USING ASSEMBLER FOR MICROCONTROLLER STUDY ON ARDUINO-BASED PLATFORM

Aleksandr Vostrukhin[1], Elena Vakhtina[2], Sergey Bondar[2]
[1]Stavropol Technological Institute of Service, Russia; [2]Stavropol State Agrarian University, Russia
avostrukhin@yandex.ru, eavakhtina@yandex.ru, bond_sn@mail.ru

Abstract. Improving the information systems is a high-priority direction for the development of world science, engineering and technology. Graduates of engineering faculties could be involved in this trend in specific sectors of industry in the case they will master real-time systems programming in the process of education at higher school.These systems are built on the basis of miniature computers – microcontrollers implementing the basic functions of classic information systems. The purpose of this article is to show the didactic possibilities of using Arduino hardware platform together with the methods of programming in Assembly language. To achieve this goal a classic example of square wave generator programming on the Arduino hardware platform in professional development environment AVR Studio in Assembly language is considered. Experience of Arduino programming in high-level languages C, C++ is presented in engineering education. In this work we seek to actualize didactic capabilities of Assembler, which were not used in the educative process: accessibility of all elements of the microcontroller architecture that allows a student developing the program to implement the most efficient algorithms; ability to generate machine instructions and to control this process by a student; visibility of the microcontroller work, which increases the level of understanding of processes occurring in the system, and, consequently, the effectiveness of training. Assembler is the foundation in the field of microprocessor technique cognition through the study of its programming methods. Arduino hardware platform is made on the microcontrollers of AVR family. These microcontrollers are in the ratio of price – performance – power consumption occupy leading positions and are recognized as industry standard. The study of microcontrollers, which is organized in a professional development environment by programming in Assembly language, is a variant of a practice-oriented cognitive technology in engineering education.

Keywords: algorithm, program, real-time information systems, experiment, generator, oscilloscope, AVR Studio

Introduction

Modern electronic devices are built on the basis of microcontrollers. A microcontroller is designed to implement the basic functions of real-time information systems: collection, processing, storage and transmission of information, as well as the formation of the control signals. The field-of-use microcontrollers are wider than microprocessors. They are used in control systems, ranging from household appliances and ending aircraft. A microcontroller contains on a single semiconductor chip almost all digital electronics devices, from the logic elements to processor. Each family of microcontrollers has its own Assembly language.

When programming in Assembly Language for a developer all the capabilities of the microcontroller architecture are accessed, and he can use them to implement the most efficient algorithms. High-level languages do not provide such an opportunity. When programming a microcontroller in high level language, such as C, it is not required to know its architecture. Assembler Language is the foundation in the cognition field of microprocessor technique through the study of software designing processes. Figuratively speaking, Assembler is the language of unlimited possibilities [1].

Arduino-based platform allows realizing a process of learning using the Assembly. Arduino is the most affordable budget solution, implemented at the popular family of AVR microcontrollers. While programming Arduino there are the most actual for information real-time systems languages C/C ++ and Assembler used [2]. Arduino contains an extensive assortment of modular hardware and software with an open source license, continues to develop and is supported by sufficient amount of literature [3-5]. Arduino is supported by: the widespread in universities Matlab tool [6]; AVRStudio7 software environment designed for the professional development of software application for 8-bit and 32-bit AVR microcontrollers [7]; the new operating system Windows 10 [8]. Briefly you can get acquainted with Arduino on the site https://www.arduino.cc.

The authors do not know works that discuss projects aimed at implementation of typical tasks of real-time information systems using Assembly on the Arduino-based platform. The purpose of the article is to consider a case that is most suitable for the initial stage of microcontroller study and show that hardware platform Arduino can be used to study microcontrollers on Assembler.

Materials and methods

We are implementing the square wave generator (hereinafter simply generator) on the Arduino-based hardware. A generator can be realized as hardware or software way. A hardware generator requires the using of a timer/counter (an inbuilt peripheral) of a microcontroller and is relatively complicated for an initial study of microcontrollers. We are realizing the programmable generator [9]. For programming in Assembly language we use the integrated development environment AVR Studio, which is distributed free and always available on Atmel's website. The actual version is AVR Studio 4.18.684.

A timing diagram of the generator signal is shown in Figure 1. The signal period is determined by the expression: $T_G = t_i + t_p$, where t_i and t_p, respectively, pulse (impulse) duration and pause duration. To simplify the task, we take $t_i = t_p$.

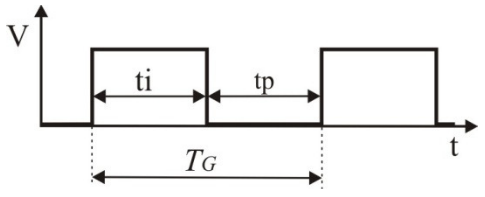


Fig. 1. **Generator signal, Timing Diagram**

The algorithm for the GENERATOR program is as follows:

Step 1: Start

Step 2: Output a high voltage level to the PD0 pin of the microcontroller, i.e. logical '1'.

Step 3: Form a time delay "DELAY" with duration t_i.

Step 4: Output a low voltage level to the microcontroller PD0 pin, i.e. logical '0'.

Step 5: Create a time delay "DELAY" with duration t_p and go to Step 2.

Step 6: Stop

A properly designed program should consist of separate functionally complete debugged modules – subroutines. Consider the subroutine DELAY, which is part of the program GENERATOR. The algorithm for the DELAY subroutine is shown in Figure 2.

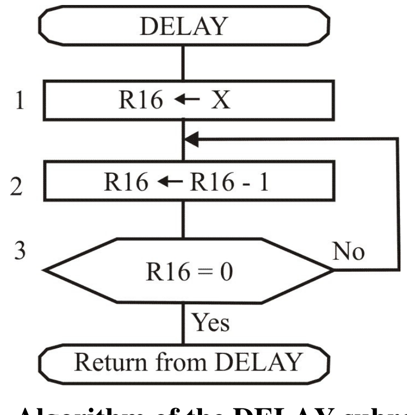


Fig. 2. **Algorithm of the DELAY subroutine**

The subroutine implementing the DELAY algorithm is presented in Figure 3.

```
DELAY:  ; The time delay subroutine
 LDI R16,5     ; Step 1. Load into the register R16 the constant 5 (X = 5)
m1: DEC R16    ; Step 2. Decrement the contents of register R16
 BRNE m1             ; Step 3. If not zero go to m1
 RET   ; Exit the DELAY subroutine
```

Fig. 3. **DELAY subroutine**

To debug the DELAY subroutine, a code fragment that simulates the main program is used. This code fragment is presented below in Figure 4.

```
m2:     RCALL DELAY    ; Call subroutine DELAY
        RJMP   m2      ; Go to m2
```

Fig. 4. **Code fragment**

The DELAY subroutine is called from the main program by the RCALL DELAY command (the first line with the m2 mark). After execution of the DELAY subroutine, the control is transferred to the main program with the RET command (see the DELAY subroutine). Then the RJMP m2 command is executed – an unconditional jump to the m2 mark (the second line of the DELAY subroutine debug fragment).

To debug the DELAY subroutine, you can select the value of the constant *X* from the range: 1 ... 255, where 255 is the decimal equivalent of the maximum binary code that can be loaded into the 8-bit microcontroller register. In the considered case $X = 5$. We check the operability of the DELAY subroutine in the step-by-step debugging mode in AVR Studio environment. If the subroutine works, then we go to calculate the time delay of the specified duration, i.e. to the calculation of the constant *X* (see the algorithm presented in Figure 2).

For example, it is required to develop a generator that forms a rectangular signal with a frequency $f_G = 20$ kHz. To solve this task, it is necessary to calculate the delay time T_D, which is determined from the condition $T_D = t_i = t_p$. The generator period is defined by the expression: $T_G = 1 / f_G = 1/20000 = 50$ µs. As $T_G = t_i + t_p = 2T_D$, then $T_D = T_G / 2 = 25$ µs.

The time delay duration is mainly determined by the cycle amount that the microcontroller must perform when executing the DELAY subroutine. The cycle consists of two commands with the mnemonics DEC and BRNE (see the DELAY subroutine). The remaining commands with mnemonics LDI, RET and RCALL are executed by the microcontroller only once.

A microcontroller operates under the action of a clock generator. The number of clock cycles required for the microcontroller to execute various commands can be determined according to the Complete Instruction Set Summary [10]. The number of cycles needed for the microcontroller to execute RCALL, LDI and RET mnemonics is given by:

$$N_{RLR} = N_{RCALL} + N_{LDI} + N_{RET}, \tag{1}$$

where *NRCALL, NLDI and NRET* – number of clock cycles required for the microcontroller to execute *RCALL, LDI* and *RET* mnemonics*,* respectively.

Then according to expression (1): $N_{RLR} = 3 + 1 + 4 = 8$ cycles.

The number of cycles needed for the microcontroller to implement the time delay of a given duration is determined by the expression:

$$X = (T_D - T_{RLR})/T_{DB}, \tag{2}$$

where T_{RLR} – time required for the microcontroller to perform *RCALL*, *LDI* and *RET* mnemonics; µs;
T_{DB} – time required for the microcontroller to perform *DEC* and *BRNE* mnemonics, i.e. a single cycle, µs.

T_{RLR} time is determined from the expression:

$$T_{RLR} = T_c \cdot N_{RLR}, \tag{3}$$

where T_c – clock period of the microcontroller, determined from the expression $T_c = 1/fc$;
fc – clock frequency of the microcontroller, $f_c = 16$ MHz is specified in the description of the Arduino Uno hardware (https://www.arduino.cc).

Then $T_c = 1/16 \cdot 10^{-6} = 0.0625$ µs.

In accordance with the expression (3), $T_{RLR} = 0.0625 \cdot 8 = 0.5$ µs.

T_{DB} time is defined by the formula:

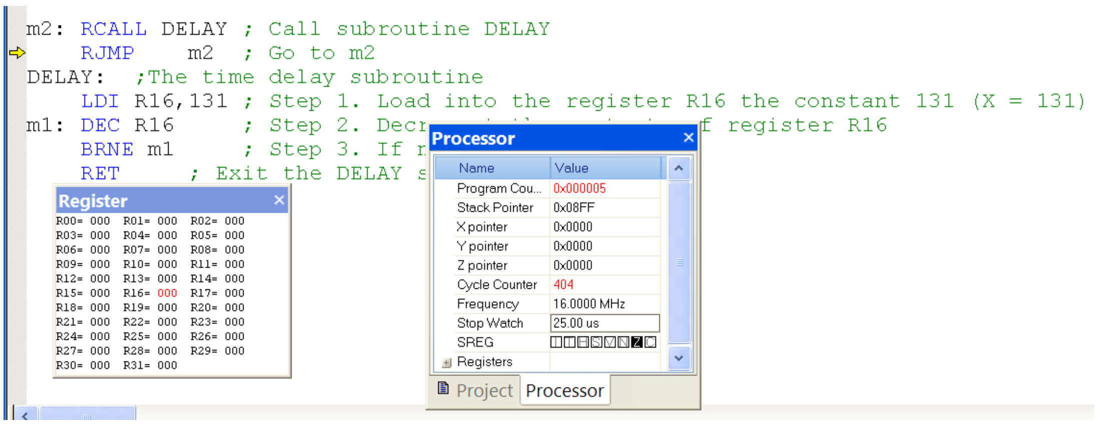
$$T_{DB} = T_c \cdot N_{DB}, \tag{4}$$

where N_{DB} is the number of cycles required for the microcontroller to perform a single cycle, i.e. *DEC* and *BRNE* mnemonics.

In accordance with the Complete Instruction Set Summary [10], we define $N_{DB} = 1 + 2 = 3$ clocks. Then, according to formula (4), we calculate $T_{DB} = 0.0625 \cdot 3 = 0.1875$ µs.

From expression (2) we define: $X = (25 – 0.5) / 0.1875 = 130.66 \approx 131$ cycles.

In the subroutine DELAY we change the initial value of the constant X to the resulting, that is, X = 131 and check the time value forming by the DELAY subroutine in the AVR Studio environment in debug mode (Figure 5).



```
m2: RCALL DELAY ; Call subroutine DELAY
    RJMP    m2  ; Go to m2
DELAY:  ;The time delay subroutine
    LDI R16,131 ; Step 1. Load into the register R16 the constant 131 (X = 131)
m1: DEC R16     ; Step 2. Decr                   f register R16
    BRNE m1     ; Step 3. If r
    RET       ; Exit the DELAY s
```

```
Register
R00= 000  R01= 000  R02= 000
R03= 000  R04= 000  R05= 000
R06= 000  R07= 000  R08= 000
R09= 000  R10= 000  R11= 000
R12= 000  R13= 000  R14= 000
R15= 000  R16= 000  R17= 000
R18= 000  R19= 000  R20= 000
R21= 000  R22= 000  R23= 000
R24= 000  R25= 000  R26= 000
R27= 000  R28= 000  R29= 000
R30= 000  R31= 000
```

Processor

Name	Value
Program Cou...	0x000005
Stack Pointer	0x08FF
X pointer	0x0000
Y pointer	0x0000
Z pointer	0x0000
Cycle Counter	404
Frequency	16.0000 MHz
Stop Watch	25.00 us
SREG	ITHSVNZC
Registers	

Fig. 5. **AVR Studio environment window in debug mode of DELAY subroutine**

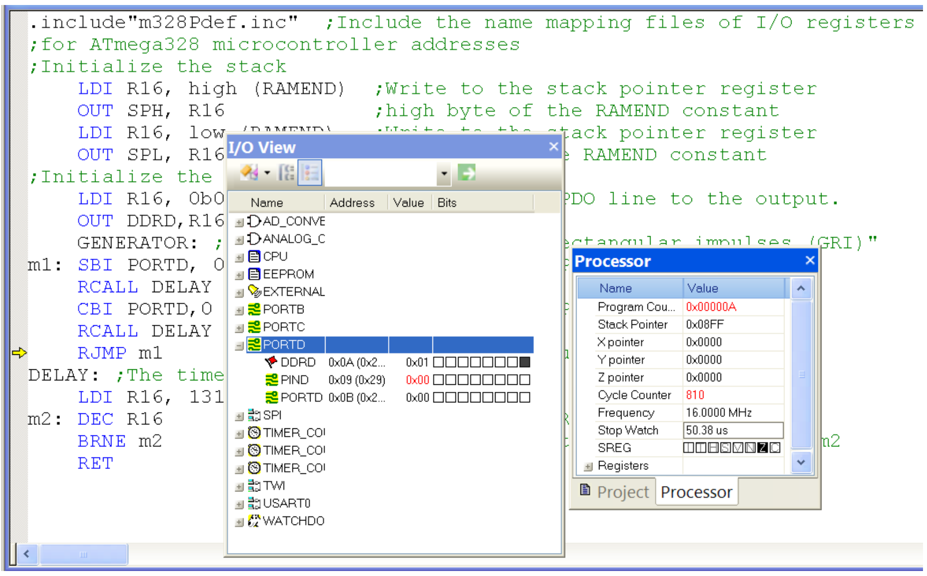
If the DELAY subroutine generates a time delay of the given duration, as seen in the Processor window (Stop Watch 25.00 us), then we go to the development of the program that implements the algorithm of the generator presented above. Debugged in AVR Studio the GENERATOR program is shown in Figure 6.

```
.include"m328Pdef.inc" ;Include the name mapping files of I / O registers
;for ATmega328 microcontroller addresses
;Initialize the stack
LDI R16, high (RAMEND) ;Write to the stack pointer register
OUT SPH, R16    ;high byte of the RAMEND constant
LDI R16, low (RAMEND)  ;Write to the stack pointer register
OUT SPL, R16    ;low byte of the RAMEND constant
;Initialize the HD Port
LDI R16, 0b00000001    ;Configure the PD0 line to the output
OUT DDRD,R16
GENERATOR: ;The program "Generator of rectangular impulses (GRI)"
m1:     SBI PORTD, 0    ;Output logical '1' to PD0 pin
RCALL DELAY     ;Call DELAY subroutine
CBI PORTD, 0    ;Output logical '0' to PD0 pin
RCALL DELAY
RJMP m1 ;Repeat the period output
DELAY: ;The time delay subroutine of 25 µs, with fc = 16MHz
LDI R16, 131    ;Load number 131 in the R16 register
m2:     DEC R16 ;Decrement of register R16 content
BRNE m2 ;If register R16 content is not zero, go to m2
RET     ;Return from DELAY
```

Fig. 6. **GENERATOR program**

Figure 7 shows a fragment of the GENERATOR program debugging mode in AVR Studio environment. After debugging, we record the GENERATOR program into the microcontroller memory using the Atmel AVRISP mkII programmer [11]. It connects to the Arduino Uno through a special connector. GENERATOR.hex file containing the program in the machine code language is located in the project folder.

Figure 8 shows the oscillogram of the signal formed by the Arduino-based generator. From the oscillogram it is seen that the time parameters of the signal almost correspond to the set values. At a given f_G = 20 kHz, the signal frequency is 19.9 kHz. For a given t_i = 25 us, the pulse width is 25.1 us.



```
.include"m328Pdef.inc"  ;Include the name mapping files of I/O registers
;for ATmega328 microcontroller addresses
;Initialize the stack
    LDI R16, high (RAMEND)  ;Write to the stack pointer register
    OUT SPH, R16            ;high byte of the RAMEND constant
    LDI R16, low                ack pointer register
    OUT SPL, R16                e RAMEND constant
;Initialize the
    LDI R16, 0b0                PDO line to the output.
    OUT DDRD,R16
    GENERATOR: ;                                         (GRI)"
m1: SBI PORTD, 0
    RCALL DELAY
    CBI PORTD,0
    RCALL DELAY
    RJMP m1
DELAY: ;The time
    LDI R16, 131
m2: DEC R16                                                m2
    BRNE m2
    RET
```

Name	Address	Value
DDRD	0x0A (0x2...	0x01
PIND	0x09 (0x29)	0x00
PORTD	0x0B (0x2...	0x00

Name	Value
Program Cou...	0x00000A
Stack Pointer	0x08FF
X pointer	0x0000
Y pointer	0x0000
Z pointer	0x0000
Cycle Counter	810
Frequency	16.0000 MHz
Stop Watch	50.38 us
SREG	
Registers	

Fig. 7. **GENERATOR program debugging window in AVR Studio**

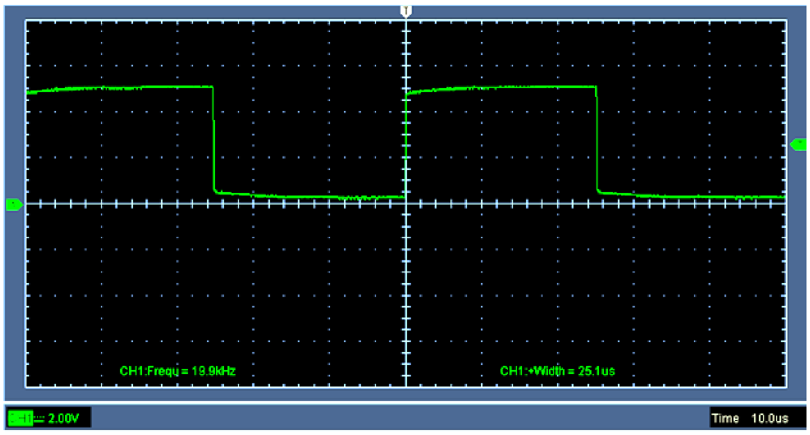


Fig. 8. **Generator signal at the ATmega328 microcontroller PD0 output, Oscillogram**

Minor discrepancies between the experimental parameters of the signal and the given ones can be explained by the fact that the calculations did not take into account the time necessary for the microcontroller to execute the commands:

```
SBI PORTD, 0     ;Output logical ‘1’ to PD0 pin
CBI PORTD, 0     ;Output logical ‘0’ to PD0 pin
RJMP m1          ;Repeat the period output
```

In experimental studies were used the devices: Arduino Uno, AVRISPmkII programmer, DSO-2090 USB Oscilloscope and Computer.

Results and discussion

The hypothesis of the expediency of using Assembler for forming the students’ universal learning action for programming microcontrollers was confirmed by the survey and responses on control questions among the students of the experimental groups. 49 students of the third year of the Electric Power Faculty took part in the survey and control testing: the 1st experimental group of students in the major of “Electric power engineering and electrical engineering” consisted of 24 people; the 2nd experimental group of students in the major of “Agroengineering” included 25 people. The results of the survey are presented in Table 1, and the control testing in Table 2.

Table 1

Student survey results

n/n	Questions	Responses, %	
		1 gr.	**2 gr.**
1	Do you like programming in Assembler?	75	60
2	Does Assembler help you better understand a microcontroller operation?	67	68
3	Can you independently program the given algorithm in Assembly? a) I can b) I can, but only with the teacher's or classmate's help	 33 66	 24 76

Table 2

Student control testing results

n/n	Questions	Answeres	Correct Answers, %	
			1 gr.	**2 gr.**
1	Name the microcontroller components participating in realization the algorithm of a programmable square wave generator …	○ ADC, general purpose registers (GPRs) block, analog comparator; ● GPRs block, program memory, parallel port; ○ timer /counter, GPRs block, analog comparator; ○ non-volatile data memory, program memory, random access memory	83	76
2	How will the programmable generator frequency change if the number of DELAY subroutine cycles is increased?	○ will increase; ● will decrease; ○ remain the same; ○ first increases, and then begins to decrease	96	88
3	What needs to be changed in the operation of the microcontroller without changing the program to reduce the frequency of the programmable generator?	○ load another constant in a register of the GPRs block; ○ increase the supply voltage; ● reduce the frequency of the clock generator; ○ increase the frequency of the clock generator	79	76
4	Indicate the action sequence when developing the program in the AVR Studio environment …	○ choosing a microcontroller, creating a new project, selecting a programming language, assigning a name to the project, specifying the project location; ● creating a new project, selecting a programming language, assigning a name to the project, specifying the project locationt, selecting a microcontroller; ○ specifying the project location, creating a new project, selecting a programming language, assigning a name to the project, choosing a microcontroller; ○ selecting a programming language, creating a new project, assigning a name to the project, specifying the project location, selecting a microcontroller	92	92
5	What basic parameters of a rectangular signal can be measured with an oscilloscope?	● frequency, pulse duration, pulse amplitude; ○ period, pulse current, pulse amplitude; ○ pause current, voltage, pulse duration; ● frequency, pause duration, voltage level of logical zero	96	92

The results of the experimental work can be subdivided into two groups:

1) Teachers' results:

- Elements of experimental work on formation of universal educational actions in the field of programming in Assembler are developed and tested;

- Training manual for training microcontroller programming in Assembler language is created.

2) Students' results:

- Assembler allows a student to see the microcontroller internal devices in the form of graphic models and their interaction with each other, which increases the understanding level of the microcontroller structure and operation;
- The considered pulse generator example can be used for studing such aspects of AVR microcontroller architecture: program memory (FLASH), non-volatile memory (EEPROM), static random access memory (SRAM), and general purpose digital input / output ports.

Conclusions

1. The article presents an example of the organization of effective links between theoretical knowledge and practical skills formed in the bachelors' training process at the engineering faculties.
2. The effectiveness of links is provided by using such cognitive principles as the inclusion of new material in the cognitive structure of the student's personality (the connection of the new material with the knowledge and skills already available to a student), algorithmization, visualization and practical orientation of training acts.
3. Theoretical knowledge in the considered example is the knowledge of the microcontroller architecture, algorithms for implementing certain actions, methods of software development.
4. Practical skills are formed in the process of calculating the microcontroller program execution time and checking the obtained results in the AVR Studio debugging mode.
5. Experimental skills are formed by conducting real studies of the signal of the developed generator, the parameters of which were given at the beginning of the designing.
6. Knowledge and skills make the basis of the formed competence. Its further development presupposes the improvement of knowledge and skills in the process of creative realization in professional or social activity.

References

1. Зубков С.В. Assembler. Язык неограниченных возможностей (Assembler. Language of unlimited possibilities). Moscow: DMK, 2002. 640 p. (In Russian).
2. Vostrukhin A., Vakhtina E. Studying Digital Signal Processing on Arduino Based platform. Proceedings of International scientific conference "Engineering for Rural Development", Volume 15, May 25-27, 2016, Jelgava, Latvia. [online] [30.05.2016]. Available at: http://www.tf.llu.lv/conference/proceedings2016/Papers /N043.pdf
3. Igoe T. Making Things Talk: Using Sensors, Networks, and Arduino to see, hear, and feel your world. O'Reilly Media, 2011. 496 p.
4. Петин В.А. Проекты с использованием контроллера Arduino (Projects using Arduino controller). 2nd ed. SPb.: BHV-Petersburg, 2015. 446 p. (In Russian).
5. Blum J. Exploring Arduino: Tools and Techniques for Engineering Wizardry. Wiley, 2013. 384 p.
6. Arduino Support from MATLAB. [online] [15.09.2015]. Available at: http://uk.mathworks.com/hardware-support/arduino-matlab.html
7. Arduino programming with Atmel Studio. [online] [10.10.2015]. Available at: http://www.visualmicro.com/page/Arduino-for-Atmel-Studio.aspx
8. Microsoft brings Windows 10 to Makers. Arduino Partnership. [online] [25.04.2015]. Available at: https://blogs.windows.com/buildingapps/2015/04/29/microsoft-brings-windows-10-to-makers
9. Вострухин А.В., Вахтина Е.А. Введение в программирование микроконтроллера AVR на языке Ассемблера: учебное пособие (Introduction in Programming of AVR Microcontroller in Assembler Language: textbook). Moscow: Publ. house Ileksa, 2010. 184 p. (In Russian).
10. Complete Instruction Set Summary [online] [23.12.2016]. Available at: http://www.avr-tutorials.com/sites/default/files/Instruction%20Set%20Summary.pdf
11. AVRISP MkII [online] [16.02.2017]. Available at: http://www.atmel.com/webdoc/avrispmkii/index.html