Matlab-Based Interactive Simulation Model of Reed-Solomon Encoders Applied in the Educational Process

Yuksel Aliev[1], Adriana Borodzhieva[2], Galina Ivanova[3]

Abstract

Today the teaching staff in the universities around the world discusses the possibility of using digital tools to deepen the connection with students. The main methods are face-to-face teaching, online teaching, independent learning, assessment and feedback, using information and communication technologies to enhance teaching and learning. Different types of channel codes – linear block, cyclic, BCH, convolutional and Reed-Solomon codes – are studied in the course "Coding in Telecommunication Systems" in the University of Ruse. Reed-Solomon codes are hardly perceived by students because of the need to possess complex mathematical knowledge in the area of the finite Galois fields. This necessitates the development of a tool for an attractive way to present the process of the systematic encoding using a linear feedback shift register instead of using the division of polynomials in Galois fields. The paper presents MATLAB-based interactive simulation model with a graphical user interface, demonstrating the process of the systematic encoding a message using (7,3) Reed-Solomon code, based on Galois field GF(2^3), generated by primitive irreducible polynomials, based on a linear feedback shift register (LFSR).The model is composed of three parts: an input sequence register, an encoder, and an output sequence register. The model can work in three modes: Step by Step, Automatic and Manual. In Step by Step mode, the micro operations required for the encoding are performed after pressing the key "Next", the user is able to monitor and learn every action in the encoding process. In Automatic mode, the entire encoding process takes place without interruption from the beginning to the end. In Manual mode, the user selects micro operations on his/her own and the system monitors the sequence of actions and in case of wrong choice, displays an error message and prevents the next action until the correct micro operation is performed. In this mode, the system enables the user to be evaluated by points for each correct answer. The application is used in the courses “Coding in Telecommunication Systems” and “Reliability and Diagnostics of Computer Systems" by students at the University of Ruse “Angel Kanchev”.

Keywords: *Teaching and learning, Reed-Solomon codes, Interactive simulation model, Encoding, MATLAB.*

1. Introduction

Today the teaching staff in universities discusses the possibility of how digital tools can help deepen the real-world connection with students. Many universities – Oxford, Cambridge, London’s Global University – develop their digital education strategies. The strategies establish a framework for engagement and creativity in this important area of education and innovation, ensuring the universities to become world leaders in the integration of education and research and remain premier institutions for teaching by adopting the most exciting teaching innovations made possible by digital technologies [1,2,3]. The main ideas: **1.**Face-to-face teaching; **2.**Online teaching, allowing to expand the teaching beyond the constraints of the classroom; **3.**Assessment and feedback; **4.**Independent learning that helps the students learn how to learn; **5.**Outreach and widening access; **6.**Public engagement; **7.**Using technology to enhance teaching/learning.

The digital generation, called generation of six screens – TV, computer, laptop, tablet, fablet, smartphone, cannot and should not be taught as children’ parents. Replacing the blackboard with white one, and the chalks with markers, does not change things. It is necessary, through the massive and effective use of ICT-based innovative educational technologies and didactic models, to adapt the education system to the digital generation, and by introducing the research approach into the educational process, to reorient it from mechanical learning to rediscovery of knowledge and developing skills.

The paper presents the effective use of ICT-based educational technologies for studying Reed-Solomon (R-S) codes in the course “Coding in Telecommunication Systems” in the University of Ruse.

[1] University of Ruse “Angel Kanchev”, Bulgaria
[2] University of Ruse “Angel Kanchev”, Bulgaria
[3] University of Ruse “Angel Kanchev”, Bulgaria

R-S codes are hardly perceived by students because of the need to possess complex mathematical knowledge about finite Galois fields. This necessitates developing tools for attractive presenting the process of systematic encoding using linear feedback shift register instead of polynomials division.

2. Reed-Solomon codes

R-S codes are *nonbinary cyclic* codes with symbols of m-bit sequences, where m is any positive integer with a value greater than 2. R-S (n,k) codes on m-bit symbols exist for all n and k for which $0<k<n<2^m+2$, where k is the number of data symbols being encoded, and n is the total number of code symbols in the encoded block. For the conventional R-S (n,k) code, $(n,k)=(2^m–1,2^m–1–2t)$, where t is the symbol-error correcting capability of the code, and $n–k=2t$ is the number of parity symbols. R-S codes achieve the *largest possible* code minimum distance for any linear code with the same encoder input and output block lengths. For R-S codes the code minimum distance is given by $d_{min}=n–k+1$. The code is capable of correcting any combination of t or fewer errors, where t is the largest integer not exceeding $(n–k)/2$. Correcting t symbol errors requires no more than $2t$ parity symbols. The decoder has $n–k$ redundant symbols, twice the amount of correctable errors. For each error, one redundant symbol is used to locate the error, and another one – to find its correct value. R-S codes are effective for *burst-error correction*, for channels with memory and channels where the set of input symbols is large. In order to understand the encoding and decoding principles of non-binary R-S codes, it is necessary to venture into the area of finite fields known as *Galois Fields* (*GF*). For any prime number p there exists a finite field $GF(p)$ with p elements. It is possible to extend $GF(p)$ to a field of p^m elements, $GF(p^m)$, where m is a nonzero positive integer. Symbols from the extension field $GF(p^m)$ are used in constructing R-S codes. More details about the finite fields, the primitive polynomial used to define $GF(p^m)$ and a simple test to determine if a polynomial is primitive, and the rules for addition and multiplication in $GF(p^m)$, can be found in [4]. The generator is $\boldsymbol{g}(X)=g_0+g_1X+g_2X^2+\ldots+g_{2t–1}X^{2t–1}+X^{2t}$. The degree of the generator polynomial is equal to the number of parity symbols $2t$. The roots of $\boldsymbol{g}(X)$ are designated as $\alpha,\alpha^2,\ldots,\alpha^{2t}$. (7,3) double-symbol error correcting R-S code is considered as an example. The generator polynomial is described by $\boldsymbol{g}(X)=X^4–\alpha^3X^3+\alpha^0X^2–\alpha^1X+\alpha^3$ for the primitive polynomial for generating $GF(8)$ $f(X)=1+X+X^3$. Following the format of low to high order, and changing negative signs to positive, the generator will be $\boldsymbol{g}(X)=\alpha^3+\alpha^1X+\alpha^0X^2+\alpha^3X^3+X^4$. Since R-S codes are cyclic codes, encoding in systematic form is analogous to the binary encoding. Therefore $\boldsymbol{m}(X)$ is multiplied by $X^{n–k}$, thereby manipulating the message polynomial algebraically so that it is right-shifted $n–k$ positions. Next, $X^{n–k}\boldsymbol{m}(X)$ is divided by $\boldsymbol{g}(X)$ – $X^{n–k}\boldsymbol{m}(X)=\boldsymbol{q}(X)\boldsymbol{g}(X)+\boldsymbol{p}(X)$, where $\boldsymbol{q}(X)$ and $\boldsymbol{p}(X)$ are quotient and remainder polynomials. The remainder is the parity. The last equation is expressed as $\boldsymbol{p}(X)=X^{n–k}\boldsymbol{m}(X)$ modulo $\boldsymbol{g}(X)$. The resulting codeword is written as $\boldsymbol{U}(X)=\boldsymbol{p}(X)+X^{n–k}\boldsymbol{m}(X)$. These steps are demonstrated by encoding the three-symbol message 010-110-111 (α^1-α^3-α^5) with (7,3) R-S code with $\boldsymbol{g}(X)=\alpha^3+\alpha^1X+\alpha^0X^2+\alpha^3X^3+X^4$. First the message polynomial $\alpha^1+\alpha^3X+\alpha^5X^2$ is multiplied by $X^{n–k}=X^4$, yielding $\alpha^1X^4+\alpha^3X^5+\alpha^5X^6$. Next this upshifted message polynomial is divided by $\boldsymbol{g}(X)$. Polynomial division with nonbinary coefficients is more tedious than its binary counterpart, because the required operations of addition and multiplication must follow the rules in the addition and multiplication table. This polynomial division results in the remainder $\boldsymbol{p}(X)=\alpha^0+\alpha^2X+\alpha^4X^2+\alpha^6X^3$. More details are given (Fig.1) for better perception of material taught. The codeword polynomial will be $\boldsymbol{U}(X)=\alpha^0+\alpha^2X+\alpha^4X^2+\alpha^6X^3+\alpha^1X^4+\alpha^3X^5+\alpha^5X^6$ [4].

Using circuitry to encode a 3-symbol sequence in systematic form with (7,3) R-S code described by $\boldsymbol{g}(X)=\alpha^3+\alpha^1X+\alpha^0X^2+\alpha^3X^3+X^4$ requires the implementation of LFSR. The multiplier terms in Fig.2 (from left to right) correspond to the coefficients of $\boldsymbol{g}(X)$ (low to high order). This encoding process is the non-binary equivalent of cyclic encoding. Here, (7,3) R-S nonzero codewords are made up of $2^m–1=7$ symbols, and each symbol is made of $m=3$ bits. In this case the number of stages in the shift register is $n–k$. In the non-binary example each stage in the shift register holds a 3-bit symbol. The non-binary operation implemented by the encoder, forming codewords in a systematic format, proceeds in the same way as the binary one. The steps are: **1.**Switch 1 is closed during the first k clock cycles to allow shifting the message symbols into the $(n–k)$-stage shift register. **2.**Switch 2 is in the down position during the first k clock cycles in order to allow simultaneous transfer of the message symbols directly to an output register. **3.**After transfer of the k^{th} message symbol to the output register, switch 1 is opened and switch 2 is moved to the up position. **4.**The remaining $n–k$ clock cycles clear the parity symbols contained in the shift register by moving them to the output register. **5.**The total number of clock cycles is equal to n, and the contents of the output register is the codeword $\boldsymbol{p}(X)+X^{n–k}\boldsymbol{m}(X)$, where $\boldsymbol{p}(X)$ represents the parity symbols, and $\boldsymbol{m}(X)$ – the message symbols. The same symbol sequence 010-110-111 (α^1-α^3-α^5) is used, where the rightmost symbol is the earliest symbol, and the rightmost bit is the earliest bit. The operational steps during the first k=3 shifts of the encoding circuit are shown in Fig.3.

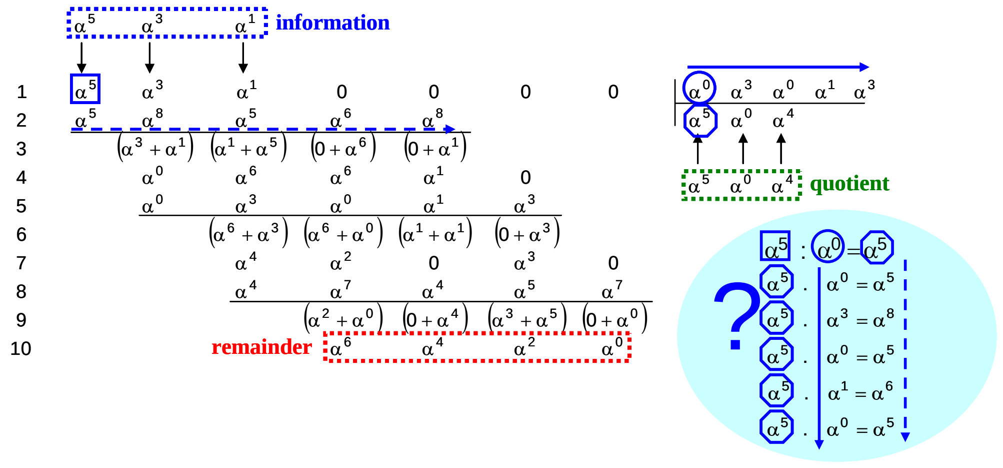


Fig.1. Encoding a message

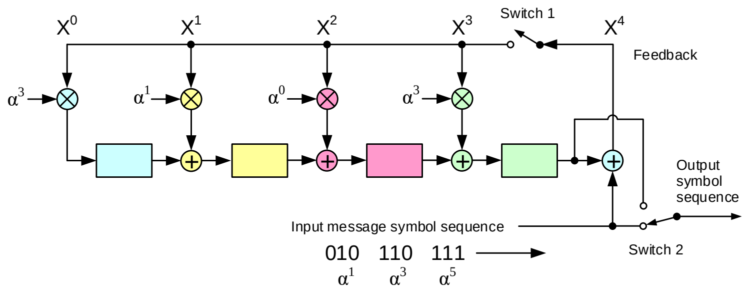
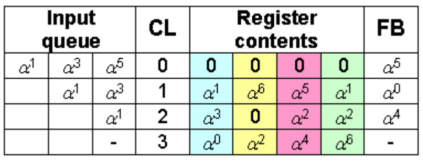


Fig.2. LFSR Encoder



Input queue			CL	Register contents				FB
α^1	α^3	α^5	0	0	0	0	0	α^5
	α^1	α^3	1	α^1	α^6	α^5	α^1	α^0
		α^1	2	α^3	0	α^2	α^2	α^4
		-	3	α^0	α^2	α^4	α^6	-

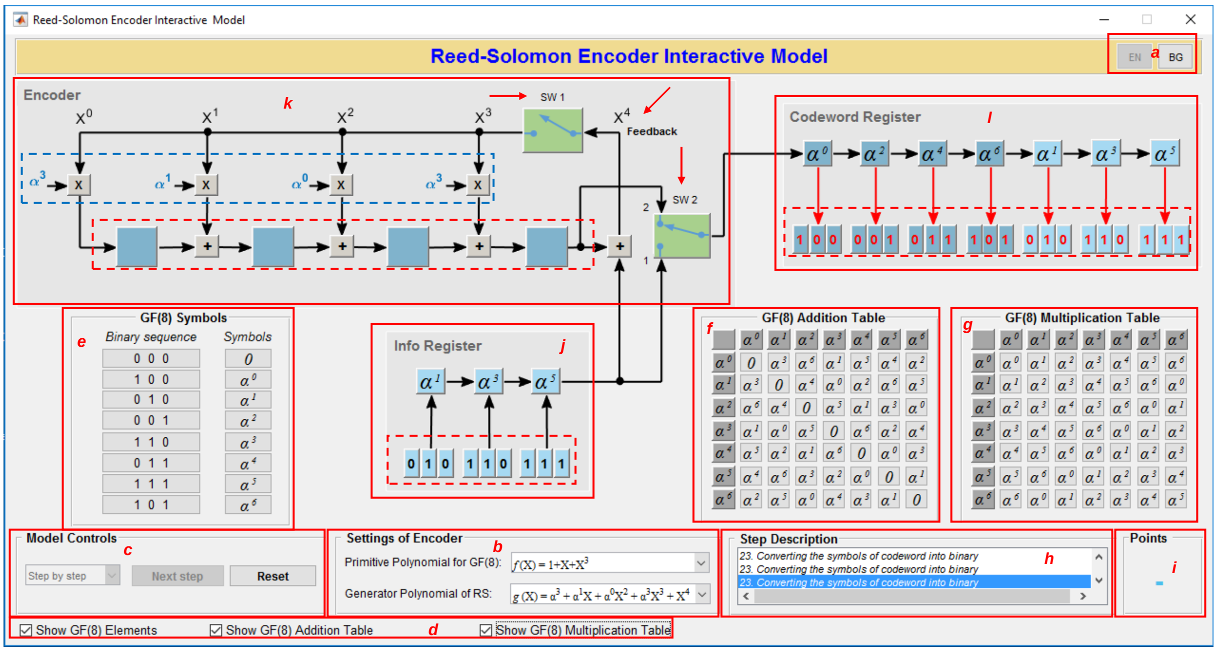
Fig. 3. Operational steps

After the third clock cycle, the register contents are the parity symbols, $\alpha^0,\alpha^2,\alpha^4,\alpha^6$. Then, switch 1 of the circuit is opened, switch 2 is toggled to the up position, and the parity symbols in the register are shifted to the output. Therefore the output codeword $\boldsymbol{U}(X)=\alpha^0+\alpha^2X+\alpha^4X^2+\alpha^6X^3+\alpha^1X^4+\alpha^3X^5+\alpha^5X^6$ (in polynomial form). The process of verifying the contents of the register at various clock cycles is more tedious than in the binary case. Here, the field elements must be added and multiplied by using the addition table and the multiplication table respectively.

3. Interactive simulation model implementing Reed-Solomon encoding

MATLAB-based interactive simulation model with graphical user interface (GUI) is developed and presented in the paper. It is compiled using MATLAB Application Compiler 6.4 (a collection of shared libraries and code that enables the execution of compiled and packaged MATLAB applications on systems without installed MATLAB). The application allows the choice of language (Fig.4,a) to visualize the process of systematic encoding a message using (7,3) R-S code, based on $GF(2^3)$, generated by primitive irreducible polynomials, by LFSR – available versions in English and Bulgarian.
By means of drop down menus located in the block "Settings of Encoder" (Fig.4,*b*) the primitive irreducible polynomial for generating *GF*(8) (Fig.5,*a*) or the generator polynomial (Fig.5,*b*) used for the encoding is chosen. For (7,3) R-S code considered, two primitive irreducible polynomial for generating *GF*(8) are available $f(X)=1+X+X^3$ and $f(X)=1+X^2+X^3$, and the corresponding generator polynomials are $\boldsymbol{g}(X)=\alpha^3+\alpha^1X+\alpha^0X^2+\alpha^3X^3+X^4$ and $\boldsymbol{g}(X)=\alpha^3+\alpha^0X+\alpha^3X^2+\alpha^2X^3+X^4$.
The student can select the mode of operation from drop-down menu in "Model controls" (Fig.4,*c*). The model can work in three modes: Step by Step, Automatic and Manual (Fig.5,*c*). In *Step by Step* mode, the micro-operations required for encoding are performed after pressing the key "Next" in "Model Controls", the user is able to monitor and learn every action in the encoding process. The description of the step is shown in "Step Description" block (Fig.4,*h*). In the interactive model, the corresponding actions are visualized in red to draw the attention of the learner. In *Automatic* mode, the encoding process takes place without interruption from beginning to end. This mode is applicable to checking the student's work, but also allows tracking the encoding process. In *Manual* mode, the user selects the micro-operations and the system monitors the sequence of actions and when wrong choice,

displays error messages and prevents the next action until the correct micro-operation is performed. In this mode, the system allows the user to be evaluated by points for each correct answer (Fig.4,*i*). Now the test module has not yet been developed.



Reed-Solomon Encoder Interactive Model

GF(8) Symbols

Binary sequence	Symbols
0 0 0	0
1 0 0	α^0
0 1 0	α^1
0 0 1	α^2
1 1 0	α^3
0 1 1	α^4
1 1 1	α^5
1 0 1	α^6

GF(8) Addition Table

	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^0	0	α^3	α^6	α^1	α^5	α^4	α^2
α^1	α^3	0	α^4	α^0	α^2	α^6	α^5
α^2	α^6	α^4	0	α^5	α^1	α^3	α^0
α^3	α^1	α^0	α^5	0	α^6	α^2	α^4
α^4	α^5	α^2	α^1	α^6	0	α^0	α^3
α^5	α^4	α^6	α^3	α^2	α^0	0	α^1
α^6	α^2	α^5	α^0	α^4	α^3	α^1	0

GF(8) Multiplication Table

	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^0	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^1	α^1	α^2	α^3	α^4	α^5	α^6	α^0
α^2	α^2	α^3	α^4	α^5	α^6	α^0	α^1
α^3	α^3	α^4	α^5	α^6	α^0	α^1	α^2
α^4	α^4	α^5	α^6	α^0	α^1	α^2	α^3
α^5	α^5	α^6	α^0	α^1	α^2	α^3	α^4
α^6	α^6	α^0	α^1	α^2	α^3	α^4	α^5

Model Controls: Step by step | Next step | Reset

Settings of Encoder
Primitive Polynomial for GF(8): $f(X) = 1+X+X^3$
Generator Polynomial of RS: $g(X) = \alpha^3 + \alpha^1X + \alpha^0X^2 + \alpha^3X^3 + X^4$

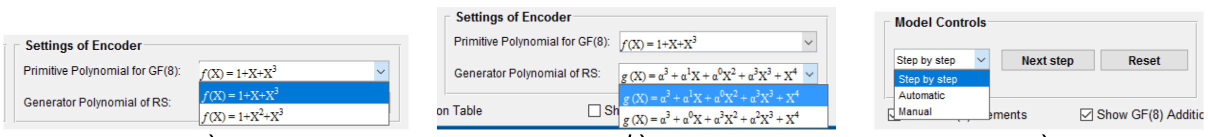
Step Description
23. Converting the symbols of codeword into binary
23. Converting the symbols of codeword into binary
23. Converting the symbols of codeword into binary

Points

☑ Show GF(8) Elements ☑ Show GF(8) Addition Table ☑ Show GF(8) Multiplication Table

Fig.4. Interactive simulation model for Reed-Solomon encoding

When the check boxes Show GF(8) Elements, Show GF(8) Addition Table and Show GF(8) Multiplication Table (Fig.4,*d*) are checked, the application allows visualizing the mapping field elements in terms of basis elements (Fig.4,*e*), addition table (Fig.4,*f*) and multiplication table (Fig.4,*g*) for *GF*(8) with $f(X)=1+X+X^3$. Their visualization assists the students in perception the process of systematic encoding using LFSR. At each step the student can trace the encoding process and verify the results of addition and multiplication being realized.
The model is composed of three parts: input sequence (info) register (Fig.4,*j*), encoder (Fig.4,*k*) and output sequence register (Fig.4,*l*).
After inserting the test message bits in Info Register (Fig.4,*j*, dotted line, by clicking on zeros for changing to 1), the student can see in detail the steps in the encoding process. The description of the steps is given in Table 1. At the end of the simulation, the corresponding encoded message is obtained in the output register (codeword register, Fig.4,*l*, dotted line).
The interactive simulation model can also be used in the polynomial division presented in Fig.1. For example, obtaining the coefficients in row 2 (Fig.1) can be traced by performing Step 5 (Fig.6, right to left). Likewise, row 5 and row 8 can be tracked by performing Step 10 and Step 15. Obtaining the coefficients in row 10 (Fig.1) can be traced by performing Step 17 (Fig.7). The values of the feedback are exactly the quotient in the polynomial division (Fig.1).


Settings of Encoder
Primitive Polynomial for GF(8): $f(X) = 1+X+X^3$
$f(X) = 1+X+X^3$
$f(X) = 1+X^2+X^3$
Generator Polynomial of RS:

Settings of Encoder
Primitive Polynomial for GF(8): $f(X) = 1+X+X^3$
Generator Polynomial of RS: $g(X) = \alpha^3 + \alpha^1X + \alpha^0X^2 + \alpha^3X^3 + X^4$
$g(X) = \alpha^3 + \alpha^1X + \alpha^0X^2 + \alpha^3X^3 + X^4$
$g(X) = \alpha^3 + \alpha^0X + \alpha^3X^2 + \alpha^2X^3 + X^4$

Model Controls
Step by step | Next step | Reset
Step by step
Automatic
Manual
☑ Show GF(8) Additic

a) b) c)

Fig.5. Options in testing the interactive module

Table 1. Step description (Fig.4,*h*)

№	Step Description
1/23	Converting binary info bits into GF(8) elements/the symbols of codeword into binary
2/18	Positioning SW1 – Close/Open. Positioning SW2 – position 1/position 2.
3/8/13	Calculating the value of Feedback (FB)
4/9/14/19…22	Shifting the codeword
5/10/15	Multiplying the coefficients of the generator polynomial with the feedback
6/11/16	Summing the results of multiplications with the values in the LFSR (Fig.6, Step 6)
7/12/17	Loading the new values in the LFSR (Fig.7, Step 17)

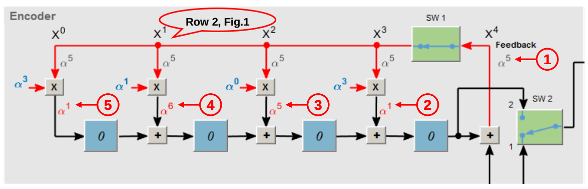


Fig.6. Step 6

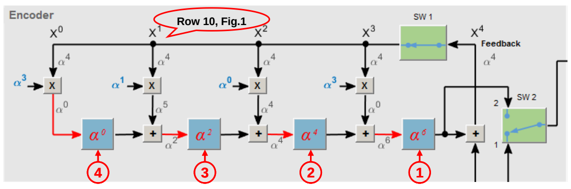


Fig.7. Step 17

4. Conclusions

The paper presents MATLAB-based interactive simulation model with GUI, demonstrating the process of systematic encoding a message using (7,3) R-S code, based on *GF*(2^3), by LFSR.The model is composed of three parts and can work in three modes, presented in details in the paper. The application is used in the course “Coding in Telecommunication Systems” by students at the University of Ruse.

The study was supported by contract of University of Ruse “Angel Kanchev”, № BG05M2OP001-2.009-0011-C01, “Support for the development of human resources for research and innovation at the University of Ruse “Angel Kanchev””. The project is funded with support from the Operational Program “Science and Education for Smart Growth 2014-2020” financed by the European Social Fund of the European Union.

References

[1] Digital Education Strategy 2016-2020, University of Oxford, April 2016, https://www.admin.ox.ac.uk/media/global/wwwadminoxacuk/localsites/educationcommittee/documents/notesofguidance/Digital_Education_Strategy_2016_-_2020-_final.pdf (January 2018).

[2] Digital Strategy for Education, University of Cambridge, Cambridge Centre for Teaching and Learning, 2016, https://www.cctl.cam.ac.uk/news/digital-strategy-education (January 2018).

[3] UCL Education Strategy 2016-21, London’s Global University, 2016, https://www.ucl.ac.uk/teaching-learning/education-strategy (January 2018).

[4] Sklar, B. “Digital Communications. Fundamentals and Applications.” Communications Engineering Services, California, 2001. http://userspages.uob.edu.bh/mangoud/mohab/Courses_files/sklar.pdf.