UDC 004.41
DOI: 10.15587/1729-4061.2024.298476

DEVELOPMENT OF THE METHOD OF DETECTING AND CORRECTING DATA TRANSMISSION ERRORS IN IOT SYSTEMS FOR MONITORING THE STATE OF OBJECTS

Vladyslav Sokolovskyi
Corresponding author
Postgraduate Student, Assistant*
E-mail: falcon.f2b@gmail.com
Eduard Zharikov
Doctor of Technical Sciences, Professor*
Sergii Telenyk
Doctor of Technical Sciences, Professor
Department of Automation and Computer Science
Cracow University of Technology
Warszawska str., 24, Krakow, Poland, 31-155
Department of Information Systems and Technologies
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Beresteiskyi ave., 37, Kyiv, Ukraine, 03056
*Department of Computer Science and Software Engineering
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Beresteiskyi ave., 37, Kyiv, Ukraine, 03056

The object of the study is the IoT system for monitoring the state of objects.

The problem being solved is the development of an innovative method of detecting and correcting data transmission errors in the networks of Internet of Things systems.

The essence of the results is that a method of detecting and correcting multiple transmission errors during byte-by-byte transmission of a block of information has been developed. The method is distinguished by an original coding scheme, which involves the calculation of control bits, as well as the shuffling of block bits by performing bit shift operations. The peculiarity of the method is that any bit can be distorted when transmitting a code word, but it belongs to different code combinations of the Hamming code. This allows multiple data transmission errors to be detected and corrected during decoding, and multiple errors of different bytes belonging to the same block can be corrected.

Simple algorithms for encoding and decoding procedures have been developed, and programs for information block encoding procedures and decoding procedures with error detection and correction have been developed. A software model of the data transmission channel was also developed with the possibility of introducing multiple errors when simulating the data transmission process. All programs are developed in Python, although other languages are possible.

An experiment was conducted using the developed software model of the data transmission channel. The efficiency of the developed method has been experimentally confirmed and it has been proven that its use increases the immunity of the data transmission channel. This is due to the fact that the developed method allows detecting and correcting all code word transmission errors with a multiplicity from 1 to 8, which was confirmed experimentally.

The main field of use of the developed method is considered to be IoT system networks. First of all, systems for monitoring the state of objects

Keywords: method of error correction, Hamming codes, Internet of Things, monitoring of the state of objects

Received date 30.11.2023
Accepted date 05.02.2024
Published date 28.02.2024

How to Cite: *Sokolovskyi, V., Zharikov, E., Telenyk, S. (2024). Development of the method of detecting and correcting data transmission errors in IoT systems for monitoring the state of objects. Eastern-European Journal of Enterprise Technologies, 1 (9 (127)), 22–33. doi: https://doi.org/10.15587/1729-4061.2024.298476*

1. Introduction

The rapid development of the high-speed Internet and the Internet of Things, in the future IoT [1], causes the need to build error-free information transmission channels.

One of the largest segments of the IoT is the Industrial Internet of Things, given the number of connected devices and the extent to which the technology is used for production, automation, and facility health monitoring purposes.

The high efficiency of IoT technology is due, among other things, to the use of the IoT platform as a service [PaaS] [2].

When developing industrial IoT networks, attention is traditionally paid to establishing a connection and the data transfer process. Without reliable data transmission technology, the IoT will be ineffective.

In industrial IoT networks, data transmission is based on short-range communication systems. Such systems usually do not follow the rules of the IP protocol.

But industrial networks, as a rule, are distributed over large areas and are affected by wide-spectrum electromagnetic interference. This can cause multiple data transmission errors. Therefore, the process of developing different methods for detecting and correcting data transmission errors in IoT is relevant because it can be used in various applications. There are many uses of Hamming codes in cases where transmission error detection and correction is required. But the main disadvantages of known applications are coding redundancy and low ability to detect multiple code word transmission errors. In addition, in the case of the usual use of Hamming codes, when it is necessary to ensure the transmission of a code word containing eight information bits, it is necessary to organize a data transmission channel with a code word length of 12 bits. With a large number of existing communication channels that provide the transmission of one-byte code words, the development of new methods for detecting and correcting errors in the transmission of information is in great demand. The development of new or modified methods based on the use of known Hamming codes is very relevant, considering the possible range of applications in networks with the above characteristics. Especially in IoT networks for monitoring the state of objects.

2. Literature review and problem statement

The possibility of error-free data transmission in the presence of noise was proven in [3]. The work is advantageously distinguished by the proposed innovative solution, which consists in the use of error-correcting codes. A disadvantage can be considered the fact that although the existence of such codes has been proven in the work, there is no explanation of how to construct the codes.

How to build error-correcting codes is described in [4]. The Hamming codes described in this work have become widely used. The advantage of these codes can be considered that they provide detection of double errors and detection and correction of single errors. Also, Hamming codes give good results in detecting and correcting errors, if there is a small probability of the occurrence of a "packet" of errors, that is, group interference. This advantageously distinguishes Hamming codes from the use of a parity bit. Because using the parity bit does not allow error correction, it can only detect an odd number of errors in the bits. The main drawback of the proposed codes is redundancy.

The use of Hamming codes is relevant today because there are applications in a wide range of areas where the reliability of data transmission is critical. The work [5] analyzed the use of different Hamming codes in data transmission. An error detection and correction method based on a parity check code is also presented together with a Hamming code algorithm to overcome the error correction problem of information transmission. The advantage of this approach can be considered its simplicity and efficiency in detecting and correcting data transmission errors. The disadvantage of the method is redundancy.

Work [6] continued the study of codes that have the properties of detecting and correcting errors. An algorithm for reducing the number of redundant bits when constructing an extended error detection and correction code is proposed. This was made possible by using the original polynomial generator. The advantage of the improved code is to reduce the overhead caused by interleaving redundant bits in a typical Hamming code and the length of redundant bits that exist in cyclic redundancy checks. The proposed code minimizes the payload bit overhead compared to the Hamming code, while reducing the resource usage of the designed memory architecture. A rather complex implementation of the proposed method and code redundancy can be considered a disadvantage of the proposed solution. The authors position these codes as alternative methods of error detection and correction. The algorithm did not solve all problems, but it can be used in many applications.

The introduction of codes with the possibility of detecting and correcting errors also occurs in the development of semiconductor RAM with random access. In [7], a variant of RAM construction in the form of a two-dimensional energy-efficient structure based on the use of error correction codes is proposed. The proposed method is based on the use of the Hamming code. The advantage of the proposed two-dimensional error correction code is to improve energy efficiency and reduce equipment cost. It is also proven that the method made it possible to reduce the area and energy consumption of memory within a certain amount of memory. The disadvantage of the two-dimensional error correction code is the redundancy and rather cumbersome procedure for detecting and correcting errors.

In [8], the introduction of information encoding algorithms during programming of field-programmable gate arrays (FPGAs) – semiconductor devices that can be configured by the manufacturer or developer after manufacturing – is considered. In this work, an improved error-detection correction code was implemented to ensure error-free transmission under packet-switched conditions, which overcomes the limitations associated with the use of cyclic redundancy check code and Hamming code. The advantage of the proposed algorithm is a higher transmission speed and an increase in the detection of random errors was achieved by reducing the number of redundant control bits. This is possible thanks to the proposed polynomial generator, as well as due to the reduction of speed losses from the interleaving of control bits in the usual Hamming code. The main drawback of the proposed method is a rather complex encoding and decoding algorithm with error detection and correction.

Work [9] is also devoted to the development of an improved reliable method of error detection and correction codes with minimum redundant bits and maximum coding speed. The advantage of the developed semi-diagonal code is greater efficiency compared to existing matrix codes. The use of a half-diagonal code reduces bit costs by at least 5.5 % and increases the encoding speed by at least 16.74 %. A disadvantage of the proposed code can be considered a cumbersome algorithm.

In [10], the problem of detecting and correcting errors when storing data in memory is considered. It is noted that technological scaling increases the density of memory bit cells and reduces the voltage of semiconductors. Therefore, the number of single radiation-induced errors and multiple errors is increasing. Error correction codes (ECC) have been proposed to solve this problem. The proposed encoding, based on the modified Hamming code described in the paper, guarantees that only correct values will be transmitted to the system output. In this case, the data is processed despite the presence of up to three-bit errors. The advantage of the proposed coding method is a high level of error correction, performance, high speed and low complexity. A typical disadvantage is coding redundancy.

The work [11] is interesting from the point of view of using interference-resistant coding. It is noted that reliable computer systems use error control codes (ECC) to protect information from transmission or storage errors. For example, memory is often protected with single-error-correcting and double-error-detecting codes. ECCs are traditionally designed to minimize the number of redundant bits as they are added to each word in the entire memory. However, using ECC causes encoding and decoding delays. The paper notes that in some applications, fast encoding and decoding is more important than redundancy. The paper summarizes previous work on ultrafast codes and proposes new codes that combine double error detection and adjacent error correction. The advantage of the work [11] can be considered the proposed synthesized high-speed codes that reduce the delay time for encoding and decoding. The disadvantages of synthesized codes are traditional: redundancy and delays in encoding and decoding.

In [12], it is proposed to expand the concept of error correction to the concept of error reduction. Several decoding methods are presented to improve the error reduction capabilities of Hamming codes. The error reduction properties of Hamming codes with standard decoding are studied. A lower bound on the average number of errors present in a decoded message when two errors are introduced by the channel for common Hamming codes is specified. Original decoding algorithms were investigated and found to improve the error reduction capabilities of Hamming codes beyond the aforementioned lower bound of standard decoding. The advantage is that by using a modification of known coding methods with the possibility of detecting and correcting errors, it is possible to reduce the number of errors compared to standard methods. The disadvantages of the proposed methods are that they are characterized by the same defects as redundancy, cumbersomeness and loss of coding time.

Comprehensive information on the use of error control codes is provided in [13]. Both classical block and lattice codes are considered. Recent developments in iterative codes such as turbo codes and low-density parity check codes are explored. Practical algorithms are presented. It is possible to consider the advantages of the work to be described in detail, built on a solid, carefully developed mathematical basis, algorithms for methods of detecting and correcting errors. The disadvantages include the fact that the practical aspects of implementing certain decoding algorithms on real equipment are not sufficiently disclosed.

An interesting review of coding schemes is made in [14]. Error correction schemes (ECS) contribute significantly to the reliability and energy efficiency of wireless sensor networks (WSNs). An overview of the different types of ECS used in communication systems and a brief description of standards for WSNs are given. Future research tasks on the development and implementation of ECS for WSNs are suggested. The advantage of the performed review is that it is based on three criteria: forward error correction (FEC), adaptive error correction methods, and other methods. The lack of algorithmic support within the framework of the review can be considered a disadvantage.

In [15], a systematic review of existing codes with the possibility of error correction for various applications is given. It has been proven that wireless sensor networks require reliable data transmission with limited power consumption, power dissipation and high performance. Thus, the selection of the optimal coding for a specific sensor network is carried out taking into account performance and energy consumption. Two coding algorithms based on block and convolutional codes are proposed. The advantage of the work is that the issues of alternative selection of the method of detection and correction of transmission errors for specific conditions of use are considered. Unfortunately, practical issues of methods implementation are somewhat ignored in the work, which is a drawback.

Work [16] contains an overview of various error correction methods, including the Hamming code. It is specifically stated that the Hamming code (7,4) is used to correct a single error. The extended Hamming (8,4) code has an extra bit and is used for single error correction as well as double error detection. A comparison of the advantages and disadvantages of this method with other methods of error correction was also carried out, which made it possible to evaluate the effectiveness of different methods. The advantage of the work is the availability of the mathematical foundations of error detection and correction methods. But the work does not sufficiently consider the practical issues of implementation of the specified methods, as well as examples of the implementation of the specified methods.

In [17], the general shortcomings of scientific works on coding theory are given. First, the practicalities of implementing a particular decoding algorithm on real hardware are largely ignored. The information that is available is patchy and scattered. Second, the information needed to evaluate a particular technique in situations encountered in practice is mostly available only in the reports of private companies. The work [17] is aimed at solving both of these problems. The advantage of the work is to highlight the general problems of implementing methods for detecting and correcting data transmission errors. But a somewhat simplified approach to the description of algorithms is a significant drawback of the work.

The performed review of literature data allows to state that there are and are used a large number of error detection and correction methods, both standard and modified. All of these methods are used to reduce data transmission and storage errors, with most methods allowing detection and correction of single errors and detection of multiple errors. The implementation of codes with an optimized number of check bits reduces the loss of time for encoding-decoding, but does not solve the problem of redundancy, the cumbersomeness of encoding-decoding algorithms and the loss of time. It can be argued that there is a problem of finding a compromise between the consistency of the algorithm for correcting multiple message errors and the performance of the algorithm in reducing the number of information transmission errors.

The development of IoT systems leads to an increase in extensive industrial networks, which are affected by electromagnetic interference, causing various types of data transmission errors. This is especially true of object condition monitoring systems based on IoT technologies, which have a large number of primary information providers. Therefore, the introduction of IoT technologies into industrial systems requires the development of new methods for detecting and correcting information transmission errors.

It is considered possible to develop such a method of detecting and correcting several errors in one byte belonging to a certain information block. And precisely due to the use of Hamming codes and the original coding scheme, as well as coding and decoding algorithms.

This method of detecting and correcting transmission errors can be effectively used in distributed IoT systems for monitoring the state of objects.

3. The aim and objectives of the study

The aim of the study is to increase the immunity of data transmission in branched IoT systems by developing a method for detecting and correcting multiple errors during the transmission of individual bytes belonging to one information block.

The objectives of the study are as follows:

– develop a coding scheme for detecting and correcting several errors in one byte based on Hamming codes;

– based on the coding scheme, develop a coding pro-cedure and a decoding procedure with the detection and correction of multi-bit errors;

– develop coding and decoding algorithms, as well as a program for implementing the method of detecting and correcting multi-bit errors;

– evaluate the proposed method in an automated mode using a software model of the communication channel.

4. Materials and methods of the study

The object of the study is the IOT system for monitoring the state of objects.

The main hypothesis of the study is the assumption that there is a scheme and method of encoding information based on Hamming codes, using which it is possible to detect and correct multiple errors in the transmission of a specific byte during byte-by-byte transmission of an information block.

When solving the main research problem, a number of assumptions were made in the work. Namely: when transmitting an information block, its length remains unchanged, and the problem of determining the first byte of an information block is solved at the protocol level.

Also, to simplify the development of the method for detecting and correcting data transmission errors, only the transmission of information was considered in conditions where only transmission errors of different multiplicity occur during the transmission of a single byte. Without taking into account the interruption of transmission under the influence of interference, etc.

The communication channel for data transmission usually corresponds to the simplified structural diagram shown in Fig. 1 [3]. The output signal from the information source is converted into an input signal in the form of a binary code, which is then converted into a code sequence with some redundancy known as a code word. This redundancy is necessary for error correction during transmission over a physical communication channel. Redundancy, and thus the error-correcting ability of the coding, is evaluated using the code rate R.

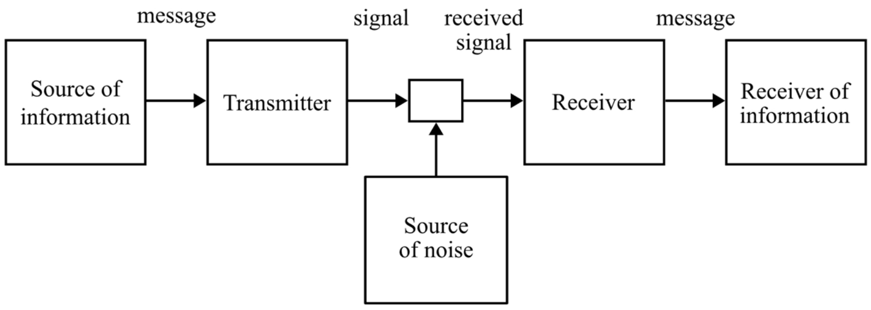

Fig. 1. Simplified structural diagram of the communication channel

The signal from the transmitter enters the real communication channel, where it is subjected to various distortions, such as additive noise, fading, etc. The receiver demodulates the signal received from the communication channel and converts it into an input signal that can be decoded.

In the absence of errors, the sequence of data and their evaluation at the output of the noise-resistant decoder coincide, otherwise the decoder performs error correction. The result of interference-tolerant decoding is the code word *U*, which is transmitted from the code decoder to the end receiver of the data. The upper limit of error-free transmission *C* for a given signal power, noise level, and occupied bandwidth is determined by Shannon's theorem [4]. In particular, if the transmission rate *R* does not exceed *C*, then there exists a code with codeword length n for which the error probability is equal to:

$$P(E)\leq 2^{-nE(R)}, \tag{1}$$

where $E(R)$ is a positive function of the argument *R*.

That is, on the one hand, an arbitrarily small error probability can be achieved by increasing the length of the code word n without reducing the transmission speed. But on the other hand, maintaining the transmission speed *R* by increasing the length of the code word n implies an increase in the length of the information word k, which leads to an exponential increase in the number of sequences of code words.

To test the main hypothesis, a method of detecting and correcting transmission errors based on the Hamming code was used. Hamming codes are one of the most widely used error correction approaches. When setting the minimum distance $d_{\min}=3$, the Hamming code allows to correct all single errors.

Hamming codes are redundant because they allow the specific symbols to be clearly identified. Symbols can be both informative and control. Such codes are usually called *n*, *k* codes, where *n* is the length of code combinations and *k* is the number of information symbols.

In Hamming codes, check symbols are the result of linear operations performed on a subset of symbols in a code combination.

The generation of *r* verification elements in combinations of these codes is based on *k* information elements, and the length of the code combination is $n=k+r$. Verification elements are linear combinations of information elements.

It should be noted that the minimum ratio of verification elements to information elements, at which the code can have error-correcting properties, is equal to: $2^r-1=n$. To calculate the main parameters of the Hamming code, it is necessary to set the number of verification elements *r*. The value of *n* can be determined from the expression above, and the number of information elements is equal to: $k=n-r$. The ratio between *k*, *r* and *n* for popular Hamming codes is given in the Table 1.

The Hamming code (12.8) is used in the development of the method for detecting and correcting multiple errors. It is a linear code that encodes eight bits of data into 12 bits by adding four check bits.

Table 1

Relationship between *k*, *r* and *n* for commonly used Hamming codes

Number of information bits k	4	4	8	9	10	11	11
Number of control bits r	3	4	4	4	4	4	5
Total number of bits n	7	8	12	13	14	15	16

A feature of the parity check matrix for the code with $d_{\min}=3$ is that its columns are different non-zero combinations of length r. For $r=4$, $n=12$, the parity check matrix of the Hamming code (12,8) has the following form:

$$H_{12,4}=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ n_1 & n_2 & n_3 & n_4 & n_5 & n_6 & n_7 & n_8 & n_9 & n_{10} & n_{11} & n_{12} \end{bmatrix}. \quad (2)$$

If to take the combinations of a four-element binary prime code and exclude all zero combinations, then it is quite easy to obtain a parity check matrix by writing all code combinations sequentially in the column of the $H_{12,4}$matrix. After permuting the columns with one "1" in each column, the matrix in formula (2) takes the form (3):

$$H_{12,4}=\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 & k_8 & r_1 & r_2 & r_3 & r_4 \end{bmatrix}. \quad (3)$$

From the matrix (3), it is possible to get a system of test equations, from which it is possible to find the values of the test discharges. When using the Hamming code (12,8), the specified eight information bits (k_1, ..., k_8), grouped at the beginning of the code word, are supplemented with four control symbols (r_1, ..., r_4). At the same time, the formulas for checking their parity can be defined as follows.

Equations for each control bit are constructed as follows. If it is necessary to find the equation for bit r_1 (the least significant bit), then the sum modulo the two least information bits (k_1, ..., k_8) which values in the least significant digits of the matrix (3) are equal to one. In this case, the value of bit r_1 will have the form given in (4).

Similarly, equations can be found for other control bits.

The system of equations for bits (r_1, ..., r_4) will have the form:

$$\begin{cases} r_4=k_5\oplus k_6\oplus k_7\oplus k_8; \\ r_3=k_2\oplus k_3\oplus k_4\oplus k_8; \\ r_2=k_1\oplus k_3\oplus k_4\oplus k_6\oplus k_7; \\ r_1=k_1\oplus k_2\oplus k_4\oplus k_5\oplus k_7. \end{cases} \quad (4)$$

For convenience, the system of equations (4) can be presented in the form of a Table 2.

The value of each check bit is zero if the number of ones in the checked bits is even. An odd sum of ones indicates an error in one of the checked bits. If the parity bits are set in this way, it is possible to determine in which bit an error occurred during transmission. The basic rule for choosing a formula for each parity bit is that when one information bit changes, at least two parity bits must change. The set of parity bits (r_1, ..., r_4) for all eight parity checks can be thought of as a hexadecimal number R.

Table 2

Procedure for calculating control bits

Parity bits	Information bits															
r_4	=	0	⊕	0	⊕	0	⊕	0	⊕	k_5	⊕	k_6	⊕	k_7	⊕	k_8
r_3	=	0	⊕	k_2	⊕	k_3	⊕	k_4	⊕	0	⊕	0	⊕	0	⊕	k_8
r_2	=	k_1	⊕	0	⊕	k_3	⊕	k_4	⊕	0	⊕	k_6	⊕	k_7	⊕	0
r_1	=	k_1	⊕	k_2	⊕	0	⊕	k_4	⊕	k_5	⊕	0	⊕	k_7	⊕	0

Let r_1=1, r_2=0, r_3=1, r_4=1, then, according to the Table 2:

$$R=1r_1+2r_2+4r_3+8r_4=1*1+2*0+4*1+8*1=13_{10}=0\text{DH}.$$

The appearance of an error in the code word leads to a violation of the parity check equations (4), which include an erroneous bit. If an error occurred in the fourth information bit, then the first, second and third equations of system (4) will not be fulfilled. In this case, the value of the bits will be equal to: (r_1, ..., r_4)=1110b [corresponding to the fourth column of the matrix in equation (3)].

The set of elements found by summation modulo two accepted verification elements and verification elements calculated by summation modulo two accepted information elements is called a syndrome. The calculation is carried out according to the same rule that is used to determine them on the transmitting side.

The set of elements found as the sum modulo two of the parity check elements received at the receiving end and the parity check elements calculated from the received information bits is called a syndrome. Thus, the column of matrix H, which corresponds to the calculated syndrome, indicates the location of the error. The calculated value of the syndrome must necessarily correspond to one of the columns of the matrix H, since all possible r-bit binary combinations are selected as columns. For the (12,8) Hamming code, see matrix (2), the results of a bitwise comparison of elements form a set of control combinations (S_1, ..., S_4), that is, syndromes, as shown in Table 3.

To do this, the parity check elements received from the data channel are summed with the calculated parity check elements using bitwise summation modulo two. There are eight syndromes, and once the syndrome is obtained in the decoding process, error correction of the information bit can be easily achieved by inverting the corresponding bit. After the errors are corrected, the decoding process is considered complete.

Table 3

Value of syndromes for code H (12,8) (control digits are located after information digits)

S_4	S_3	S_2	S_1	Value of syndrome	Diagnosis according to the meaning of the syndrome	What needs to be done to correct the received information
1	2	3	4	5	6	7
0	0	0	0	00H	There are no errors	No adjustment is required

Continuation of Table 3

1	2	3	4	5	6	7
0	0	0	1	01H	Error. Control bit r_1	No adjustment is required
0	0	1	0	02H	Error. Control bit r_2	No adjustment is required
0	0	1	1	03H	Error. Information bit k_1	Inversion of information bit k_1
0	1	0	0	04H	Error. Control bit r_3	No adjustment is required
0	1	0	1	05H	Error. Information bit k_2	Inversion of information bit k_2
0	1	1	0	06H	Error. Information bit k_3	Inversion of information bit k_3
0	1	1	1	07H	Error. Information bit k_4	Inversion of information bit k_4
1	0	0	0	08H	Error. Control bit r_4	No adjustment is required
1	0	0	1	09H	Error. Information bit k_5	Inversion of information bit k_5
1	0	1	0	0AH	Error. Information bit k_6	Inversion of information bit k_6
1	0	1	1	0BH	Error. Information bit k_7	Inversion of information bit k_7
1	1	0	0	0CH	Error. Information bit k_8	Inversion of information bit k_8
1	1	0	1	0DH	Multiple error	No adjustment is required
1	1	1	0	0EH	Multiple error	No adjustment is required
1	1	1	1	0FH	Multiple error	No adjustment is required

5. Research results of the method of detection and correction of data transmission errors based on the Hamming code

5. 1. The original coding scheme of the information block for detection and correction of several transmission errors in one byte based on Hamming codes

The construction of the coding scheme was carried out in three main stages.

At the first stage, the information block was generated, the location of the bits of which corresponded to the Table 4.

At the same time, four control bits were calculated for the bits of the same digits of the first eight bytes of the information block. That is, 12 bits of one digit of 12 bytes form the Hamming code (12,8). A total of 32 control bits are calculated for the first eight bytes, which form 8 control bytes. For the second eight bytes of the information block, also for bits of the same digits, four parity check bits were calculated as described above. Thus, for 16 bytes of the information block, 8 bytes containing control bits were calculated. That is, after the 16 information bytes, there are first four bytes that consist of the check bits of the first eight bytes, and then four more bytes that contain the check bits calculated from the second eight bytes.

For each 16 bytes of information, 64 control bits are calculated. That is, sixteen information bytes are considered as an 8×16 information bit matrix, to which an 8×8 parity bit matrix is added. In this case, all control bits r for the first 8 bytes (Table 4) were calculated as shown in Table 5.

Table 5

The order of calculation of control bits of the first eight bytes

Control bit	Information bits														
$r_{i,4}$=	0	⊕	0	⊕	0	⊕	0	⊕	$k_{i,5}$	⊕	$k_{i,6}$	⊕	$k_{i,7}$	⊕	$k_{i,8}$
$r_{i,3}$=	0	⊕	$k_{i,2}$	⊕	$k_{i,3}$	⊕	$k_{i,4}$	⊕	0	⊕	0	⊕	0	⊕	$k_{i,8}$
$r_{i,2}$=	$k_{i,1}$	⊕	0	⊕	$k_{i,3}$	⊕	$k_{i,4}$	⊕	0	⊕	$k_{i,6}$	⊕	$k_{i,7}$	⊕	0
$r_{i,1}$=	$k_{i,1}$	⊕	$k_{i,2}$	⊕	0	⊕	$k_{i,4}$	⊕	$k_{i,5}$	⊕	0	⊕	$k_{i,7}$	⊕	0

Note: i=1...8.

For bytes with numbers (9, ..., 16), control bits were calculated as shown in the table. 6. It should be noted that Table 6 is essentially a modification of Table 5, where only the indices i have been changed.

Table 4

Location of bits in the information block

Bytes of the transfer buffer																							
$n_{i,1}$	$n_{i,2}$	$n_{i,3}$	$n_{i,4}$	$n_{i,5}$	$n_{i,6}$	$n_{i,7}$	$n_{i,8}$	$n_{i,9}$	$n_{i,10}$	$n_{i,11}$	$n_{i,12}$	$n_{i,13}$	$n_{i,14}$	$n_{i,15}$	$n_{i,16}$	$n_{i,17}$	$n_{i,18}$	$n_{i,19}$	$n_{i,20}$	$n_{i,21}$	$n_{i,22}$	$n_{i,23}$	$n_{i,24}$
Information bits																Control bits							
$k_{i,1}$	$k_{i,2}$	$k_{i,3}$	$k_{i,4}$	$k_{i,5}$	$k_{i,6}$	$k_{i,7}$	$k_{i,8}$	$k_{i,9}$	$k_{i,10}$	$k_{i,11}$	$k_{i,12}$	$k_{i,13}$	$k_{i,14}$	$k_{i,15}$	$k_{i,16}$	$r_{i,1}$	$r_{i,2}$	$r_{i,3}$	$r_{i,4}$	$r_{i,5}$	$r_{i,6}$	$r_{i,7}$	$r_{i,8}$
$k_{8,1}$	$k_{8,2}$	$k_{8,3}$	$k_{8,4}$	$k_{8,5}$	$k_{8,6}$	$k_{8,7}$	$k_{8,8}$	$k_{8,9}$	$k_{8,10}$	$k_{8,11}$	$k_{8,12}$	$k_{8,13}$	$k_{8,14}$	$k_{8,15}$	$k_{8,16}$	$r_{8,1}$	$r_{8,2}$	$r_{8,3}$	$r_{8,4}$	$r_{8,5}$	$r_{8,6}$	$r_{8,7}$	$r_{8,8}$
$k_{7,1}$	$k_{7,2}$	$k_{7,3}$	$k_{7,4}$	$k_{7,5}$	$k_{7,6}$	$k_{7,7}$	$k_{7,8}$	$k_{7,9}$	$k_{7,10}$	$k_{7,11}$	$k_{7,12}$	$k_{7,13}$	$k_{7,14}$	$k_{7,15}$	$k_{7,16}$	$r_{7,1}$	$r_{7,2}$	$r_{7,3}$	$r_{7,4}$	$r_{7,5}$	$r_{7,6}$	$r_{7,7}$	$r_{7,8}$
$k_{6,1}$	$k_{6,2}$	$k_{6,3}$	$k_{6,4}$	$k_{6,5}$	$k_{6,6}$	$k_{6,7}$	$k_{6,8}$	$k_{6,9}$	$k_{6,10}$	$k_{6,11}$	$k_{6,12}$	$k_{6,13}$	$k_{6,14}$	$k_{6,15}$	$k_{6,16}$	$r_{6,1}$	$r_{6,2}$	$r_{6,3}$	$r_{6,4}$	$r_{6,5}$	$r_{6,6}$	$r_{6,7}$	$r_{6,8}$
$k_{5,1}$	$k_{5,2}$	$k_{5,3}$	$k_{5,4}$	$k_{5,5}$	$k_{5,6}$	$k_{5,7}$	$k_{5,8}$	$k_{5,9}$	$k_{5,10}$	$k_{5,11}$	$k_{5,12}$	$k_{5,13}$	$k_{5,14}$	$k_{5,15}$	$k_{5,16}$	$r_{5,1}$	$r_{5,2}$	$r_{5,3}$	$r_{5,4}$	$r_{5,5}$	$r_{5,6}$	$r_{5,7}$	$r_{5,8}$
$k_{4,1}$	$k_{4,2}$	$k_{4,3}$	$k_{4,4}$	$k_{4,5}$	$k_{4,6}$	$k_{4,7}$	$k_{4,8}$	$k_{4,9}$	$k_{4,10}$	$k_{4,11}$	$k_{4,12}$	$k_{4,13}$	$k_{4,14}$	$k_{4,15}$	$k_{4,16}$	$r_{4,1}$	$r_{4,2}$	$r_{4,3}$	$r_{4,4}$	$r_{4,5}$	$r_{4,6}$	$r_{4,7}$	$r_{4,8}$
$k_{3,1}$	$k_{3,2}$	$k_{3,3}$	$k_{3,4}$	$k_{3,5}$	$k_{3,6}$	$k_{3,7}$	$k_{3,8}$	$k_{3,9}$	$k_{3,10}$	$k_{3,11}$	$k_{3,12}$	$k_{3,13}$	$k_{3,14}$	$k_{3,15}$	$k_{3,16}$	$r_{3,1}$	$r_{3,2}$	$r_{3,3}$	$r_{3,4}$	$r_{3,5}$	$r_{3,6}$	$r_{3,7}$	$r_{3,8}$
$k_{2,1}$	$k_{2,2}$	$k_{2,3}$	$k_{2,4}$	$k_{2,5}$	$k_{2,6}$	$k_{2,7}$	$k_{2,8}$	$k_{2,9}$	$k_{2,10}$	$k_{2,11}$	$k_{2,12}$	$k_{2,13}$	$k_{2,14}$	$k_{2,15}$	$k_{2,16}$	$r_{2,1}$	$r_{2,2}$	$r_{2,3}$	$r_{2,4}$	$r_{2,5}$	$r_{2,6}$	$r_{2,7}$	$r_{2,8}$
$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$	$k_{1,6}$	$k_{1,7}$	$k_{1,8}$	$k_{1,9}$	$k_{1,10}$	$k_{1,11}$	$k_{1,12}$	$k_{1,13}$	$k_{1,14}$	$k_{1,15}$	$k_{1,16}$	$r_{1,1}$	$r_{1,2}$	$r_{1,3}$	$r_{1,4}$	$r_{1,5}$	$r_{1,6}$	$r_{1,7}$	$r_{1,8}$

Table 6

The order of calculation of control bits of bytes 9…16

Control bit	Information bits														
$r_{i,8}$=	0	⊕	0	⊕	0	⊕	0	⊕	$k_{i,13}$	⊕	$k_{i,14}$	⊕	$k_{i,15}$	⊕	$k_{i,16}$
$r_{i,7}$=	0	⊕	$k_{i,10}$	⊕	$k_{i,11}$	⊕	$k_{i,12}$	⊕	0	⊕	0	⊕	0	⊕	$k_{i,16}$
$r_{i,6}$=	$k_{i,9}$	⊕	0	⊕	$k_{i,11}$	⊕	$k_{i,12}$	⊕	0	⊕	$k_{i,14}$	⊕	$k_{i,15}$	⊕	0
$r_{i,5}$=	$k_{i,9}$	⊕	$k_{i,10}$	⊕	0	⊕	$k_{i,12}$	⊕	$k_{i,13}$	⊕	0	⊕	$k_{i,15}$	⊕	0

Note: i=1…8.

The next stage was the shuffling of data in the information block by applying bitwise shift operations to the information bits and parity bits according to the Table 7. In this process, all least significant bits (those in bit position 1) of all bytes in the transmit buffer remain unchanged (no shift is performed). All bits of all bytes in the information block of the second digit are shifted one bit to the right, and so on.

The arrangement of bits in the information block after performing these right shift operations is shown in Table 7.

The third and final stage of the formation of the information block is also the shuffling of bits. This shuffling was performed by bitwise shifting operations from low to high bits.

The final proposed coding scheme has the form shown in Table 8. The information from the transmission buffer was then transmitted by individual bytes to the data transmission channel.

On the receiving side, the received information (24 bytes) is stored in the reception buffer. When all 24 bytes of the information block are received, bitwise shift operations are performed on each byte of the receive buffer. In reverse order, relative to the bitwise shift operations performed during the preparation of the transmit buffer.

5. 2. Encoding procedure and decoding procedure with detection and correction of multi-bit errors

First, an information block is formed from information and control bits. Control circuits are calculated by performing linear operations. The known Hamming code is used to calculate the check bits, and the check bits are calculated for the information bits of the same byte bits of the information block. The Hamming code (12.8) was used in the research. The coding scheme according to which the information and control bits are placed in the transmission buffer is given above.

In this form, byte-by-byte transmission of the information block is carried out. After receiving the entire information block (in the case of the experiment, the size is 24 bytes), bitwise shift operations are performed in the reverse direction relative to the shift operations used during encoding.

As a result, the received information in the reception buffer will be in the same order as shown in the Table 4, but some bits may be distorted due to physical effects on the communication channel.

Next, control bits are calculated for all 16 received information bytes. In the next step, bitwise exclusive OR operations are performed to compare the corresponding received check bits with the check bits that were calculated based on the received 16 information bytes.

The result of these operations are 16 syndromes, each of which allows diagnosing the presence of errors in the transmission of the corresponding code words, and also provides the opportunity to correct errors, if they occurred during the transmission of information bits.

The algorithm for diagnosing and correcting errors of transmitted information bits was implemented using the Table 3, which shows all possible values of individual syndromes in hexadecimal format.

Table 7

The structure of placing information in the information block after performing right shift operations

Bytes of the transfer buffer																							
$n_{i,1}$	$n_{i,2}$	$n_{i,3}$	$n_{i,4}$	$n_{i,5}$	$n_{i,6}$	$n_{i,7}$	$n_{i,8}$	$n_{i,9}$	$n_{i,10}$	$n_{i,11}$	$n_{i,12}$	$n_{i,13}$	$n_{i,14}$	$n_{i,15}$	$n_{i,16}$	$n_{i,17}$	$n_{i,18}$	$n_{i,19}$	$n_{i,20}$	$n_{i,21}$	$n_{i,22}$	$n_{i,23}$	$n_{i,24}$
$r_{8,2}$	$r_{8,3}$	$r_{8,4}$	$r_{8,5}$	$r_{8,6}$	$r_{8,7}$	$r_{8,8}$	$k_{8,1}$	$k_{8,2}$	$k_{8,3}$	$k_{8,4}$	$k_{8,5}$	$k_{8,6}$	$k_{8,7}$	$k_{8,8}$	$k_{8,9}$	$k_{8,10}$	$k_{8,11}$	$k_{8,12}$	$k_{8,13}$	$k_{8,14}$	$k_{8,15}$	$k_{8,16}$	$r_{8,1}$
$r_{7,3}$	$r_{7,4}$	$r_{7,5}$	$r_{7,6}$	$r_{7,7}$	$r_{7,8}$	$k_{7,1}$	$k_{7,2}$	$k_{7,3}$	$k_{7,4}$	$k_{7,5}$	$k_{7,6}$	$k_{7,7}$	$k_{7,8}$	$k_{7,9}$	$k_{7,10}$	$k_{7,11}$	$k_{7,12}$	$k_{7,13}$	$k_{7,14}$	$k_{7,15}$	$k_{7,16}$	$r_{7,1}$	$r_{7,2}$
$r_{6,4}$	$r_{6,5}$	$r_{6,6}$	$r_{6,7}$	$r_{6,8}$	$k_{6,1}$	$k_{6,2}$	$k_{6,3}$	$k_{6,4}$	$k_{6,5}$	$k_{6,6}$	$k_{6,7}$	$k_{6,8}$	$k_{6,9}$	$k_{6,10}$	$k_{6,11}$	$k_{6,12}$	$k_{6,13}$	$k_{6,14}$	$k_{6,15}$	$k_{6,16}$	$r_{6,1}$	$r_{6,2}$	$r_{6,3}$
$r_{5,5}$	$r_{5,6}$	$r_{5,7}$	$r_{5,8}$	$k_{5,1}$	$k_{5,2}$	$k_{5,3}$	$k_{5,4}$	$k_{5,5}$	$k_{5,6}$	$k_{5,7}$	$k_{5,8}$	$k_{5,9}$	$k_{5,10}$	$k_{5,11}$	$k_{5,12}$	$k_{5,13}$	$k_{5,14}$	$k_{5,15}$	$k_{5,16}$	$r_{5,1}$	$r_{5,2}$	$r_{5,3}$	$r_{5,4}$
$r_{4,6}$	$r_{4,7}$	$r_{4,8}$	$k_{4,1}$	$k_{4,2}$	$k_{4,3}$	$k_{4,4}$	$k_{4,5}$	$k_{4,6}$	$k_{4,7}$	$k_{4,8}$	$k_{4,9}$	$k_{4,10}$	$k_{4,11}$	$k_{4,12}$	$k_{4,13}$	$k_{4,14}$	$k_{4,15}$	$k_{4,16}$	$r_{4,1}$	$r_{4,2}$	$r_{4,3}$	$r_{4,4}$	$r_{4,5}$
$r_{3,7}$	$r_{4,8}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$	$k_{3,4}$	$k_{3,5}$	$k_{3,6}$	$k_{3,7}$	$k_{3,8}$	$k_{3,9}$	$k_{3,10}$	$k_{3,11}$	$k_{3,12}$	$k_{3,13}$	$k_{3,14}$	$k_{3,15}$	$k_{3,16}$	$r_{3,1}$	$r_{3,3}$	$r_{3,3}$	$r_{3,4}$	$r_{3,5}$	$r_{3,6}$
$r_{2,8}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$	$k_{2,4}$	$k_{2,5}$	$k_{2,6}$	$k_{2,7}$	$k_{2,8}$	$k_{2,9}$	$k_{2,10}$	$k_{2,11}$	$k_{2,12}$	$k_{2,13}$	$k_{2,14}$	$k_{2,15}$	$k_{2,16}$	$r_{2,1}$	$r_{2,2}$	$r_{2,3}$	$r_{2,4}$	$r_{2,5}$	$r_{2,6}$	$r_{2,7}$
$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$	$k_{1,6}$	$k_{1,7}$	$k_{1,8}$	$k_{1,9}$	$k_{1,10}$	$k_{1,11}$	$k_{1,12}$	$k_{1,13}$	$k_{1,14}$	$k_{1,15}$	$k_{1,16}$	$r_{1,1}$	$r_{1,2}$	$r_{1,3}$	$r_{1,4}$	$r_{1,5}$	$r_{1,6}$	$r_{1,7}$	$r_{1,8}$

Table 8

Information encoding scheme in the transmission buffer

Bytes of the transfer buffer																							
$n_{i,1}$	$n_{i,2}$	$n_{i,3}$	$n_{i,4}$	$n_{i,5}$	$n_{i,6}$	$n_{i,7}$	$n_{i,8}$	$n_{i,9}$	$n_{i,10}$	$n_{i,11}$	$n_{i,12}$	$n_{i,13}$	$n_{i,14}$	$n_{i,15}$	$n_{i,16}$	$n_{i,17}$	$n_{i,18}$	$n_{i,19}$	$n_{i,20}$	$n_{i,21}$	$n_{i,22}$	$n_{i,23}$	$n_{i,24}$
$r_{8,2}$	$r_{7,4}$	$r_{6,6}$	$r_{5,8}$	$k_{4,2}$	$k_{3,4}$	$k_{2,6}$	$k_{1,8}$	$k_{8,2}$	$k_{7,4}$	$k_{5,6}$	$k_{5,8}$	$k_{4,10}$	$k_{3,12}$	$k_{2,14}$	$k_{1,16}$	$k_{8,10}$	$k_{7,12}$	$k_{6,14}$	$k_{5,16}$	$r_{4,2}$	$r_{3,4}$	$r_{2,6}$	$r_{1,8}$
$r_{7,3}$	$r_{6,5}$	$r_{5,7}$	$k_{4,1}$	$k_{3,3}$	$k_{2,5}$	$k_{1,7}$	$k_{8,1}$	$k_{7,3}$	$k_{5,5}$	$k_{5,7}$	$k_{4,9}$	$k_{3,11}$	$k_{2,13}$	$k_{1,15}$	$k_{8,9}$	$k_{7,11}$	$k_{6,13}$	$k_{5,15}$	$r_{4,1}$	$r_{3,3}$	$r_{2,5}$	$r_{1,7}$	$r_{8,1}$
$r_{6,4}$	$r_{5,6}$	$r_{4,8}$	$k_{3,2}$	$k_{2,4}$	$k_{1,6}$	$r_{8,8}$	$k_{7,2}$	$k_{5,4}$	$k_{5,6}$	$k_{4,8}$	$k_{3,10}$	$k_{2,12}$	$k_{1,14}$	$k_{8,8}$	$k_{7,10}$	$k_{6,12}$	$k_{5,14}$	$k_{4,16}$	$r_{3,3}$	$r_{2,4}$	$r_{1,6}$	$k_{8,16}$	$r_{7,2}$
$r_{5,5}$	$r_{4,7}$	$k_{3,1}$	$k_{2,3}$	$k_{1,5}$	$r_{8,7}$	$k_{7,1}$	$k_{6,3}$	$k_{5,5}$	$k_{4,7}$	$k_{3,9}$	$k_{2,11}$	$k_{1,13}$	$k_{8,7}$	$k_{7,9}$	$k_{6,11}$	$k_{5,13}$	$k_{4,15}$	$r_{3,1}$	$r_{2,3}$	$r_{1,5}$	$k_{8,15}$	$r_{7,1}$	$r_{6,3}$
$r_{4,6}$	$r_{4,8}$	$k_{2,2}$	$k_{1,4}$	$r_{8,6}$	$r_{7,8}$	$k_{6,2}$	$k_{5,4}$	$k_{4,6}$	$k_{3,8}$	$k_{2,10}$	$k_{1,12}$	$k_{8,6}$	$k_{7,8}$	$k_{6,10}$	$k_{5,12}$	$k_{4,14}$	$k_{3,16}$	$r_{2,2}$	$r_{1,4}$	$k_{8,14}$	$k_{7,16}$	$r_{6,2}$	$r_{5,4}$
$r_{3,7}$	$k_{2,1}$	$k_{1,3}$	$r_{8,5}$	$r_{7,7}$	$k_{6,1}$	$k_{5,3}$	$k_{4,5}$	$k_{3,7}$	$k_{2,9}$	$k_{1,11}$	$k_{8,5}$	$k_{7,7}$	$k_{6,9}$	$k_{5,11}$	$k_{4,13}$	$k_{3,15}$	$r_{2,1}$	$r_{1,3}$	$k_{8,13}$	$k_{7,15}$	$r_{6,1}$	$r_{5,3}$	$r_{4,5}$
$r_{2,8}$	$k_{1,2}$	$r_{8,4}$	$r_{7,6}$	$r_{6,8}$	$k_{5,2}$	$k_{4,4}$	$k_{3,6}$	$k_{2,8}$	$k_{1,10}$	$k_{8,4}$	$k_{7,6}$	$k_{5,8}$	$k_{5,10}$	$k_{4,12}$	$k_{3,14}$	$k_{2,16}$	$r_{1,2}$	$k_{8,12}$	$k_{7,14}$	$k_{6,16}$	$r_{5,2}$	$r_{4,4}$	$r_{3,6}$
$k_{1,1}$	$r_{8,3}$	$r_{7,5}$	$r_{6,7}$	$k_{5,1}$	$k_{3,3}$	$k_{3,5}$	$k_{2,7}$	$k_{1,9}$	$k_{8,3}$	$k_{7,5}$	$k_{5,7}$	$k_{5,9}$	$k_{4,11}$	$k_{3,13}$	$k_{2,15}$	$r_{1,1}$	$k_{8,11}$	$k_{7,13}$	$k_{6,15}$	$k_{5,1}$	$r_{4,3}$	$r_{3,5}$	$r_{2,7}$

In the general case, when the checksum received from the communication channel differed from the checksum calculated on the receiving side, one of the options was performed:

– if only one of the checksum bits does not match, then only the checksum bits are corrupted, not the information. Adjustments are not performed;

– if several checksum bits do not match, further validation is required. If one of the information bits does not match, then the corrupted information bit can be determined using the checksum bits. This error is corrected by inverting the corrupted information bit;

– all other scenarios indicate that multiple errors have occurred. At the same time, although it is impossible to correct the information, the user is informed of multiple errors.

The proposed method of transmitting information through communication channels in blocks of 24 bytes long allows for diagnostic detection and correction of transmission errors on the receiving side. Even in cases of multiple (multiple) errors within one byte belonging to a certain information block.

5. 3. Developed coding and decoding algorithms, as well as a program for implementing the method of detecting and correcting multi-bit errors

In order to test the main hypothesis, the proposed method of detecting and correcting multi-bit errors was implemented as a Python program. But it should be noted that the coding and decoding algorithms that implement the original method of detecting and correcting multiple errors can be implemented using different programming languages: C, C++, Java, and others.

The block – scheme of the coding algorithm is shown in Fig. 2.

The algorithm is quite concise, but needs some explanation. The *HammingCodec* class with methods for encoding and decoding data using the Hamming code (12.8) was used to perform the operations of calculating the control bits of one byte and the operations of correcting transmission errors.

In the Python language, such a concept as "methods" is widely used. A method is essentially a function that is "attached" to a certain object and that performs certain actions on this object. The techniques described below were also used in the specific implementation of the encoding and decoding algorithms.

The *calculate_control_bits* method is used to calculate control bits for a given array of data according to the Table 2. The method receives an array of data as an input, and returns an array of control bits.

The *horizontal_shift* method is used to perform a cyclic bit shift of row *n* by *n*–1 bits to the right or left according to the Table 7.

The *vertical_shift* method is used to perform a cyclic bit shift operations of column j by i bits up or down according to the Table 8.

The *encode* method takes an object of type bytes as input and encodes it using the Hamming code, as shown in Algorithm 1. It converts the bytes object to a uint8 array using the *frombuffer* method of the NumPy module. The control bits for the data array are then calculated using the *calculate_control_bits* method. The data is combined with the control bits to form the transmission buffer TBUF, which is shifted according to the Table 7 using *horizontal_shift* method, then shifted according to the Table 8 using *vertical_shift* method and then converted back to a bytes object using the *tobytes* method.

In Algorithms 1 and 2: *n* is the number of control bytes, *j* is the number of bytes in the message, and *i* is the number of offsets.

Algorithm 2, which implements the decoding procedure and error detection and correction, is shown in Fig. 3.

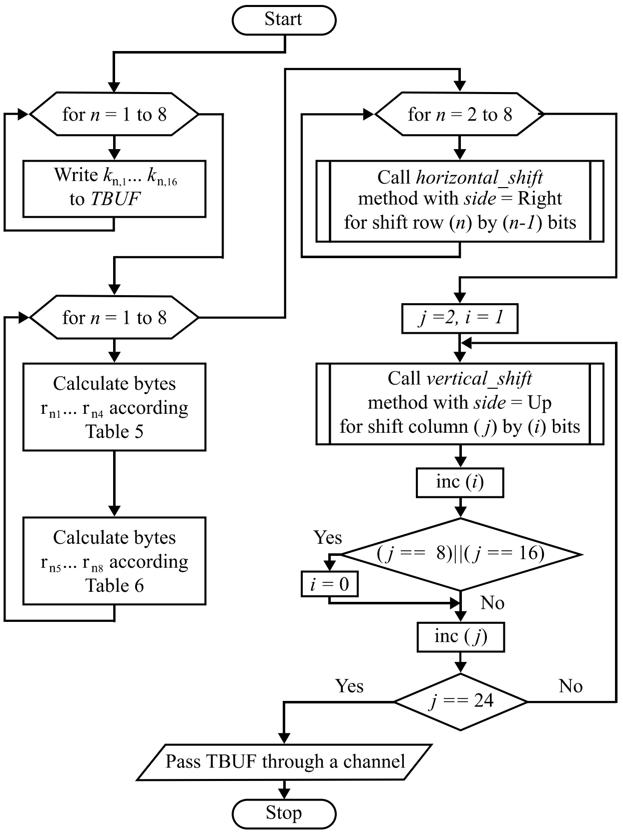
Start
for n = 1 to 8
Write $k_{n,1}$... $k_{n,16}$ to *TBUF*
for n = 1 to 8
Calculate bytes r_{n1}... r_{n4} according Table 5
Calculate bytes r_{n5}... r_{n8} according Table 6
for n = 2 to 8
Call *horizontal_shift* method with *side* = Right for shift row (n) by (n-1) bits
j =2, i = 1
Call *vertical_shift* method with *side* = Up for shift column (j) by (i) bits
inc (i)
Yes
(j == 8)||(j == 16)
i = 0
No
inc (j)
Yes
j == 24
No
Pass TBUF through a channel
Stop

Fig. 2. Block diagram for algorithm 1. Coding procedure

```
Algorithm 2. Decoding method
Input data: RBUF is a 24-byte object
Output data: input data, D
$RBUF \leftarrow data$
$j \leftarrow 2$ # column number
$i \leftarrow 1$ # shift count
 while $j \le 24$ do
     Call vertical_shift method for shift column ($j$) by ($i$) bits with side = Down
     $i \leftarrow i+1$
     If $(j == 8) || (j == 16)$ then $i \leftarrow 0$
     $j \leftarrow j+1$
 end while
 $n \leftarrow 2$ # row number ; $n-1$ – shift count
 while $n \le 8$ do
     Call horizontal_shift method for shift column ($n$) by ($n-1$) bits with side = Left
     $n \leftarrow n+1$
 end while
 $CBUF \leftarrow RBUF$ #CBUF is a control buffer
 Calculate control bytes $B_c17 \ldots B_c24$ using $B_c1 \ldots B_c16$ of CBUF
 If $(B_c17 \ldots B_c24) <> (r_1 \ldots r_8)$ then correct $k_1 \ldots k_{16}$ of RBUF using 16 syndromes
 Log errors and corrections to the service file
 $D \leftarrow RBUF$
return D
```

Fig. 3. Algorithm 2. Decoding procedure with detection and correction of multi-bit transmission errors

The decode method takes a bytes object as input and decodes it using a Hamming code, as shown in Algorithm 2. The bytes object is first converted to a uint8 array using the *frombuffer* method of the NumPy module, and based on the size of the RBUF receive buffer and the size of the output data, the size of the reception buffer is calculated. Then the data is decoded in the reverse sequence of the steps performed by the *encode* method.

5. 4. Evaluation of the interference resistance of the proposed method was performed when testing the software model in automated mode

The evaluation of the immunity of the proposed method of correcting multi-bit errors that occur during the transmission of information over the data transmission channel was carried out by means of automated testing. A software model of the data transmission channel was developed for the assessment. The structural diagram of the model is shown in Fig. 4.

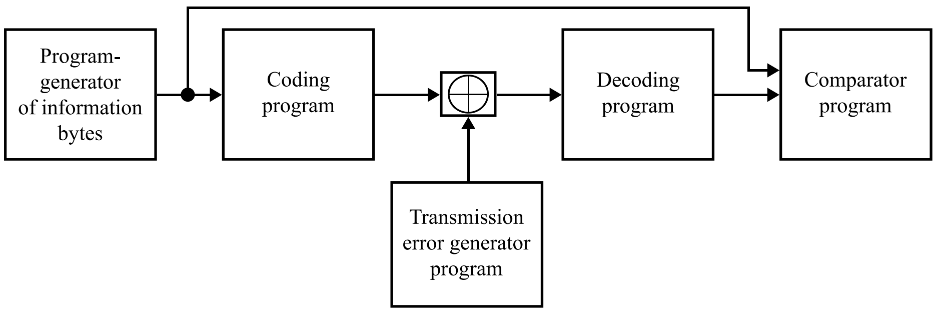
Program-generator of information bytes
Coding program
Decoding program
Comparator program
Transmission error generator program

Fig. 4. Structural diagram of the software model of the data transmission channel

The specified model was used in an automated mode to evaluate the performance of the proposed decoding method and to detect and correct information transmission errors.

The program model of the data transmission channel, which was used in testing, was written in the Python language.

Brief description of the structural scheme. The information byte generator program generates 16 information bytes that enter the inputs of the encoding program and the comparator program. The coding program performs coding according to the developed coding scheme (Table 8). The result of the encoding program is a 24-byte information block. The transmission error generator program also forms a 24-byte information block, but each bit of the information block can be either zero or one.

Next, the information blocks received from the encoding program and from the transmission error generator program are summarized using exclusive OR operations. In this way, external influence on the data transmission channel is simulated. The information block, 24 bytes long, obtained as a result of bitwise summation, enters the input of the decoding program. The result of the decoding program, which implements algorithm 2, is 16 recovered information bytes. This information enters the input of the comparator program. The comparator program compares the information received from the information byte generator program and the restored information from the decoding program. The result of the comparison is recorded and displayed.

Testing was carried out in an automated mode (the term “automated”, in contrast to the term “automatic”, emphasizes the preservation by the human operator of control functions of the most general, purposeful nature), according to the developed methodology. The automated mode made it possible to free the operator from numerous monotonous operations, leaving the operator with the opportunity to make decisions in the research process.

During the experiment, errors in the transmission of individual information bits of code words were artificially created by using linear EXCLUSIVE OR operations. The information block after decoding with error detection and correction is compared with the original information block, and transmission errors are captured and stored. A total of 6120 blocks were transferred during the experiment.

The results of the experiment are shown in Table 9.

Depending on the frequency of errors, the following absolute indicators were obtained. During testing, it has been assumed that the transmission is carried out by individual bytes, which, in turn, may contain several errors with a multiplicity of one to eight. Therefore, the following testing principle was implemented:

1. For each of the 24 bytes prepared for transmission by the communication channel, artificial errors with a multiplicity of one to eight were consistently simulated during testing.

2. At the same time, each block contained one byte with erroneous bits, and the multiplicity of errors in one byte varied from 1 to 8.

3. The proposed method and algorithm were used to correct errors and restore the original information.

4. After performing error correction, the obtained data were compared with the original ones, and the main indicators characterizing the ability and quality of error correction were also calculated.

5. Depending on the multiplicity of the error, the following absolute indicators were obtained (Table 9):

– error multiplicity in one block byte;
– the number of transferred blocks.
– the number of transmitted bits;
– the number of distorted bits;
– the number of errors in information bits;
– the number of corrected information bits;
– the number of uncorrectable parity bit errors.

After analyzing the obtained results, it can be stated:

1. If an error occurs in one byte of the block, regardless of its multiplicity (from 1 to 8), all information bits can be restored.

2. When the multiplicity of errors in one byte of an information block is equal to 8, there is a minimum number of blocks with erroneous bytes, which reaches 24. At the same time, all blocks will contain 128 informational corrupted bits that can be corrected.

3. When the multiplicity of errors in one byte of an information block is 4, there can be a maximum number of blocks with erroneous bytes, which reaches 322560. At the same time, all blocks will contain 4480 information corrupted bits that can be corrected.

Table 9

Results of testing the proposed method of detection and correction of multi-bit errors in the transmission of information blocks

Error multiplicity in one block byte	The number of transferred blocks	The number of transmitted bits	The number of distorted bits	The number of errors in information bits	The number of corrected information bits	The number of uncorrectable parity bit errors (do not correct)
1	192	36864	192	128	128	64
2	672	129024	1344	896	896	448
3	1344	258048	4032	2688	2688	1344
4	1680	322560	6720	4480	4480	2240
5	1344	258048	6720	4480	4480	2240
6	672	129024	4032	2688	2688	1344
7	192	36864	1344	896	896	448
8	24	4608	192	128	128	64

6. Discussion of the research results of the proposed method of tamper-resistant data coding in IoT systems

The main result of the work is the development of an innovative method of detecting and correcting multiple errors in the transmission or storage of information based on the well-known Hamming codes [18]. Also, as a result of the work, the following information block coding scheme, coding and decoding algorithms implementing the proposed method were developed and researched. A software model of the data transmission channel was developed and investigated to evaluate the reliability of the method of detecting and correcting multiple errors.

The method can be used in the case when information is transmitted by information blocks of fixed length (24 bits), and the transmission is carried out by code words with a length of one byte.

The development of software for testing the method allowed for a significant number of experiments. The obtained test results were obtained thanks to the use of research automation.

Python software was developed to test the proposed multiple error detection and correction method based on Hamming codes (12,8). This software made it possible to simulate the following processes: information encoding, information transmission and information decoding, generation of transmission errors and fixation of decoding results with the calculation of information recovery errors. Based on the research results of the program model of the data transmission channel, the following has been proven:

1. The proposed method for detecting and correcting multi-bit errors is based on Hamming codes (12,8) and performs the transmission of an information block consisting of 24 bytes in the form of a sequence of individual code words with a length of one byte.

2. The method has relatively high basic error correction rates.

3. When increasing the length of the information block by 1.5 times (8 control bytes are added to every 16 information bytes that contain the block before transmission) and when using the developed encoding-decoding algorithms, it is possible to achieve the following results:

– when transmitting an information block with code words of one-byte length, it is possible to detect transmission (storage) errors of a separate code word in multiples from 1 to 8. All detected transmission errors of a single code word with a multiplicity of 1 to 8 can be recovered by inverting the damaged bits;

– constant errors of one digit of one block are detected and corrected. For example, a permanent error in a specific bit of all 24 bytes.

Application of the proposed method makes it possible to increase the interference resistance of the data transmission channel.

Particular attention should also be paid to the formation on the receiving side of the information block, as the proposed method is sensitive to changes in the length of the block.

The obtained results are explained by the use of a new information coding scheme, which is based on the use of Hamming codes.

The use of the proposed method makes it possible to significantly increase the immunity of data transmission channels of distributed IoT systems. For example, systems such as facility condition monitoring systems. To some extent, this is proven in work [14], where a systematic review of existing codes with the possibility of error correction for various applications is given. Also, in [15], it was noted that Hamming codes are widely used in wireless sensor networks to reduce data transmission errors and increase communication reliability.

The ability to detect and correct multiple errors of a single code word with a multiplicity from 1 to 8 favorably distinguishes the proposed method from known ones, which usually allow correcting single and detecting double errors of code word transmission (storage). It is interesting to compare the proposed method with known coding schemes and error detection and correction methods, the analysis of which is given in [16]. In this work, a large family of error-correcting block codes is considered. The possibility of using different types of block codes for different applications is considered. Also, [16] proposed a coding scheme for correcting single and double contiguous errors, which allows detecting 2-bit errors and correcting 1-bit errors.

A positive point is the ease of implementation of the proposed coding and decoding algorithms that implement the proposed method of detecting and correcting multiple transmission errors. The possibility of using different programming languages for the programmatic implementation of the method is also a positive point that expands the scope of use.

The use of Hamming codes is relevant today because there are applications in a wide range of areas where the reliability of data transmission is critical. The rapid development of industrial IoT networks, which ensure the transfer of primary information from sensors to a processing and decision-making device, are usually built without the use of error detection and correction methods. The proposed method can be easily implemented in practice due to the simplicity of the encoding and decoding algorithms and the possibility of using a large number of programming languages.

The proposed method of detecting and correcting multiple errors is interesting in that, thanks to the original use of known Hamming codes, it was possible to provide increased immunity compared to the classical variant of using Hamming codes. First of all, this was achieved thanks to the development of the coding scheme of the information block. The peculiarity of the coding scheme is that the control bits are calculated and added not to each code word that is transmitted over the data transmission channel, but according to the scheme of the developed coding scheme of the information block. Also, the proposed coding scheme provides for shuffling with the help of shifting operations of information and control bits before transmission to the communication channel, which prevents the occurrence of multiple errors with frequent distortion of a specific bit of code words. As for operations on the receiving side, after receiving the entire information block, in accordance with the proposed method, it is necessary to perform decoding. The decoding algorithm is based on performing shift operations, but in the reverse order of the shift operations that were performed during encoding.

The resulting block will have the structure shown in the Table 4. In the presence of interference, some bits will be damaged. Detecting and fixing corrupted bits is already a trivial task. The essence of the method is that due to the special mixing of the bits of the information block, which consists of information and control bits, a block is formed for transmission to the channel. Moreover, the transmission of such a prepared block with code words of one-byte length leads to the fact that the transmission errors that occur in a separate code word belong to different code combinations. Therefore, even in the case of multiple errors in a single codeword, they can be detected and corrected by inverting the corrupted bits.

Further development of the proposed method is possible. For example, it is really possible to reduce the loss of coding by using Hamming codes of a higher bit rate, but this requires a change in the information coding scheme.

The difference of the proposed method is that when the information block is transmitted sequentially byte by byte, it is possible to detect and correct transmission (storage) errors of a single byte in multiples of 1 to 8. The positive results of the development and testing of the method of detecting and correcting multi-bit errors (Table 9) can be explained by the use of the original coding scheme.

Redundancy can be considered the main drawback of the proposed method. However, this drawback, as stated in [11], is not critical from the point of view of using interference-resistant coding. This is because in many applications fast encoding and decoding is more important than redundancy. The use of the method is limited to those cases when the information block does not change in length during transmission, but only bit transmission errors occur. The use of the method may have limited use in the case of the need to organize high-speed data transfer. The fact that the process of detecting the beginning of the information block on the receiving side is not considered is also a limitation of the study. But this problem can be solved due to the exchange protocol.

A significant drawback of the method can be considered the unreliability of the transmission results in case of complete loss of individual bytes during the transmission of the information block over the information transmission channel. Excessive coding is also a significant drawback of the developed method.

In the future, it is possible to reduce coding redundancy if a coding optimization procedure is developed by using Hamming codes of different lengths.

7. Conclusions

1. The developed coding scheme has been confirmed the correctness of the decisions made and made it possible to create a workable software model of the data transmission channel based on the use of the proposed error detection and correction methods. The proposed coding scheme is based on the permutation of bits in the information block before transmitting the block over the communication channel. In the case when data is transmitted by separate code words, it allows diagnosing and correcting multiple errors when transmitting individual bytes belonging to one information block. Only one of the possible ways of permuting the bits in the information block before its transmission over the communication channel has been considered. But the methods of permuting bits within the information block can be different. It is considered interesting to implement the permutation of bits in the information block based on the use of a certain key. That is, the order of bitwise operations used when permuting the bits of a data block before transmission over a communication channel, as well as after receiving a data block at the receiving end, may depend on the key. In this case, the proposed method of detecting and correcting multi-bit errors in information transmission will be supplemented with the encryption function using a key.

2. Encoding procedure and decoding procedure with detection and correction of multi-bit errors developed on the basis of the coding scheme made it possible to significantly increase the interference resistance of the information transmission channel, which was confirmed by testing the software model implemented in the Python language. The developed procedures for detecting and correcting multiple errors can be used for a variety of data transmission channels. These procedures make particular sense to use in distributed IoT systems due to their large linear dimensions and high level of electromagnetic field interference.

3. Coding and decoding algorithms have been developed, which made it possible to implement the proposed method of detecting and correcting several errors in one byte based on Hamming codes when transmitting information over a communication channel. The features of the developed algorithms are that they allow, when transmitting an information block with code words of one-byte length, to detect transmission (storage) errors of a separate code word with a multiplicity from 1 to 8. That is, a single corrupted byte will be corrected after performing the decoding algorithm even if it consists of eight corrupted bits. The developed algorithms can be used to transmit information through serial communication channels. They can also be applied in systems for monitoring the state of potentially dangerous objects, IoT systems, sensor networks as well.

4. The evaluation of the proposed method of detecting and correcting multiple errors has been carried out using the developed software model of the information block

transmission channel. The program model of the data transmission channel, developed in the Python language, made it possible to evaluate the interference resistance of the communication channel under conditions of interference when using the developed method of error detection and correction. The software model of the communication channel was tested in an automated mode. According to the evaluation results, it can be stated that even with an eight-fold transmission error within one byte of the information block, fatal errors do not occur. This is evidenced by the absence of errors when transferring 24 bytes. At the same time, 128 information bits out of 192 transmitted were corrected.

Conflict of interest

The authors declare that they have no conflicts of interest with respect to this re-study, whether financial, personal, authorship, or otherwise, that could affect the study and its results presented in this article.

Financing

The study was performed without financial support.

Data availability

The manuscript contains data included as electronic supplementary material.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the presented work.

References

1. Internet Of Things (IoT). Available at: https://www.gartner.com/en/information-technology/glossary/internet-of-things
2. IoT Platforms. Available at: https://www.gartner.com/en/information-technology/glossary/iot-platforms
3. Shannon, C. E. (1948). A Mathematical Theory of Communication. Bell System Technical Journal, 27 (4), 623–656. https://doi.org/10.1002/j.1538-7305.1948.tb00917.x
4. Huffman, W. C., Pless, V. (2003). Fundamentals of Error-Correcting Codes. Cambridge University Press. https://doi.org/10.1017/cbo9780511807077
5. Subhasri, G., Radha, N. (2019). VLSI design of Parity check Code with Hamming Code for Error Detection and Correction. 2019 International Conference on Intelligent Computing and Control Systems (ICCS). https://doi.org/10.1109/iccs45141.2019.9065790
6. Tolentino, L. K. S., Valenzuela, I. C., Juan, R. O. S. (2019). Overhead Interspersing of Redundancy Bits Reduction Algorithm by Enhanced Error Detection Correction Code. Journal of Engineering Science and Technology Review, 12 (2), 34–39. https://doi.org/10.25103/jestr.122.05
7. Chen, Z., Zhao, Y., Lu, J., Liang, B., Chen, X., Li, C. (2022). TECED: A Two-Dimensional Error-Correction Codes Based Energy-Efficiency SRAM Design. Electronics, 11 (10), 1638. https://doi.org/10.3390/electronics11101638
8. Tolentino, L. K., Padilla, M. V., Serfa Juan, R. (2018). FPGA-based redundancy bits reduction algorithm using the enhanced error detection correction code. International Journal of Engineering & Technology, 7 (3), 1008. https://doi.org/10.14419/ijet.v7i3.12681
9. Koppala, N., Subhas, C. (2022). Low overhead optimal parity codes. TELKOMNIKA (Telecommunication Computing Electronics and Control), 20 (3), 501. https://doi.org/10.12928/telkomnika.v20i3.23301
10. Toghuj, W. (2020). Modifying Hamming code and using the replication method to protect memory against triple soft errors. TELKOMNIKA (Telecommunication Computing Electronics and Control), 18 (5), 2533. https://doi.org/10.12928/telkomnika.v18i5.13345
11. Saiz-Adalid, L.-J., Gil, P., Ruiz, J.-C., Gracia-Moran, J., Gil-Tomas, D., Baraza-Calvo, J.-C. (2016). Ultrafast Error Correction Codes for Double Error Detection/Correction. 2016 12th European Dependable Computing Conference (EDCC). https://doi.org/10.1109/edcc.2016.28
12. Rurik, W., Mazumdar, A. (2016). Hamming codes as error-reducing codes. 2016 IEEE Information Theory Workshop (ITW). https://doi.org/10.1109/itw.2016.7606865
13. Moon, T. K. (2005). Error correction coding: mathematical methods and algorithms. John Wiley & Sons. https://doi.org/10.1002/0471739219
14. Kadel, R., Paudel, K., Guruge, D. B., Halder, S. J. (2020). Opportunities and Challenges for Error Control Schemes for Wireless Sensor Networks: A Review. Electronics, 9 (3), 504. https://doi.org/10.3390/electronics9030504
15. Bettayeb, M., Ghunaim, S., Mohamed, N., Nasir, Q. (2019). Error Correction Codes in Wireless Sensor Networks: A Systematic Literature Review. 2019 International Conference on Communications, Signal Processing, and Their Applications (ICCSPA). https://doi.org/10.1109/iccspa.2019.8713725
16. Sridevi, N., Jamal, K., Mannem, K. (2021). Implementation of Error Correction Techniques in Memory Applications. 2021 5th International Conference on Computing Methodologies and Communication (ICCMC). https://doi.org/10.1109/iccmc51019.2021.9418432
17. Clark, G. C., Cain, J. B. (1981). Error-Correction Coding for Digital Communications. Springer US. https://doi.org/10.1007/978-1-4899-2174-1
18. Hamming, R. W. (1950). Error Detecting and Error Correcting Codes. Bell System Technical Journal, 29 (2), 147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x