Date of publication MAR-30, 2022, date of current version JAN-14, 2022.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.21.1.2523

A Method of IoT Information Compression

YURIY S. MANZHOS, YEVHENIIA V. SOKOLOVA
Software Engineering and Business Faculty, National Aerospace University – Kharkiv Aviation Institute,
Chkalova str., 17, Kharkiv, Ukraine (e-mail: y.manzhos@khai.edu, y.sokolova@khai.edu),
http://faculty6.khai.edu/uk/site/kafedra-inzheneriyi-progr.html

Corresponding author: Yuriy S. Manzhos (e-mail: y.manzhos@khai.edu).

This research is supported by the project STARC (Methodology of **S**us**TA**inable Development and Info**R**mation Technologies of Green **C**omputing and Communication) funded by the Department of Education and Science of Ukraine.

ABSTRACT The Internet of Things (IoT) is a modern paradigm that consists of heterogeneous intercommunicated devices that send and receive messages in various formats through different protocols. Thanks to the smart things mainstream, it is becoming common to collect large quantities of data generated by resource-constrained, distributed devices at one or more servers. However, the wireless transmitting of data is very expensive. For example, in IoT, Bluetooth Low Energy using costs tens of millijoules per connection, while computing at full energy costs only tens of micrjoules, and sitting idle costs close to 1 microjoules per second for STM processors. We need compression of data on smart devices. We introduce an IoT compression method based on the concurrent Cosine and Chebyshev Discrete Transforms. For performance increasing, the modification of Transforms algorithms is proposed. This method is suitable not only for IoT devices collecting data but also for the big servers.

KEYWORDS data approximation; Internet of Things; general orthogonal polynomials; lossy signal compression; modification of Chebyshev discrete transformation.

I. INTRODUCTION

THE Internet of Things (IoT) is a modern paradigm that consists of heterogeneous intercommunicated devices that send and receive messages in various formats through different protocols to achieve different goals [1]. IoT has more than 20 billion devices with a unique identifier that can interoperate via existing Internet infrastructure [2]. These devices can be used in different regions from the inside the human body to deep inside of the oceans and underground.

This heterogeneity in devices brings management challenges in architectural and protocol issues [3], that requires a network, embedded, and distributed programming knowledge.

Today IoT is a union of smart things, i.e., different electronic devices with embedded computers, sensors, actuators, and connectivity that enables these devices to connect and exchange data [1]. Each device has a unique address and can interoperate within the Internet infrastructure.

Thanks to the proliferation of smartphones, wearables, autonomous vehicles, and other connected devices, it is becoming common to collect large quantities of sensor-generated data [4, 5]. Often this data is collected from distributed, resource-constrained devices and centralized at servers [6, 7].

Unfortunately, wireless transmitting data is extremely power expensive. For example [8, 9], transmitting data over Bluetooth Low Energy costs tens of milliJoules, while computing at full power costs only tens of microJoules, and sitting idle costs close to 1 microJoules per second. STM32L5 series is Ultra-low-power microcontroller unit (MCU) and supports up to 125°C. The STM32L5 is the solution and provides a new optimal balance between performance, power and security that is very important for applications in IoT, medical, industrial and consumer sectors. For example, best power consumption of MCU consists of: 33 nA in shutdown mode; 3.6 µA in stop mode with full SRAM and peripheral states retention with 5µs wake-up time; down to current efficiency = 60 μA/MHz in active mode [10].

Researching the features of IoT device data [11] allows concluding that a natural solution is to compress the data on smart devices [12].

Unfortunately, existing compression methods either 1) are only applicable for specific types of data, such as timestamps [13, 14], audio [15] or medical data [16] recordings; or 2) use algorithms that are ill-suited to sensor-generated data [17].

Digitized signals in IoT replace conventional analogue signals. The transformation is based on the Nyquist–Shannon sampling theorem [18]. According to this theorem, for frequency of 8000 Hz and 8 bit per sample we need bitrate 64 Kb per second for voice information saving. That is why a modern IoT system requires storage and transmission of large quantities of data. Due to storage capacity and transmission speed constraints, it is necessary to reduce the quantities of stored data. Therefore, efficient data compression in IoT devices data is significant. Signal compression aims to achieve a high compression ratio while keeping the relevant information in the compressed signal.

Generally, compression of the signal can be realized as lossless and lossy. Lossless compression allows exact reconstruction of the source signal; however, a compression ratio is limited. The lossy compression allows to achieve a high compression ratio within some error between the source and reconstructed signals.

In this article, we mainly focus on IoT devices signal lossy compression [19] through Fourier series and Chebyshev polynomials [20]. Concurrent using Fourier and Chebyshev transformation requires many resources.

For performance increasing, the modification of Chebyshev and Cosine discrete transformations is proposed.

The article is organized as follows. In Section 2 we describe the features of discrete data transformations, briefly introduce Cosine and Chebyshev orthogonal polynomials transformations [20] and describe our proposed technique. Results are shown in Section 3. Finally, we draw a conclusion in Section 4.

II. MATERIAL AND METHODS

A. COMPRESSION ALGORITHMS

The amount of data needed to describe IoT devices signals is transmitted very slowly. Also, storage of IoT devices is costly. The information contained in the signals, therefore, must be compressed because signal compression means reducing the amount of data needed to present IoT device signals.

The source signal samples are partitioned into array of blocks. Every Block$_i$ transforms into TBlock$_i$. According to the defined threshold we clear some items in TBlock$_i$.

In the end we use Entropy coding (see Fig.1).

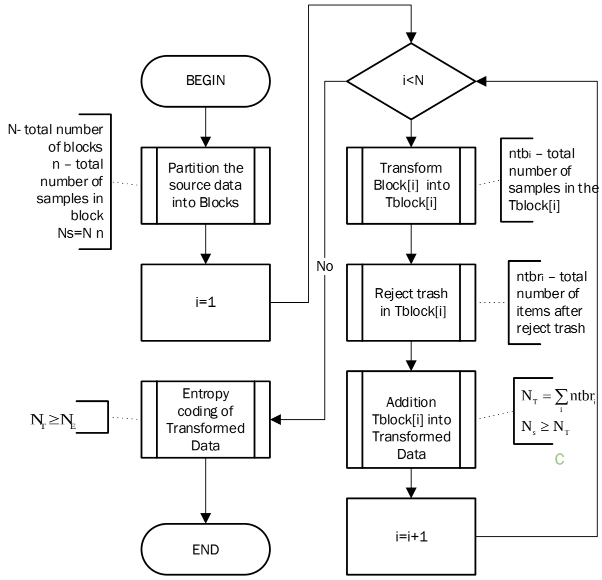

Figure 1. The block-diagram of the Transformation data of signal

Discrete unitary transformations are useful in signal processing applications that include signal restoration and data compression [20]. Examples of popular unitary transforms are the discrete cosine transform and the discrete Chebyshev transformations.

There are types of signals for which one transformation is compressed better than another. Below we present a brief review of some composed compression method of the signal based on the Cosine and Chebyshev transformations.

For even function $s(t)$ Cosine approximation is:

$$\hat{s}_f(t) = \frac{c_0}{2} + \sum_{n=1}^{\infty} c_n \cos\left(\frac{\pi n t}{L}\right), \tag{1}$$

where $2L$ is a period of $s(t)$ and

$$c_n = \frac{1}{\pi}\int_{-\pi}^{\pi} s(t)\cos(nt)dt\,. \tag{2}$$

Expressions (1, 2) are inverse and forward transformation for a signal $s(t)$. Forward transformation is based on the (2) often used for lossy data compression.

For IoT sensors we have discrete signals – time series of samples. In the most of cases IoT sampling is selected at equidistant points that is why we can use a Discrete cosine Transformation (DCT) [21] for approximation and compression of source data (see Fig.2). Expressions (3, 4) are forward and inverse discrete cosine transformation for samples of a signal $s(t)$.

$$c_k = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} s_n \cos\left(\frac{\pi(n+0.5)}{N} k\right), \qquad (3)$$

$$\hat{s}_k = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} c_n \cos\left(\frac{\pi(n+0.5)}{N} k\right). \qquad (4)$$

The Complexity of the algorithm is $O(n^2 \cos(x))$. That is why we need to improve this algorithm for using in the IoT systems.

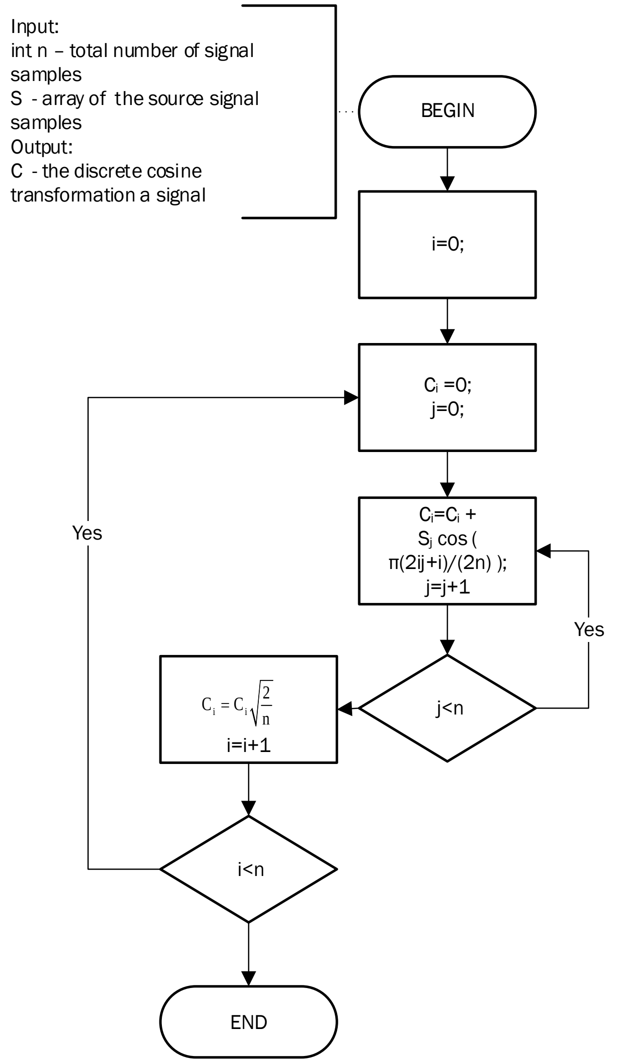

Figure 2. The block-diagram of the DCT

Based on this algorithm two algorithms were elaborated: Part1 – initialization of DCT and Part2 – calculation of DCT (see Fig. 3)

Part1 calculates the temporal two dimensional array of cosines as $Z_{ij} = \cos\left(\frac{\pi(j+0.5)}{n} i\right)$.

Part2 calculates the DCT with the help of Z_{ij}. Chebyshev polynomials $T_n(t)$ of the first kind is defined as [21]:

$$T_0(t) = 1;\; T_1(t) = t;\; T_2(t) = 2t^2 - 1 \ldots \qquad (5)$$
$$T_{n+1}(t) = 2tT_n(t) - 1 \quad n \geq 1.$$

The polynomial $T_n(t)$ has k zeros in the interval [−1, 1], and they are located at the points:

$$t_k = \cos\left(\frac{\pi(k-0.5)}{n}\right). \qquad (6)$$

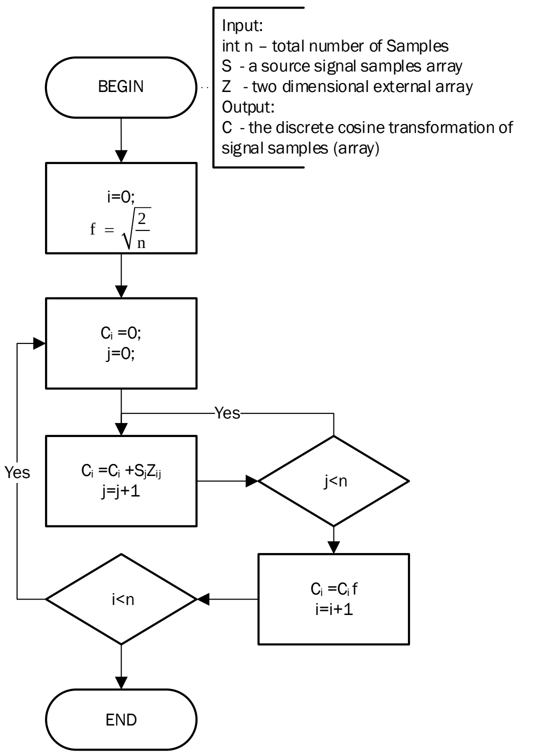

Figure 3. The block-diagram of the optimized DCT

Also $T_n(t)$ satisfies a discrete orthogonality relation: if i, j < m, then

$$\sum_{k=1}^{n} T_i(t_k) T_j(t_k) = \begin{cases} 0 & i \neq j \\ \frac{m}{2} & i = j \neq 0 \\ m & i = j = 0 \end{cases}. \qquad (7)$$

If $s(t)$ is an arbitrary function in the interval [−1, 1], and if n coefficients cj, j = 0,...,n − 1, are defined by

$$c_j = \frac{1}{n}\sum_{k=1}^{n} s(t_k)T_j(t_k),$$

$$c_j = \frac{1}{n}\sum_{k=1}^{n} s\left(\cos\left(\frac{\pi(k-0.5)}{n}\right)\right)\cos\left(\frac{\pi j(k-0.5)}{n}\right), \quad (8)$$

then the approximation formula of Chebyshev approximation of $s(t)$ is defined as:

$$\hat{s}_c(t) = c_0 + \sum_{n=1}^{\infty} c_n T_n(t) \quad -1 < t < 1. \quad (9)$$

Expressions (8, 9) are forward and inverse Discrete Chebyshev transformations for samples of a signal $s(t)$. The main feature of Chebyshev polynomials is a minimal error for $-1 < t < 1$.

There is a Discrete Chebyshev Transformation (DChT) [20] that is shown in Fig. 4. The Complexity of the algorithm is $O(n^2\cos(x))$.

In order to improve performance, an optimization of the algorithm is proposed.

Since a real signal of IoT is an array of samples in equidistant points we can not use the expression (7). That is why the block "Initialization" uses "Interpolation" for calculation of samples between points.

Also, the mapping time interval of block samples to [-1; 1] was used in our algorithm.

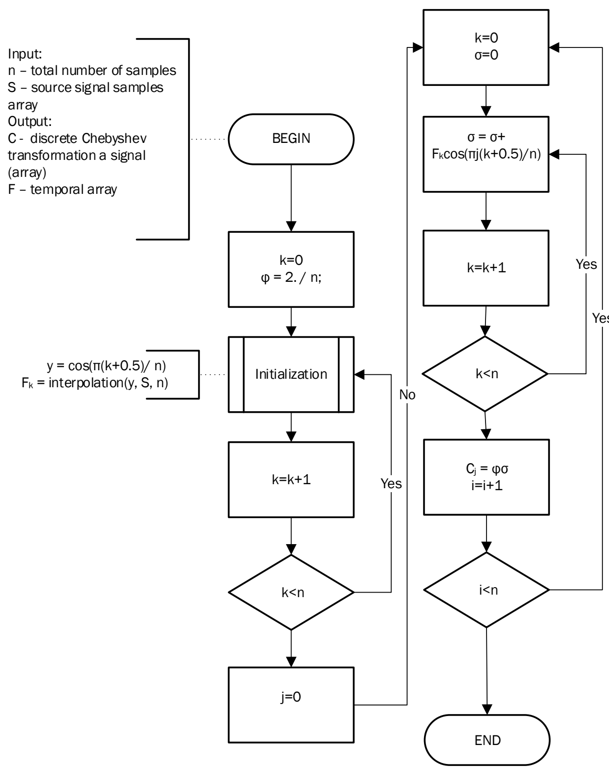

Figure 4. The block-diagram of the DChT

Based on this algorithm two algorithms were elaborated: Part1 – initialization of DChT and Part2 – calculation of DCT (see Fig.5).

Part1 calculates:

temporal array for zeros of $T_n(t)$ as:

$$X_k = \cos\left(\frac{\pi(k+0.5)}{n}\right),$$

temporal two dimensional array of cosines as

$$Z_{jk} = \cos\left(\frac{\pi j(k+0.5)}{n}\right).$$

Part2 calculates the DCT with the help of Z_{jk} and X_k.

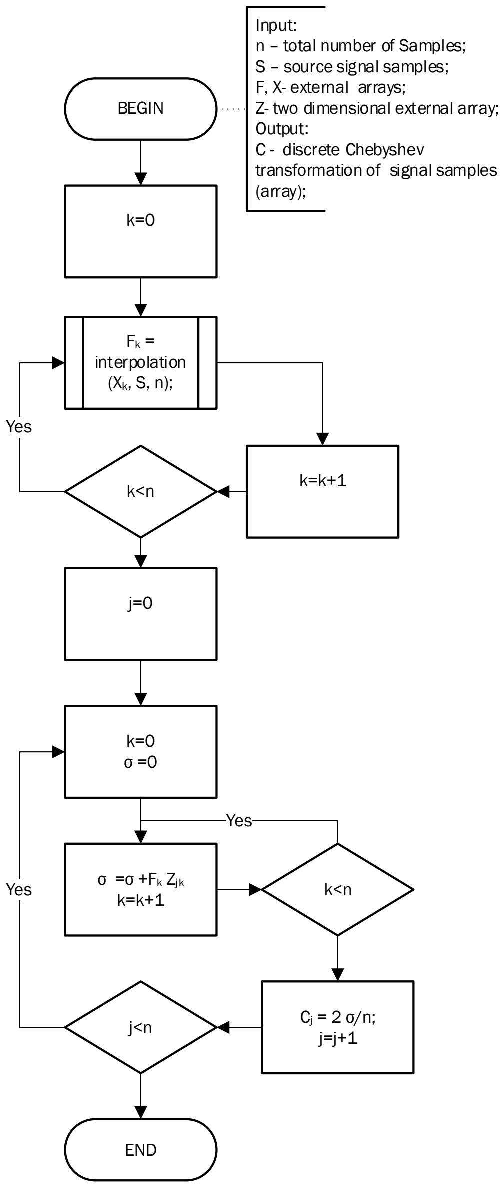

Figure 5. The block-diagram of the optimized DChT

Optimized algorithms were used for the elaboration of the composed compressed technique based on the concurrent transformation of samples $s(t_i)$. After the transformation we have an approximated signal $\hat{s}(t_i)$.

Let n be the total number of the IoT samples. In most compression algorithms, there are various performance metrics, for example:

The Signal Relative Maximum Error (S_{RME}) evaluated as:

$$S_{RME} = \frac{\max\limits_i |s(t_i) - \hat{s}(t_i)|}{\max\limits_i |s(t_i)|}. \qquad (10)$$

Current Signal Relative Error (S_{RE}):

$$S_{REi} = \frac{|s(t_i) - \hat{s}(t_i)|}{\max\limits_i |s(t_i)|}. \qquad (11)$$

We use Eq (10,11) for minimization of errors in the restored data.

For compression current relative error of transformation was used:

$$C_{REi} = \frac{|C_i|}{\max\limits_i |C_i|}. \qquad (12)$$

C_{REi} allows eliminating small items in the compressed data.

Compression Ratio (CR) is defined as a ratio between the number of samples needed to represent the original and the number of items in compressed data.

$$CR = \frac{n}{p}, \qquad (13)$$

where p is the total number of C_i for $C_{REi} > \delta$, where δ is the threshold level (0.001..0.01).

In order to achieve the optimal CR, the functional architecture, based on DCT and DChT, was elaborated (see Fig. 6).

Each of these transformations has some peculiarities of use, but the simultaneous use of these transformations allows us to achieve an increase in CR.

In the first step in order to achieve the CR MCU calculates DCT and DChT for minimal bock data size (for example size =5 samples). The maximal size of data is determined by the highest order of the used generalized polynomials (maximal performance of MCU and memory size), for example, maximal size =64. After this operation, we have two sets of approximation coefficients A_i, C_i.

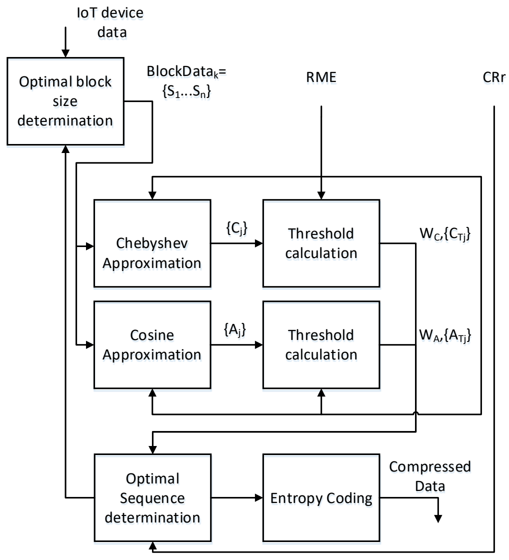

Figure 6. Functional architecture of adaptive compression of the IoT devices signal

For the creation of Chebyshev and Cosine approximations, we propose unique algorithms.

Algorithm ChebApprox:

inputs: n – a number of samples $s(t_i)$ in the block data; ε = $10^{-5..-2}$ requires a Relative Maximum Error (RME) of approximation;

output: $\{C_i\}$ – vector of Chebyshev approximation.

BEGIN algorithm:

1. Fix the order m=n of the Chebyshev approximation.
2. Transform the input time series and find the C_i using interpolation of source data.
3. Construct the approximated function $\hat{s}(t_i)$ using Eq. (9).
5. Calculate RME using Eq. (10).
6. If RME < ε set m = m - 1 and go to 2

END algorithm ChebApprox

Algorithm CosineApprox:

Inputs: n – a number of sample $s(t_i)$ in the block data; [a; b]- interval of approximation, $\varepsilon = 10^{-5..-2}$ requires an error of approximation – RME

Outputs: $\{A_i\}$ -vector of Cosine approximation

BEGIN algorithm:

1. Fix the order m=n of the Cosine approximation.
2. Transform the input time series into the A_i.
3. Construct the approximated function $\hat{s}(t_i)$ using Eq. (4).
5. Calculate RME using Eq. (10).
6. If RME > ε set m = m + 1 and go to 2
7. If RME < ε set m = m-1 and go to 2;

END algorithm CosineApprox

In the second step of MCU, according to the threshold level, δ calculates two subsets of coefficients $\{A_{Ti}\}$, $\{C_{Ti}\}$ and controls words W_A and W_c (see Eq.14):

$$C_i = \begin{cases} 0, & |C_i| < \delta \\ C_i, & |C_i| \geq \delta \end{cases} \quad , \quad A_i = \begin{cases} 0, & |A_i| < \delta \\ A_i, & |A_i| \geq \delta \end{cases}. \tag{14}$$

The number of items in these sets is less than a number of items in the source sets.

According to Eq(13) end we define current CR.

The valid bits in control words define the position of nonzero items of the source sets – the first bit in control word equals true for DCT and false for DChT.

If current $CR \leq CR_r$ we set the block data as size =size + 2. Also, if the new size is less than the maximal size, we go to the First step. After repeating, we have the optimal CR for this block of data.

In the third step, we compress Data via Entropy Coding. So we can send the compressed data to other IoT devices.

The operation for step One-Three is repeated for the complete set of IoT raw data.

Proposed algorithms allow eliminating the information redundancy in digital streams at the output of IoT devices.

B. EFFECTIVENESS OF COMPRESSION

Now we need energy and time effectiveness of compression for IoT.

Let η_E be the energy effectiveness:

$$\eta_E = \frac{E_{TS}}{E_{TC} + E_C}, \tag{15}$$

where E_{TS} is an energy for source data transmitting via wireless communication; E_C is an energy for data compressing; E_{TC} is the energy for compressed data transmitting measured in Joules.

In our case:

$$E_{TS} = P_T \tau_{TS}\,,\; E_C = P_C \tau_C\,,\; E_{TC} = P_T \tau_{TC}\,, \tag{16}$$

where P_T, P_C are power consumptions for transmiting/compressing of data [W], τ_{TS}, τ_C are time intervals of transmiting/compressing of source data, τ_{TC} is a time interval for compressed data [sec] transmiting.

For compressing data define

$$\alpha = \frac{V_S}{V_C}, \tag{17}$$

where V_S, V_C are source/compressed volumes of data in bits.

For defined data rate β [bit/sec]

$$\tau_{TS} = \frac{V_S}{\beta}, \tau_{TC} = \frac{V_C}{\beta}, \tag{18}$$

$$\eta_E = \frac{P_T \tau_{TS}}{P_T \tau_{TC} + P_C \tau_C} = \frac{P_T \dfrac{V_S}{\beta}}{P_T \dfrac{V_C}{\beta} + P_C \tau_C},$$

$$\eta_E = \frac{V_S}{V_C + \dfrac{P_C}{P_T} \beta \tau_C} = \frac{\alpha}{1 + \dfrac{P_C}{P_T} \beta \dfrac{\tau_C}{V_C}}, \tag{19}$$

where $\tau_C = \dfrac{N_{iC}}{\Pi}$, where Π is a processor performance, measured in instruction/sec; N_{iC} is the total number of instructions for source data compression.

The volume of source information is defined as

$$V_S = N_S R_S\,, \tag{20}$$

where N_S is the total number of samples, R_S is the total number bit per samples.

For block of samples:

$$N_S = N_B L_{BS}\,, \tag{21}$$

where N_B is the total number of blocks, L_{BS} is the length of data block in samples.

Since discreate cosine or Chebyshev transformation have asymptotic time complexity of $O(n^2)$ then:

$$N_{iC} = N_B L_{BS}^2 \gamma\,, \tag{22}$$

where γ is the coefficient of proportionality (total number of instructions per elementar block), defined by program in [instruction]. For cosine transformation $\gamma \approx 30$ [ins] for Chebyshev transformation $\gamma \approx 10$ [ins]

$$\text{or } N_{iC} = \frac{V_S}{R_S L_{BS}} L_{BS}^2 \gamma = \frac{V_S L_{BS}}{R_S} \gamma,$$

$$\tau_C = \frac{N_{iC}}{\Pi} = \frac{V_S L_{BS}}{\Pi R_S} \gamma,$$

$$\eta_E = \frac{\alpha}{1 + \dfrac{P_C}{P_T} \beta \dfrac{V_S L_{BS}}{\Pi R_S V_C} \gamma} = \frac{\alpha}{1 + \dfrac{P_C}{P_T} \dfrac{\alpha \beta L_{BS}}{\Pi R_S} \gamma}. \tag{23}$$

In our case α =10…13 for block size=20..30.

Power consumption for Bluetooth unit is P_T =100 mW. Data rate is β=125000 bit/sec [22]. For STM32L5 the current consumption 16 mA (3.3 V) P_C =50 mW, the performance Π =165 DMIPS [23]. Total number bit per samples R_S =16 bit. The length of data block L_{BS} =10.

$$\eta_E = \frac{\alpha}{1 + \frac{P_C}{P_T} \frac{\alpha \beta L_{BS}}{\Pi R_S} \gamma} .$$

Even with $\gamma \approx 50$ [ins] the energy effectiveness is:

$$\eta_E = \frac{\alpha}{1 + \frac{50[mW]}{100[mW]} \frac{\alpha \cdot 125000 \left[\frac{bit}{\sec}\right] \cdot 10}{1.65 \cdot 10^8 \left[\frac{ins}{\sec}\right] \cdot 16[bit]} 50[ins]} ,$$

$\eta_E \approx \alpha = 10..13$. The amount of energy for compressing and transmitting data is less then 10 % of amount of energy for transmitting of uncompressed data.

Time effectiveness 1 (it is important for security and reliability, because decreasing of transmitting time to increase the reliability and security level of IoT) is:

$$\eta_{T1} = \frac{\tau_{TS}}{\tau_{TC}} . \quad (24)$$

After substitution τ_{TS} and τ_{TC} (see Eq. 17)

$$\eta_{T1} \approx \alpha = 10..13 .$$

Time effectiveness 2 (it is important for evaluation of delay for real time systems of IoT):

$$\eta_{T2} = \frac{\tau_{TS}}{\tau_{TC} + \tau_C} . \quad (25)$$

After substitution τ_{TS} , τ_{TC} and τ_C (see Eq. 17)

$$\eta_{T2} \approx \alpha = 10..13 .$$

Compression allows increasing the life time, security and reliability of IoT to decrease the delay for transmitting of data in the real time systems.

III. RESULTS

After implementation in C++ the relative performance (RP) as the ratio between the time of execution for classic Transforms to time of execution for optimized Transforms for different input data was researched.

To avoid the influence of cache memory, every transformation was repeated 10000 times. The transformation time was calculated as an average time for every input signals. The result of this research is shown in Fig. 7.

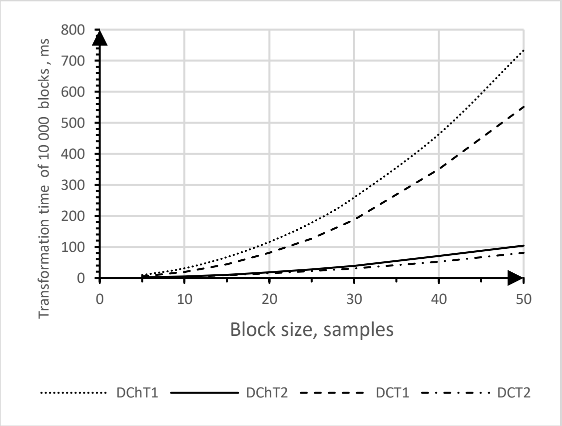

Figure 7. Performance of DCT and DChT

We observed the performance increase for optimized algorithms. As a result, we have performance increase of $5...9$ for such signals:

$$s_n(t) = t^n ,\ T_n(t),\ \sin(nt),\ t\sin(nt), (1-t)\sin(nt) .$$

The compression ratio (Eq 13) is defined by number of samples in block data and threshold. For test signals the CR=3..50. For real audio signal (450 kB) the CR=3...15 (see Fig. 8).

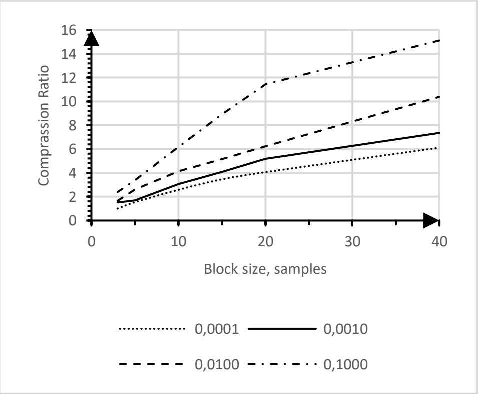

Figure 8. Compression ratio for audio signal and different thresholds

The CR for real audio signals is defined by ratio Discretization frequency to frequency band of signal. For small value of this ratio we have the small CR.

In the figures we can see the Compression ratio for different test signals and thresholds (see. Fig. 9, 11, 13, 14, 17):

CRC2, CRC5 – compression ratio for Cosine Transformation and treshholds 10^{-2} and 10^{-5};

CRT2, CRT5 – compression ratio for Tchebyshev Transformation and thresholds 10-2 and 10-5.

In the figures we can see the Relative Maximum Error for different test signals and thresholds: (see Fig. 10, 12, 15, 16, 17):

EC2, EC5 – compression ratio for Cosine Transformation and treshholds 10^{-2} and 10^{-5}.

ET2, ET5 – compression ratio for Tchebyshev Transformation and treshholds 10^{-2} and 10^{-5}.

Where $T_5(t) = 16t^5 - 20t^3 + 5t$.

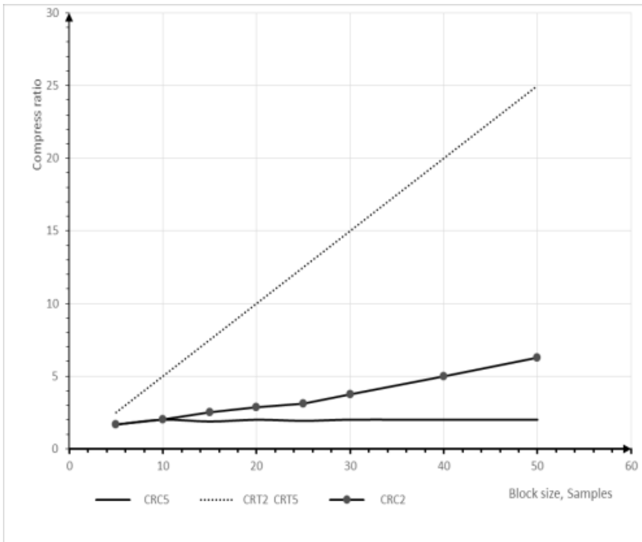

Figure 9. Compression ratio for test signal $s(t)=t^2$

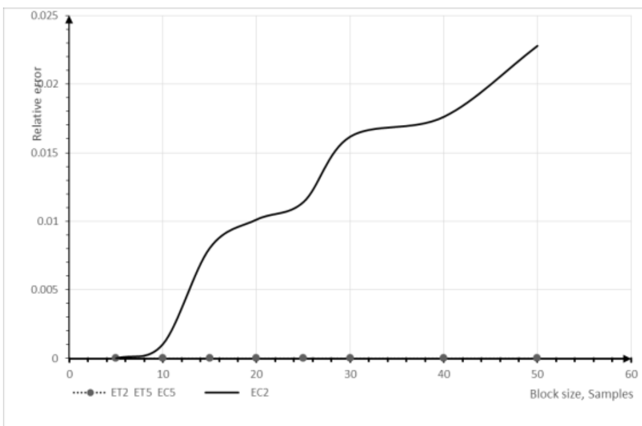

Figure 10. The Relative Maximum Error for test signal $s(t)=t^2$

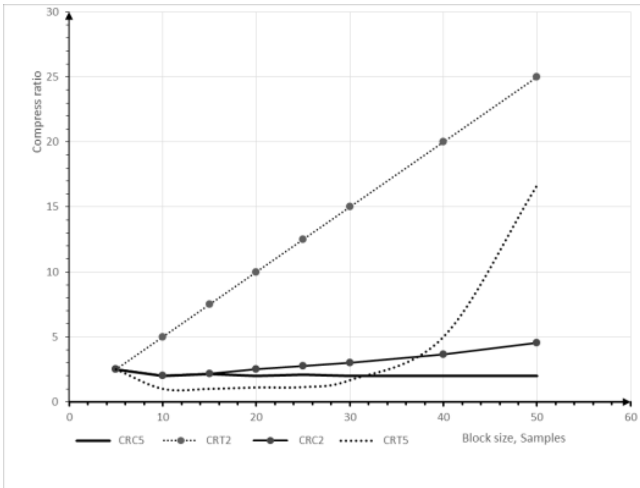

Figure 11. Compression ratio for test signal $s(t)=t^3$

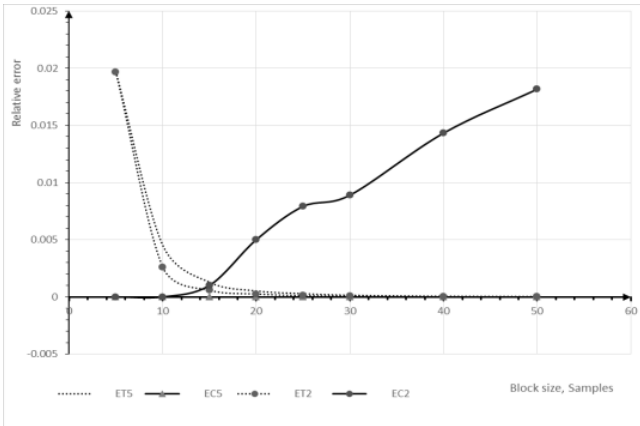

Figure 12. The Relative Maximum Error for test signal $s(t)=t^3$

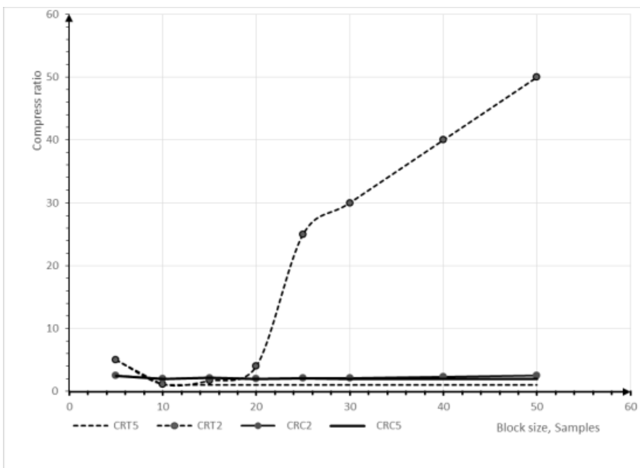

Figure 13. Compression ratio for test signal $s(t)=T_5(t)$

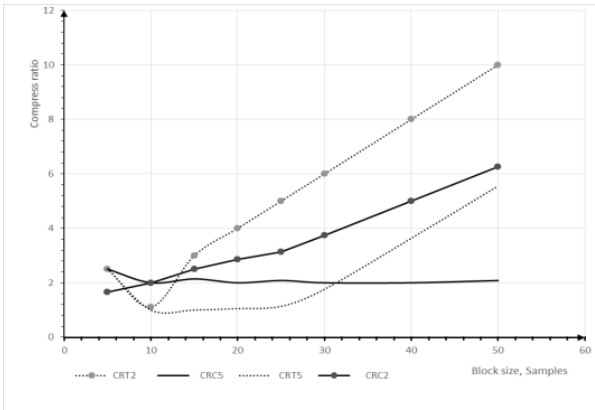

Figure 14. Compression ratio for test signal s(t)=sin(5t)

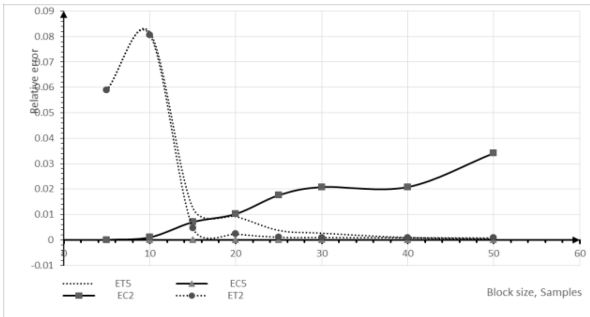

Figure 15. The Relative Maximum Error for test signal s(t)= sin(5t)

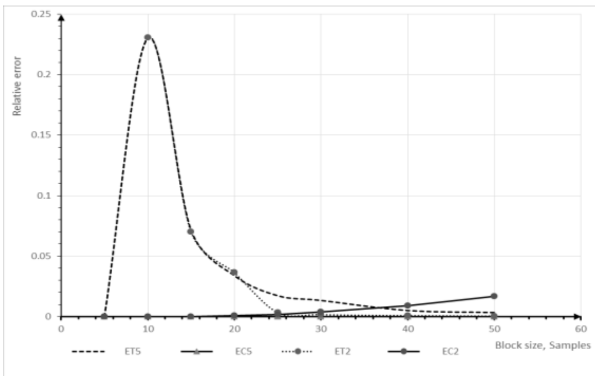

Figure 16. The Relative Maximum Error for test signal s(t)= $T^5(t)$

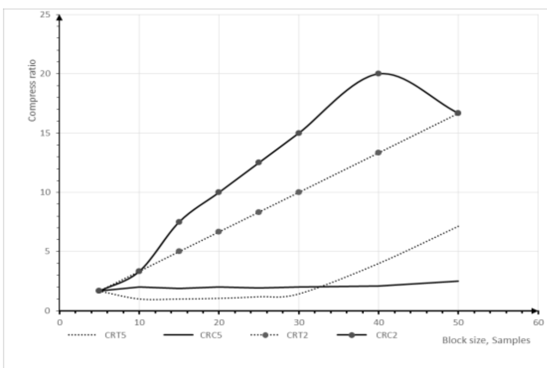

Figure 17. Compression ratio for test signal s(t)=tsin(2t)

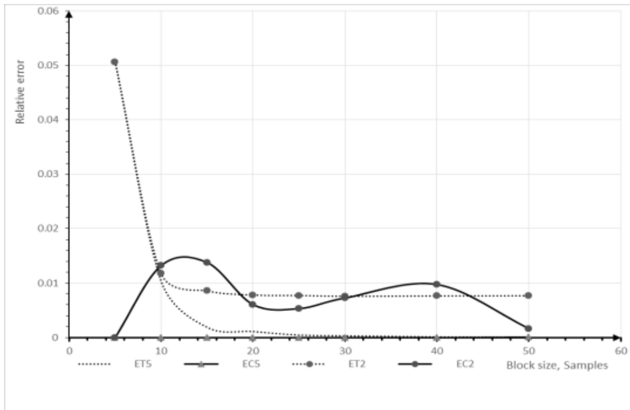

Figure 18. The Relative Maximum Error for test signal s(t)= tsin(2t)

In the most cases (See Fig.9-17) the Chebyshev Compression Ratio > Cosine Compression rate. But for high Compression Ratio the concurrent compression was used.

In the following figures we can see the Restored signals after Tchebyshev Transformation () and after Cosine Transormation for treshholds 10^{-2} (see. Fig. 19, 20, 21).

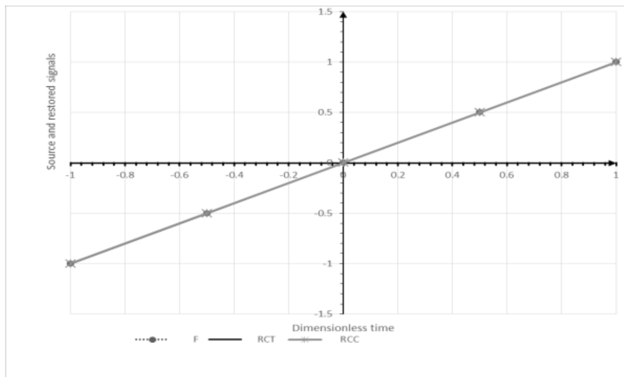

Figure 19. Restored signal for test signal s(t)=T5(t) and thresholds=0.01, block size =5

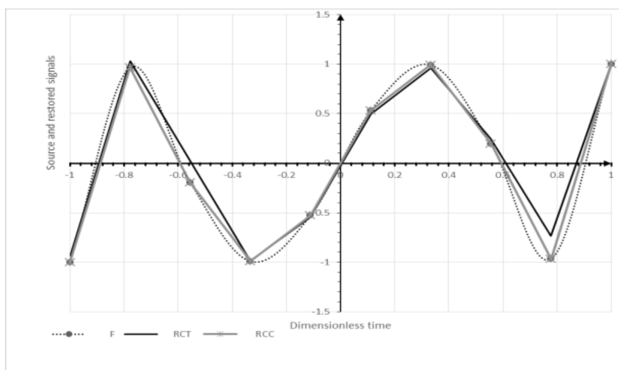

Figure 20. Restored signal for test signal s(t)=$T_5(t)$ and thresholds=0.01, block size =10

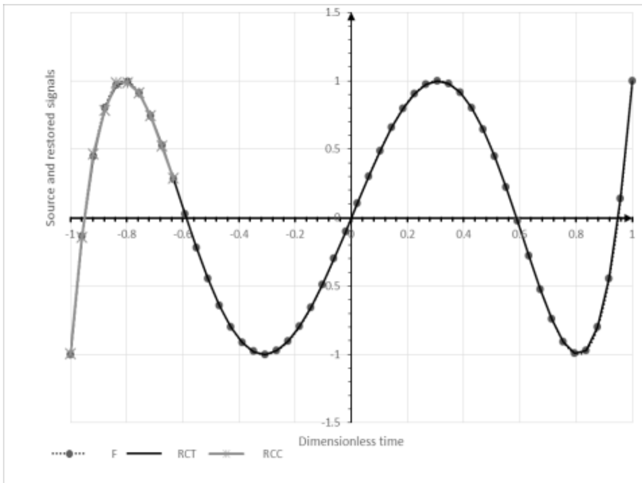

Figure 21. Restored signal for test signal s(t)=T_5(t) and thresholds=0.01, block size =50

For block size=5 we can see approximation of T5(t) as the line. Bad approximation is explained by using of square polynomial interpolation. But for block size>10 we can see the high quality approximation. We can say that block size must be > 10.

We can see a positive correlation between threshold, block size and compression ratio if block size is less then 50 (see Fig. 9, 11, 13, 14, 17).

The maximum relative error is calculated according to Eq(10) for every block and every samples. The error is defined by total number of samples in the block and threshold (see Fig. 10, 12, 16, 15, 18).

According to Fig. 8-10 we have the following optimal values: Block Size 20..30, Threshold 10^{-2}, Maximum Relative Error of block 0…0.02 and Compression Ratio 6..9. But real compression ratio and maximum relative error of block is defined by IoT signal.

IV. CONCLUSIONS

In this article, we have proposed a new IoT device signal compression technique based on using optimized discrete cosine and Chebyshev transformations that provides the high performance.

Due to the concurrent usage of two different transformations, the proposed technique provides better compression ratio, energy consumption and time effectivenes.

The technique can be used to transmit the time series of data, e.g., the sound in smart homes or the environment; in the healthcare (e.g., arterial pressure, etc.); in industry (e.g., different technical parameters, etc.) because in most cases the precision of IoT sensors is 8..16 bit per values but the precision of restored signal is 10..16 bit.

We continue to work with data of different IoT devices using other relevant lossy compression techniques to improve the efficiency of our proposed scheme.

References

[1] L. S. Bai, R. P. Dick, and P. A. Dinda, "Archetype-based design: Sensor network programming for application experts, not just programming experts," in *Proceedings of the 2009 IEEE International Conference on Information Processing in Sensor Networks*, 2009, pp. 85–96.

[2] *Internet of Things – number of connected devices worldwide 2015–2025*, [Online]. Available at: http://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide

[3] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of Things: vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, issue 7, pp. 1497–1516, 2012. https://doi.org/10.1016/j.adhoc.2012.02.016.

[4] M. Buevich, A. Wright, R. Sargent, and A. Rowe, "Respawn: A distributed multi-resolution time-series datastore," *Proceedings of the 2013 IEEE 34th Real-Time Systems Symposium (RTSS)*, 2013, pp. 288–297. https://doi.org/10.1109/RTSS.2013.36.

[5] D. Blalock, S. Madden, J. Guttag, "Sprintz: Time series compression for the Internet of Things," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 3, article 93, pp. 93:1-93:23, 2018. https://doi.org/10.1145/3264903.

[6] S. Rhea, E. Wang, E. Wong, E. Atkins, and N. Storer., "Littletable: a time-series database and its uses," *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 125–138. https://doi.org/10.1145/3035918.3056102.

[7] I. Sokolov, I. Turkin, "Resource efficient data warehouse optimization," *Proceedings of the 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Kyiv, 2018, pp. 491-495. https://doi.org/10.1109/DESSERT.2018.8409183.

[8] Texas Instruments, 2.4-GHz Bluetooth low energy system-on-chip, 2013. [Online]. Available at: http://www.ti.com/lit/ds/symlink/cc2540.pdf

[9] Texas Instruments, Cc2640 simplelink bluetooth wireless mcu, 2016. [Online]. Available at: http://www.ti.com/lit/ds/swrs176b/swrs176b.pdf

[10] STM32L562QE, 2019. [Online]. Available at: http://www.st.com/en/microcontrollers-microprocessors/stm32l562qe.html

[11] T. Bose, S. Bandyopadhyay, S. Kumar, A. Bhattacharyya, and A. Pal, "Signal characteristics on sensor data compression in IoT – An investigation," *Proceedings of the 2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2016, pp. 1–6. https://doi.org/10.1109/SAHCN.2016.7733016.

[12] A. Ukil, S. Bandyopadhyay, and A. Pal. "IoT data compression: Sensor-agnostic approach," *Proceedings of the IEEE Data Compression Conference (DCC)*, 2015, pp. 303–312. https://doi.org/10.1109/DCC.2015.66.

[13] M. P. Andersen and D. E. Culler, "Btrdb: Optimizing storage system design for timeseries processing," *Proceedings of the FAST*, 2016, pp. 39–52.

[14] T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza, and K. Veeraraghavan, "Gorilla: A fast, scalable, in-memory time series database", *Proceedings of the VLDB Endowment*, vol. 8, issue 12, pp. 1816–1827, 2015. https://doi.org/10.14778/2824032.2824078.

[15] Information technology – generic coding of moving pictures and associated audio information – part 7: Advanced audio coding (aac), 2006. [Online]. Available at: https://www.iso.org/standard/43345.html

[16] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A. P. Chandrakasan, "A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system," *IEEE Journal of Solid-State Circuits*, vol. 45, issue 4, pp. 804–816, 2010. https://doi.org/10.1109/JSSC.2010.2042245.

[17] N. Q. V. Hung, H. Jeung, and K. Aberer, "An evaluation of model-based approaches to sensor data compression," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, issue 11, pp. 2434–2447, 2013. https://doi.org/10.1109/TKDE.2012.237.

[18] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10-21, 1949. https://doi.org/10.1109/JRPROC.1949.232969.

[19] J. F. De Castro Mota, E. Zimos, M. Rodrigues, N. Deligiannis, "Internet-of-things data aggregation using compressed sensing with side information," *Proceedings of the 23rd International Conference on Telecommunications (ICT)*, Thessaloniki, Greece, May 16-18, 2016, pp. 402-406.

[20] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3nd ed., Cambridge University Press, 2007, 1262 p.

[21] K. R. Rao, P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications,* Academic Press, Boston, 1990, 490 p. https://doi.org/10.1016/B978-0-08-092534-9.50007-2.

[22] K. Pothuganti, A. Chitneni, *A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi*, 2014, [Online]. Available at: https://www.iso.org/standard/43345.html

[23] *STM32L5 Series microcontroller ultra-low-power features overview*, 2020, [Online]. Available at: https://www.st.com/resource/en/application_note/an5213-stm32l5-series-microcontroller-ultralowpower-features-overview-stmicroelectronics.pdf.

Yuriy S. Manzhos, Candidate of Technical Sciences (PhD), Docent, Associate Professor at the Department of Software Engineering. Areas of scientific interests are Data Compression in Iot data, Model-checking, Optimization of Software Source Code, Software Invariants and Reliability.

Yevheniya V.Sokolova, Candidate of Technical Sciences (PhD), Docent, Associate Professor at the Department of Software Engineering. Areas of scientific interests are Adaptive Control, Cloud Computing, Data Compression in Iot data, Information Urgency and Reliability, Interval Arithmetic, Optimization of Software Testing, Testing of the Complex Technical Systems.