitting with SF10, and in approximately 50% of the transmitting periods, the downlink was received (SF12). The average transmission energy consumption was measured and calculated to be 91.47 mWs. Using this value, the battery lifetime was calculated to be approximately 490 days if no compression algorithm was implemented, 630 days if the LTC algorithm was used and 616 days if the RT-LRbTC algorithm was used. This case is valid in that situation; however, in some other circumstances, the LoRa node may use other SF values, and its effect on the lifetime can be estimated by the results presented in Table 5.

7. Conclusions

From the results achieved in this study, the LoRa DR has a significant effect on the overall power consumption of the LoRa sensor node, especially if no compression algorithm is used. However, normally it is not possible to control the DR because the ADR adjusts the optimal SF value. If the base station is very far away, then a high SF must be used to achieve that long range, resulting in higher power consumption.

Simple temporal compression algorithms are very effective for reducing the overall energy consumption of the LoRa sensor node if the reconstruction error, determined by the error bound, is acceptable. From the results achieved in this study, the algorithm calculations did not have a significant effect on energy consumption. Nevertheless, these algorithms can significantly reduce the number of LoRa transmission periods and thus achieve significant energy consumption savings. The overall reduction in energy consumption was due to the reduced number of radio transmission periods. The LTC algorithm is very effective and simple algorithm, but its unpredictable latency is not well suited for online applications with latency requirements. RT-LRbTC is not as effective compression algorithm, and it is a bit more complicated, but with predictable latency, it is well suited for compressing environmental data in the online mode. In this research, the measurement interval was rather long, and thus, the LoRa node deep-sleep consumption became a significant factor determining the device lifetime.

References

Azar, J., Makhoul, A., Darazi, R., Demerjian, J. and Couturier, R. (2018), "On the performance of resource-aware compression techniques for vital signs data in wireless body sensor networks", *IEEE Middle East and North Africa Communications Conference (MENACOMM), Jounieh*, pp. 1-6, doi: 10.1109/MENACOMM.2018.8371032.

Casals, L., Gomez, C. and Vidal, R. (2021), "The SF12 well in LoRaWAN: problem and end-device-based solutions", *Sensors*, Vol. 21 No. 19, p. 6478, doi: 10.3390/s21196478.

Duvignau, R., Gulisano, V., Papatriantafilou, M. and Savic, V. (2019), "Streaming piecewise linear approximation for efficient data management in edge computing", *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. Association for Computing Machinery, New York, NY, pp. 593-596.

Farnell (2021), "2021 Global IoT trends report", available at: https://uk.farnell.com/iot-trends-2021 (accessed 6 October 2021).

Giorgi, G. (2017), "A combined approach for real-time data compression in wireless body sensor networks", *IEEE Sensors Journal*, Vol. 17 No. 18, pp. 6129-6135, doi: 10.1109/JSEN.2017.2736249.

Harrison, D., Burmester, D., Seah, W. and Rayudu, R. (2016), "Busting myths of energy models for wireless sensor networks", *Electronics Letters*, Vol. 52 No. 16, pp. 1412-1414, available at: https://doi.org/10.1049/el.2016.1591

Hung, N.Q.V., Jeung, H. and Aberer, K. (2013), "An evaluation of model-based approaches to sensor data compression", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25 No. 11, pp. 2434-2447, doi: 10.1109/TKDE.2012.237.

Jawad, H.M., Nordin, R., Gharghan, S.K., Jawad, A.M. and Ismail, M. (2017), "Energy-efficient wireless sensor networks for precision agriculture: a review", *Sensors*, Vol. 17 No. 8, p. 1781, doi: 10.3390/s17081781.

Klus, L., Klus, R., Lohan, E.S., Granell, C., Talvitie, T., Valkama, M. and Nurmi, J. (2021), "Direct lightweight temporal compression for wearable sensor data", *IEEE Sensors Letters, 2021*, doi: 10.1109/LSENS.2021.3051809.

Lavric, A. and Popa, V. (2018), "Performance evaluation of LoRaWAN communication scalability in large-scale wireless sensor networks", *Wireless Communications and Mobile Computing*, Vol. 2018, pp. 1-9, doi: 10.1155/2018/6730719.

Li, Q., Lu, D., Zheng, J., Jiao, S., Luo, L., Wang, C. and Xu, W. (2017), "Li+-desolvation dictating Lithium-Ion battery's low-temperature performances", *ACS Applied Materials & Interfaces*, Vol. 9 No. 49, pp. 42761-42768, available at: https://doi.org/10.1021/acsami.7b13887

Lin, J.W., Liao, S.W. and Leu, F.Y. (2019), "Sensor data compression using bounded error piecewise linear approximation with resolution reduction", *Energies*, Vol. 12 No. 13, p. 2523, doi: 10.3390/en12132523.

Lin, D., Wang, Q., Min, W., Xu, J. and Zhang, Z. (2021), "A survey on energy-efficient strategies in static wireless sensor networks", *ACM Transactions on Sensor Networks*, Vol. 17 No. 1, pp. 1-48, doi: 10.1145/3414315.

Li, B., Sarbishei, O., Nourani, H. and Glatard, T. (2018), "A multi-dimensional extension of the lightweight temporal compression method", *IEEE International Conference on Big Data (Big Data), Seattle, WA, USA*, pp. 2918-2923, doi: 10.1109/BigData.2018.8621946.

Lu, S., Xia, Q., Tang, X., Zhang, X., Lu, Y. and She, J. (2021), "A reliable data compression scheme in sensor-cloud systems based on edge computing", *IEEE Access*, Vol. 9, pp. 49007-49015, doi: 10.1109/ACCESS.2021.3068753.

Maudet, S., Andrieux, G., Chevillon, R. and Diouris, J. (2021), "Refined node energy consumption modeling in a LoRaWAN network", *Sensors*, Vol. 21 No. 19, p. 6398, doi: 10.3390/s21196398.

Morin, E., Maman, M., Guizzetti, R. and Duda, A. (2017), "Comparison of the device lifetime in wireless networks for the internet of things", *IEEE Access*, Vol. 5, pp. 7097-7114, doi: 10.1109/ACCESS.2017.2688279.

Parker, D., Stojanovic, M. and Yu, C. (2013), "Exploiting temporal and spatial correlation in wireless sensor networks", *2013 Asilomar Conference on Signals, Systems and Computers*, pp. 442-446, doi: 10.1109/ACSSC.2013.6810315.

Prauzek, M., Konecny, J., Borova, M., Janosova, K., Hlavica, J. and Musilek, P. (2018), "Energy harvesting sources, storage devices and system topologies for environmental wireless sensor networks: a review", *Sensors*, Vol. 18 No. 8, p. 2446, doi: 10.3390/s18082446.

Rehman, A., Saba, T., Kashif, M., Fati, S.M., Bahaj, S.A. and Chaudhry, H. (2022), "A revisit of internet of things technologies for monitoring and control strategies in smart agriculture", *Agronomy*, Vol. 12 No. 1, p. 127, doi: 10.3390/agronomy12010127.

Săcăleanu, D.I., Popescu, R., Manciu, I.P. and Perişoară, L.A. (2018), "Data compression in wireless sensor nodes with LoRa", *2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2018*, pp. 1-4, doi: 10.1109/ECAI.2018.8679003.

Sarbishei, O. (2019), "Refined lightweight temporal compression for energy-efficient sensor data streaming", *IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland*, pp. 550-553, doi: 10.1109/WF-IoT.2019.8767351.

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

Ullo, S.L. and Sinha, G.R. (2020), "Advances in smart environment monitoring systems using IoT and sensors", *Sensors*, Vol. 20 No. 11, p. 3113, doi: 10.3390/s20113113.

Väänänen, O. and Hämäläinen, T. (2019), "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*.

Väänänen, O. and Hämäläinen, T. (2020), "Sensor data stream on-line compression with linearity-based methods", *IEEE International Conference on Smart Computing (SMARTCOMP), Bologna, Italy*, pp. 220-225, doi: 10.1109/SMARTCOMP50058.2020.00049.

Väänänen, O. and Hämäläinen, T. (2021), "LoRa-based sensor node energy consumption with data compression", *2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*, pp. 6-11, doi: 10.1109/MetroInd4.0IoT51437.2021.9488434.

Further reading

Arduino MKR WAN 1310 (2022), available at: https://store.arduino.cc/mkr-wan-1310 (accessed 6 October 2021).

Digita (2022), "IoT LoRaWAN network coverage in Finland", available at: www.digita.fi/en/iot-lorawan-network-coverage-map/ (accessed 6 October 2021).

DHT22 temperature and humidity sensor (2022), available at: www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf (accessed 6 October 2021).

LoRa Alliance (2022a), "What is LoRaWAN specification", available at: https://lora-alliance.org/about-lorawan/ (accessed 6 October 2021).

LoRa Alliance (2022b), "RP002-1.0.0 LoRaWAN regional parameters", available at: https://lora-alliance.org/wp-content/uploads/2019/11/rp_2-1.0.0_final_release.pdf (accessed 6 October 2021).

Semtech, (2022) "What are LoRa and loRaWAN", available at: https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan (accessed 6 October 2021).

Semtech, (2022) "What is an adaptive data rate", available at: https://lora-developers.semtech.com/documentation/tech-papers-and-guides/understanding-adr/ (accessed 6 October 2021).

Corresponding author

Olli Väänänen can be contacted at: olli.vaananen@jamk.fi