MODELING AND INVESTIGATION OF NOISE IMMUNITY REED-SOLOMON CODE TO PROVIDE INCREASED IMMUNITY IN DVB-T2 STANDARD DIGITAL TELEVISION SYSTEM

A. A. Yarmukhamedov, A.B. Jabborov. Tashkent State Technical University, 100095, Tashkent city, Republic of Uzbekistan

Abstract

In this paper, the noise-correcting Reed-Solomon code is investigated and modeled to provide an increase in noise immunity in a digital television system of the DVB-T2 standard. Theoretical and practical algorithm of functioning of the Reed-Solomon noise-correcting codec has been studied. The error-correcting Reed-Solomon code in the DVB-T2 standard digital television system with the developed interactive computer simulation program in the Matlab/Simulink programming language, the dependence of the number of errors on the error probability for the noise-correcting Reed-Solomon code is modeled and investigated.

Keywords: Reed-Solomon code, error-correcting code, DVB-T2 standard, encoder and decoder, noise immunity, interference, noise, error probability, syndrome, code word length.

When receiving code packets (digital television signal of the DVB-T2 standard), an erroneous signal reception is formed on the receiver due to the influence of various interference and noise or interference on the communication channel.

Channel coding and decoding uses an outer Reed-Solomon code to detect and correct errors in digital television broadcasting.

The Reed-Solomon code is a linear block code and is a subset of BCH codes. Reed-Solomon codes are non-binary block cyclic codes that allow you to correct errors in data blocks. They are non-binary, because the elements of the code word are not bits, but bytes. This code allows you to correct up to 8 erroneously received bytes in each transport packet.

We analyze the Reed-Solomon encoder algorithm, the Reed-Solomon encoder receives a block of digital streams and adds additional "redundant" bits to them to ensure the correction of received errors and restore the transmitted code packets using the Reed-Solomon decoder. Generated number and types of errors that can be recovered, depending on the properties of the Reed-Solomon encoder.

The Reed-Solomon code is specified as RS (n, k) s - bit characters. This means that the encoder takes k information symbols in s-bits each and adds parity symbols to form an n symbol code word [1].

The coding process. To generate the Reed-Solomon code, arithmetic with polynomials in Galois fields and a polynomial generator are required, for example, as a result of generation, you can get the following:

$$(a^{1}-x)^{*}(a^{2}-x)^{*}(2^{3}-x)^{*}(a^{4}-x),$$
 (1)

(1) the expression can be rewritten as follows:

$$(2^{1}-x)^{*}(2^{2}-x)^{*}(2^{3}-x)^{*}(2^{4}-x)$$
 (2)

Where: a is a primitive field member (usually 2 is chosen) and x is the number of redundant characters.

The Reed-Solomon decoder can correct up to t symbols that contain errors in the code word, where 2t=n-k. The diagram below shows a typical Reed-Solomon code word (Figure 1): [1].

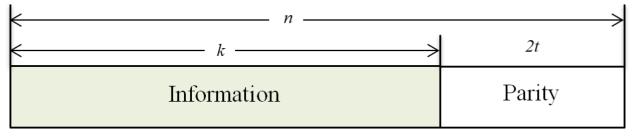


Fig.1. Code word for RS code

Decoding process. According to the functioning algorithm, the Reed-Solomon code during decoding can correct errors and restore lost everyday characters. The Reed-Solomon decoder can correct errors up to t and recover losses up to 2t. The generated loss of everyday symbols may be captured by the demodulator.

To ensure the decoding of received code packets, you can choose three decoding options [1]: 1. If 2s+r < 2t (s errors, r losses), then the original transmitted code word will always be recovered.

2. The decoder detects a situation where it cannot recover the original code word.

3. The decoder incorrectly decodes and incorrectly reconstructs the code word without any indication of this fact.

The probability of each of these options depends on the type of Reed Solomon code used, as well as on the number and distribution of errors [1].

To study and simulate the noise-correcting Reed-Solomon code in a digital television system of the DVB-T2 standard, an interactive computer simulation program was developed in the Matlab/Simulink programming language. The program is designed to simulate and calculate the dependence of the number of errors on the error probability for the error-correcting Reed-Solomon code.

To study and simulate the noise-correcting Reed-Solomon code in a digital television system of the DVB-T2 standard using the Matlab / Simulink software environment, we use the following parameters (Fig. 2,3,4,5,6) and set the block characteristics for the code (128.98) digital television system of the DVB-T2 standard (table 1).

N⁰	Parameter name	Parameter value	
1.	Frequency range:	(474-858) MHz with 8MHz	
	DMV	channel bandwidth	
2.	Modulation type	16-QAM	
3.	Constellation position	16	
4.	Bandwidth	8 MHz	
5.	Error-correcting code	Reed-Solomon	
6.	Code word length	128	
7.	Message length	98	

Table 1

Below are the parameters of the model (Fig.2,3,4,5,6).

On (Fig.7,8,9) the studied scheme of the model of the noise-correcting Reed-Solomon code in the digital television system of the DVB-T2 standard is shown. The model scheme under study consists of the following blocks: Bernoulli Binary Generator (generator), RS Encoder (encoder), Binary Symmetric Channel (transmission channel), RS Decoder (decoder), Error Rate Calculation (error analyzer) and Display.

Source Block Parameters: Bernoulli Binary Generator					
Bernoulli Binary Genera	ator				
Generate random Bern	oulli distributed binary numbers.				
Source code					
Parameters					
Probability of zero:	0.2				
Source of initial seed: Parameter					
Initial seed:	Initial seed: 61				
Sample time:	1				
Samples per frame: 784					
Output data type: volean v					
Simulate using: Interpreted execution					
OK Cancel Help Apply					

Fig.2. Parameters of Bernoulli Binary Generator

Binary-Input RS Encoder (mask) (link) Encode the message in the input vector using an (N,K) Reed-Solomon encoder with the narrow-sense generator polynomial. This block accepts a column vector input signal with an integer multiple of K*ceil(log2(N+1)) bits. Each group of K*ceil(log2(N+1)) input bits represents one message word to be encoded.	Binary-Input RS Encoder (mask) (link) Encode the message in the input vector using an (N,K) Reed-Solomon encoder with the narrow-sense generator polynomial. This block accepts column vector input signal with an integer multiple of K*ceil(log2(N+1)) bits. Each group of K*ceil(log2(N+1)) input bits represents one message word to be encoded. Shorten the code by setting the shortened message length parameter S. I this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: 128 Message length K: 98 □ Specify shortened message length	Binary-Input RS Encoder (mask) (link) Encode the message in the input vector using an (N,K) Reed-Solomon encoder with the narrow-sense generator polynomial. This block accepts a column vector input signal with an integer multiple of K*ceil(log2(N+1)) bits. Each group of K*ceil(log2(N+1)) input bits represents one message word to be encoded. Shorten the code by setting the shortened message length parameter S. In this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: 28 Specify shortened message length Specify shortened message length Specify generator polynomial Specify primitive polynomial Puncture code			
Encode the message in the input vector using an (N,K) Reed-Solomon encoder with the narrow-sense generator polynomial. This block accepts a column vector input signal with an integer multiple of K*ceil(log2(N+1)) bits. Each group of K*ceil(log2(N+1)) input bits represents one message word to be encoded. Shorten the code by setting the shortened message length parameter S. In this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: [128] Message length K: [98] [Specify shortened message length [Specify generator polynomial]	Encode the message in the input vector using an (N,K) Reed-Solomon encoder with the narrow-sense generator polynomial. This block accepts column vector input signal with an integer multiple of K*ceil(log2(N+1)) bits. Each group of K*ceil(log2(N+1)) input bits represents one message word to be encoded. Shorten the code by setting the shortened message length parameter S. I this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: [126] Message length K: [98] [] Specify shortened message length	Encode the message in the input vector using an (N,K) Reed-Solomon encoder with the narrow-sense generator polynomial. This block accepts a column vector input signal with an integer multiple of K*ceil(log2(N+1)) bits. Each group of K*ceil(log2(N+1)) input bits represents one message word to be encoded. Shorten the code by setting the shortened message length parameter S. In this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: 126 Message length K: 98 Specify shortened message length Specify generator polynomial Specify primitive polynomial Puncture code	🚹 Function Block Param	neters: Binary-Input RS Encoder	×
encoder with the narrow-sense generator polynomial. This block accepts a column vector input signal with an integer multiple of K*ceil(log2(N+1)) bits. Each group of K*ceil(log2(N+1)) input bits represents one message word to be encoded. Shorten the code by setting the shortened message length parameter S. In this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: [126] Message length K: [98] [Specify shortened message length [Specify generator polynomial [Specify primitive polynomial	encoder with the narrow-sense generator polynomial. This block accepts column vector input signal with an integer multiple of K*ceil(log2(N+1)) bits. Each group of K*ceil(log2(N+1)) input bits represents one message word to be encoded. Shorten the code by setting the shortened message length parameter S. I this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: 128 Message length K: 98 Specify shortened message length	encoder with the narrow-sense generator polynomial. This block accepts a column vector input signal with an integer multiple of K*ceil(log2(N+1)) bits. Each group of K*ceil(log2(N+1)) input bits represents one message word to be encoded. Shorten the code by setting the shortened message length parameter S. In this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: 128 Message length K: 98 Specify shortened message length Specify generator polynomial Specify primitive polynomial Puncture code	Binary-Input RS Encode	er (mask) (link)	
this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: 128 Message length K: 98 Specify shortened message length Specify generator polynomial Specify primitive polynomial	this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: 128 Message length K: 98 Specify shortened message length	this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code. Parameters Codeword length N: 128 Message length K: 98 Specify shortened message length Specify generator polynomial Specify primitive polynomial Puncture code	encoder with the narro column vector input sig bits. Each group of K*	w-sense generator polynomial mal with an integer multiple of	. This block accepts a K*ceil(log2(N+1))
Codeword length N: Codeword length N:	Codeword length N: Codeword length K:	Codeword length N:	this case, use full lengt	h N and K values to specify the	
128 Message length K: 98 Specify shortened message length Specify generator polynomial Specify primitive polynomial	128 Message length K: 98 Specify shortened message length	128 Message length K: 98 Specify shortened message length Specify generator polynomial Specify primitive polynomial Puncture code	Parameters		
Message length K: 98 Specify shortened message length Specify generator polynomial Specify primitive polynomial	Message length K: 98 Specify shortened message length	Message length K: 98 Specify shortened message length Specify generator polynomial Specify primitive polynomial Puncture code	Codeword length N:		
98 98 Specify shortened message length Specify generator polynomial Specify primitive polynomial	98 Specify shortened message length	98 98 Specify shortened message length Specify generator polynomial Specify primitive polynomial Puncture code	128		
Specify shortened message length Specify generator polynomial Specify primitive polynomial	Specify shortened message length	Specify shortened message length Specify generator polynomial Specify primitive polynomial Puncture code	Message length K:		
Specify generator polynomial Specify primitive polynomial		Specify generator polynomial Specify primitive polynomial Puncture code	98		
Specify primitive polynomial	Specify generator polynomial	Specify primitive polynomial Puncture code	Specify shortened n	nessage length	
			Specify generator p	olynomial	
Puncture code	Specify primitive polynomial		Specify primitive po	lynomial	
	Puncture code	Output data type: Same as input	Puncture code		
Output data type: Same as input	Output data type: Same as input		Output data type: Sar	me as input	•
				OK Cancel	Help Apply

Fig.3. Parameters of Reed-Solomon Encoder

📔 Function Block P	arameters: Binary Symmetric Channel	×		
Binary Symmetric (Channel (mask) (link)			
Add binary errors t	o the input signal.			
Parameters				
Error probability:				
0.2				
Initial seed:				
71				
☑ Output error vector				
Output data type:	boolean	•		
0	OK Cancel Help	Apply		

Fig.4. Parameters of Binary Symmetric Channel

👔 Function Block Parameters: Binary-Output RS Decoder1 🛛 🗙				
Binary-Output RS Decoder (mask) (link)				
Attempt to decode the input received signal using an (N,K) Reed-Solomon decoder with the narrow-sense generator polynomial. This block accepts a column vector input signal with an integer multiple of N*ceil(log2(N+1)) bits. Each group of N*ceil(log2(N+1)) input bits represents one received word to be decoded.				
Shorten the code by setting the shortened message length parameter S. In this case, use full length N and K values to specify the (N, K) code that is shortened to an (N - K + S, S) code.				
Parameters				
Codeword length N:				
128				
Message length K:				
98				
Specify shortened message length				
Specify generator polynomial				
Specify primitive polynomial				
Punctured code				
Enable erasures input port				
Output number of corrected errors				
Output data type: Same as input 👻				
OK Cancel Help Apply				



 Function Block Parameters: Error Rate Calculation1 × Error Rate Calculation (mask) (link) Compute the error rate of the received data by comparing it to a delayed version of the transmitted data. The block output is a three-element vector consisting of the error rate, followed by the number of errors detected and the total number of symbols compared. This vector can be sent to either the workspace or an output port. The delays are specified in number of samples, regardless of whether the input is a scalar or a vector. The inputs to the 'Tx' and 'Rx' ports must be scalars or column vectors. The 'Stop simulation' option stops the simulation upon detecting a target number of errors or a maximum number of symbols, whichever comes first. Parameters Receive delay: 0 Computation delay: 0 Computation mode: Entire frame Output data: Port Reset port Stop simulation OK Cancel Help Apply 				
Compute the error rate of the received data by comparing it to a delayed version of the transmitted data. The block output is a three-element vector consisting of the error rate, followed by the number of errors detected and the total number of symbols compared. This vector can be sent to either the workspace or an output port. The delays are specified in number of samples, regardless of whether the input is a scalar or a vector. The inputs to the 'Tx' and 'Rx' ports must be scalars or column vectors. The 'Stop simulation' option stops the simulation upon detecting a target number of errors or a maximum number of symbols, whichever comes first. Parameters Receive delay: 0 Computation delay: 0 Computation mode: Entire frame Output data: Port Computation	🔁 Function Block Parameters: Error Rate Calculation1	\times		
version of the transmitted data. The block output is a three-element vector consisting of the error rate, followed by the number of errors detected and the total number of symbols compared. This vector can be sent to either the workspace or an output port. The delays are specified in number of samples, regardless of whether the input is a scalar or a vector. The inputs to the 'Tx' and 'Rx' ports must be scalars or column vectors. The 'Stop simulation' option stops the simulation upon detecting a target number of errors or a maximum number of symbols, whichever comes first. Parameters Receive delay: 0 Computation delay: 0 Computation mode: Entire frame Output data: Port Stop simulation	Error Rate Calculation (mask) (link)			
input is a scalar or a vector. The inputs to the 'Tx' and 'Rx' ports must be scalars or column vectors. The 'Stop simulation' option stops the simulation upon detecting a target number of errors or a maximum number of symbols, whichever comes first. Parameters Receive delay: 0 Computation delay: 0 Computation mode: Entire frame Output data: Port Reset port Stop simulation	version of the transmitted data. The block output is a three-element vector consisting of the error rate, followed by the number of errors detected and the total number of symbols compared. This vector can be sent to either			
number of errors or a maximum number of symbols, whichever comes first. Parameters Receive delay: 0 Computation delay: 0 Computation mode: Entire frame Output data: Port Reset port Stop simulation	input is a scalar or a vector. The inputs to the 'Tx' and 'Rx' ports must be			
Receive delay: 0 Computation delay: 0 Computation mode: Entire frame Output data: Port Reset port Stop simulation	number of errors or a maximum number of symbols, whichever come			
0 Computation delay: 0 Computation mode: Entire frame Output data: Port Reset port Stop simulation	Parameters			
Computation delay: 0 0 Computation mode: Entire frame Output data: Port Reset port Stop simulation	Receive delay:			
0 Computation mode: Entire frame Output data: Port Reset port Stop simulation	0			
Computation mode: Entire frame Output data: Port Reset port Stop simulation	Computation delay:			
Output data: Port Reset port Stop simulation	0			
Reset port Stop simulation	Computation mode: Entire frame	•		
Stop simulation	Output data: Port 🔹			
	Reset port			
OK Cancel Help Apply	Stop simulation			
OK Cancel Help Apply				
	OK Cancel Help A	pply		

Fig.6. Parameters of Error Rate Calculation

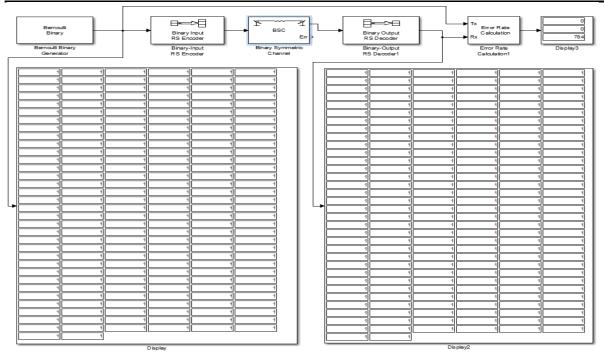


Fig.7. Scheme of the model of the error-correcting Reed-Solomon code in a digital television system of the DVB-T2 standard (error probability is 0)The displays of the model demonstrate: a transmission line using the Reed-Solomon code. Input combination, encoded sequence, output combination, errors (probability of errors is

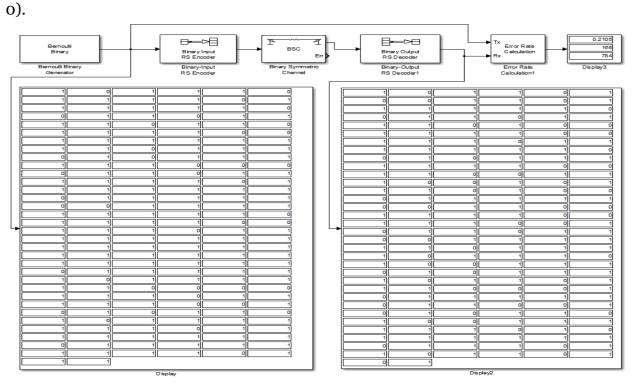


Fig.8. Scheme of the model of the error-correcting Reed-Solomon code in the digital television system of the DVB-T2 standard (error probability is 0.2)

102 | Page

Volume 14, March, 2023

The displays of the model demonstrate: a transmission line using the Reed-Solomon code. Input combination, encoded sequence, output combination, errors (error probability is 0.2).

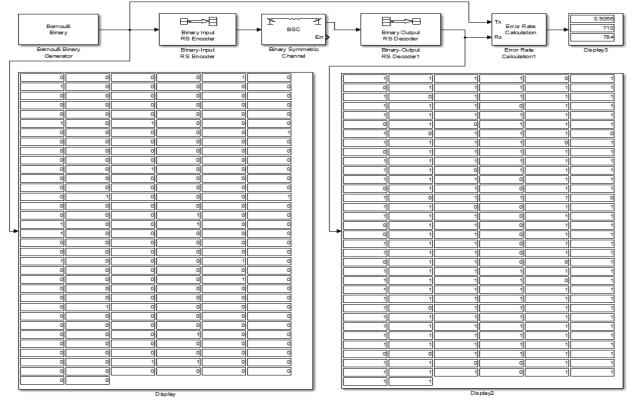


Fig.9. Scheme of the model of the error-correcting Reed-Solomon code in the digital television system of the DVB-T2 standard (error probability is 0.9)

The displays of the model demonstrate: a transmission line using the Reed-Solomon code. Input combination, encoded sequence, output combination, errors (error probability is 0.9). The simulation results and calculated values are summarized in Table 2.

		-		
Nº	Probability of errors	Frequency error	Number of detected errors	Total number of characters compared
1.	0	0	0	784
2.	0,1	0,1135	89	784
3.	0,2	0,2115	165	784
4.	0,3	0,3214	252	784
5.	0,4	0,4158	326	784
6.	0,5	0,5179	406	784
7.	0,6	0,5995	470	784
8.	0,7	0,7003	549	784
9.	0,8	0,7959	624	784
10.	0,9	0,9056	710	784
11.	1	1	784	784

Table 2.

A study was made using the model (Fig. 7,8.9) of the dependence of the number of errors on the error probability for the noise-correcting Reed-Solomon code in the digital television system of the DVB-T2 standard.

In Figure 10, the simulation results are presented: the dependence of the number of errors (OY) on the probability of error (OX) for

30 25 Number of detected errors 20 15 10 5 0 0 0,2 0,4 0,6 8,0 1 1,2 **Probability of errors**

Fig.10. Graph of the number of errors (OY) against the probability of error (OX) for the Reed-Solomon code (128.98)

The results of the study and modeling show that in channel coding and decoding, an external Reed-Solomon code is used to detect and correct errors in digital television broadcasting.

In practice, in a digital television system, the Reed-Solomon code is used for external coding. The Reed-Solomon code is a linear block code and is a subset of BCH codes. Reed-Solomon codes are non-binary block cyclic codes that allow you to correct errors in data blocks. The dependence of the number of errors on the error probability for the error-correcting Reed-Solomon code was produced and modeled, the results show that such a code provides protection of information from errors, the code allows you to correct up to 8 bytes received with errors in each transport packet. The lower the probability of the influence of interference on the communication channel, the less will be the erroneous reception of binary symbols.

Reed – Solomon code (128.98).

REFERENCES

1. Golikov A.M. Modulation, coding and modeling in telecommunication systems. Theory and practice: textbook - Tomsk: Tomsk. state un-t control systems. and radio electronics, 2016. - 516 p.: ill. - (Educational literature for universities).

2. EN 301 210: "Digital Video Broadcasting (DVB), Framing structure, channel coding and modulation for Digital Satellite News Gathering (SNG) and other contribution applications by satellite".

3. O. B. Grafov, S. I. Egorov, and V. S. Titov, "Soft decoding of Reed-Olomon codes," Izv. SWSU. Ser. "Management, computer technology, informatics" .. 2012. No. Part 1.S. 17-23.

4. S. I. Egorov, "An algorithm for decoding Reed-Solomon codes that corrects up to n-k errors in a code word," Tr. RNTORES them. A. S. Popova. Ser. "Digital signal processing and its applications". M., 2009. Issue. XI-1. pp. 27-30.

5. Lipnitsky V.A., Konopelko V.K. Norm decoding of error-correcting codes and algebraic equations. Mn.: BSU Publishing Center, 2007.

6. Werner R. Fundamentals of coding. Moscow: Technosphere. 2006.

7. Lipnitsky V.A., [et al.] Applied mathematics and the theory of norms of syndromes: methodical. allowance. Minsk: Ministry of Education of the Republic of Belarus, BSUIR. 2011.

8. Lipnitsky, V.A. Oleksyuk A.O.// Reports of BSUIR. 2015. No. 3 (89). pp.117–123.

9. M. Werner. Fundamentals of Coding (World of Programming) - 2004.

10. R. Morelos-Zaragoza. The art of error-correcting coding (World of Communications) - 2006.

11. Prokis J. Digital communication: textbook. M., 2000.

12. Dyakonov V.P. MATLAB. Complete tutorial. – M.: DMK Press, 2012. –

768 p.: ill.

13. G.V. Mamchev Theory and practice of terrestrial digital television broadcasting: Textbook / SibGUTI - Novosibirsk, 2010. 340 p.

14. Shakhnovich, I.V. Modern technologies of wireless communication./ Shakhnovich I.V. // Moscow: Technosphere - 2006. - 288 p.

15. Sklyar, B. Digital communication. Theoretical foundations and practical application / B. Sklyar; - M .: ed. house "Williams", 2007. - 1104 p.

16. Lipnitsky V.A., Oleksyuk A.O. // Reports of BSUIR.2014. No. 8 (86). pp. 72–78.

17. Oleksyuk A.O., Lipnitsky V.A. Error correction device multiplicity four No. A20130054 dated January 16, 2013

AUTHOR'S BIOGRAPHY

Yarmukhamedov Alisher Akbarovich - Head of the department: Candidate of Technical Sciences, Associate Professor of the Department of radio Engineering devices and systems of Tashkent State Technical University, ayarmuhamedov@mail.ru

Jabborov Alibek Botirkul ogli -senior lecturer of the Department of radio Engineering devices and systems of Tashkent State Technical University, jabbborovalibek92@gmail.com