STUDYING DIGITAL SIGNAL PROCESSING ON ARDUINO BASED PLATFORM

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

Abstract. Digital Signal Processing (DSP) has become today an integral part of the basic skills required for scientists and engineers of many specialties. Most courses offer only using for DSP laboratory classes the Matlab system. This approach has the disadvantage – it excludes the students' acquisition of the basic skills of software development for implementation of DSP algorithms based on real digital devices. DSP is informative technologies in real-time and in most cases they use the closest to the hardware Assembler and C/C++ languages. To eliminate this drawback the authors propose using for DSP laboratory classes the known Arduino platform. Nowadays, this platform is one of the most affordable budgetary and rather universal decisions produced on popular AVR microcontrollers. It allows using of the most popular in DSP Assembler and C++ languages. As an example the structure of the laboratory lesson: "Development and research of the moving average filter in Matlab and Arduino» is presented. This filter is the most common. Embodiments of the filter developed in Matlab and ArduinoUno-based controller are given. And oscillograms of the signals at the input of the filter and at its output are shown. Integrated using of the Matlab system and Arduino platform contributes to a better understanding of theoretical fundamentals of DSP, and more importantly, influence development of practical skills in design and development of microprocessor systems for DSP.

Keywords: lab work, virtual-real method, Matlab, moving average filter, microcontroller, program.

Introduction

Digital Signal Processing (DSP) is one of the most powerful technologies that will determine the further development of science and technology in the XXI century. DSP is a set of the mathematical methods, the algorithms, and the techniques that are used to control the signals after they have been converted into digital form. Signal processing can solve many tasks, such as: enhancement of the image quality, recognition and synthesis of speech, compression of data for storage and transmission, etc. Revolutionary changes have already happened in a broad range of activity spheres: communications and broadcasting, radar and sonar, high quality sound reproduction and image compression, in medicine, in monitoring and control systems, etc. [1]. These changes have a direct affect to engineering education. In this article, we will focus on improving the DSP training methods. This is important, because DSP is one of the most topical disciplines for the modern engineer development.

Note that most DSP courses have offered using for lab works only the program Matlab. Virtual lab, has a lack – excludes students' acquisition of practical software development skills to implement DSP based on real-programmable systems (PS). DSP is a real-time information technology. In most cases for implementation DSP based on real-PS C/C++ and Assembler programming languages are used. Popularity rating of these languages is above Matlab [2]. The purpose of this article is to show the ability of using Arduino platform for real lab works on DSP.

Arduino platform (hereinafter simply 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 the most affordable budget solution, implemented on the popular AVR microcontrollers and allows using C/C++ and Assembler languages the most requested for DSP.

The DSP study process should be divided into two stages. The first stage is studying DSP algorithms using computer-aided design (CAD) systems, for example, the Matlab program [6]. The second step – implementation of DSP algorithms developed in Matlab on the base, e.g., Digital Signal Processor. However, Digital Signal Processor is quite complicated PS for beginners to learn DSP [7]. Therefore, the authors suggest for the initial stage of DSP studying to use Arduino as a simplified "analog" of the Digital Signal Processor. Arduino hardware is constructed on AVR microcontrollers of Atmel's company [3-5]. It is a well-known microcontroller family, on the basis of which the Microprocessor Technique is studied in many universities [8-11]. In addition, Arduino software is supported by such program environments as – Matlab, AVR Studio 7 (for 8-bit and 32-bit AVR microcontrollers) and new Windows 10 operating system [12-14].

Materials and methods

As an example, implementation in Matlab and Arduino moving average filter, which is the most common in DSP, is considered. We rely on a generally accepted approach used in the implementation phase of research and development (R & D) during designing PS. This approach with regard to lab classes is: "from virtual model to real sample". Didactic effectiveness of this approach was discussed in detail in the works [15-17]. In this example, the virtual model is created in Matlab, while real PS – on Arduino.

Studying of the moving average filter is based on its principle of operation by averaging a number of points from the input signal. This is expressed by equation (1) from the work [1]:

$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i+j], \tag{1}$$

where $x[]$ – input signal;
$y[]$ – output signal;
M – number of points used in the moving average;
i – sample number (index) of the output signal;
j – number of averaged sample.

This equation only uses points on one side of the output sample being calculated. For example, in the 4 point moving average filter, point 27 in the output signal is given by (2):

$$y[27] = \frac{(x[27] + x[28] + x[29] + x[30])}{4}. \tag{2}$$

Sensor signals of automated process control systems are influenced by various interferences called noise. Noising of a useful signal is determined by:

- *signal-to-noise ratio,* which is equal to the mean divided by the standard deviation;
- *coefficient of variation* (CV), which is defined as the standard deviation divided by the mean, multiplied by 100 % [1].

Student's task: to evaluate effectiveness of the moving average filter for noise suppression by the CV. As initial data the filter order and input signal parameters are used.

For solving this task a student must obtain an input signal with given parameters of digital noise. Digital noise is an important concept in both electronics and DSP [1]. Generation of digital noise is realized by the random number generator (RNG). There is a special function for this task in most programming languages. So, in C/C++ language of Arduino the expression: X = random (0, 255) performs appropriation of variable X of a random value with each new execution of this command. This random value will be located in the range from 0 to 255 and has an equal probability of appearance within these limits.

Step 1 – Matlab-based (virtual) part of the lab work. The model of the system for forming the input signal samples, which is shown in Fig. 1, is developing in the Simulink (included in Matlab) software. Uniform Random Number (URN) block is a RNG with uniform distribution. Rounding Function block is intended for rounding off numbers, generated by URN, to integers. Scope unit is designed for visualization of a signal applied to its input.

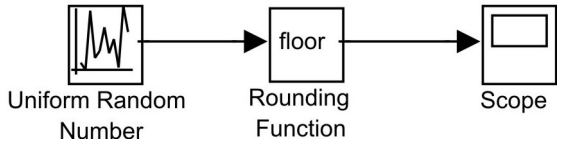

Fig. 1. **Model of the system for forming the input signal samples**

Range generated numbers from 0 to 255 are set in URN settings. This range is convenient because the maximum decimal number 255 is equivalent to the maximum value of 8-bit binary code. To store this code, you can use 8-bit registers, which simplifies the task solution while implementing DSP algorithms in Assembler.

Fig. 2 shows an input signal consisting of 30 samples. The values of the input signal samples are read visually from the oscillogram of the Scope block (Fig. 1).

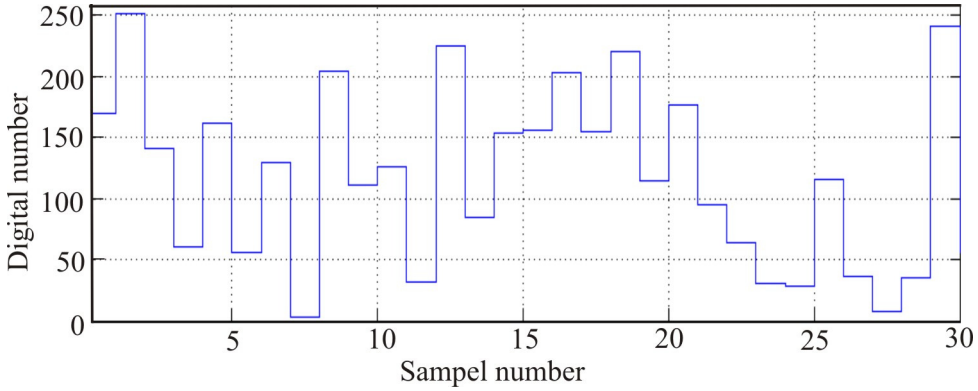

Fig. 2. **Filter input signal**

Let us imagine the signal as a vector consisting of 30 samples in the form of one-dimensional matrix from 10 columns by 3 samples in each. This matrix is a part of the Program 1, which is presented in Fig. 3. Matlab processes each sample (matrix element) by columns, starting with the first sample in the following sequence: $x[1] = 170$, $x[2] = 251$, $x[3] = 141$, $x[4] = 60$ … $x[30] = 241$.

```
% FINITE IMPULSE RESPONSE (MOVING AVERAGE FILTER)
% This program is designed to process 30 samples of the input signal by the 4 point moving
% average filter
x = [170 60 129 111 225 156 220 95 28 7
 251 162 3 126 84 203 114 64 116 35
 141 56 204 32 153 155 176 30 36 241];
for i = 1:26                    % Loop through each sample in the output signal
       y(i) = 0;                % Fill sample of the output signal with null values
       for j = 0:3              % Loop for calculation of the next sample
       y(i) = y(i)+ x(i+j);
end                             % Loop end
y(i) = y(i)/4;                  % Calculate the average value of the next output signal
end                             % Loop end
y                               % Output of values of the output vector on monitor
i = [1:26]; stairs(i,y);        % Build the graph of the vector of output samples
```

Fig. 3. **Program 1**

The result of the Program 1 is the output signal vector consisting of 26 samples and shown in Fig. 4. The value of the 27th sample is not calculated, because this calculation requires the 31st sample of the input signal, which is absent.

Matlab has functions meant to work with a vector argument. To calculate the average values the function (*mean*), standard deviation – function (*std*) are used. These functions could be utilized for processing the input and output filter signals. As a result, we obtain for the input signal the average value– $\mu_i \approx 119.4$ and its standard deviation – $\sigma_i \approx 72.2$. For the output signal – $\mu_o \approx 115.4$ and $\sigma_o \approx 39.9$, respectively. Then the CV of the signals: at the filter input $C_i = \sigma_i / \mu_i =$ $= (72.2 / 119.4) \cdot 100\ \% \approx 60\ \%$ and at the filter output $C_o = \sigma_o / \mu_o = (39.9 / 115.4) \cdot 100\ \% \approx 35\ \%$.

Evaluating effectiveness of the filter allows the noise reduction factor, which shows in how many times the noise at the filter output is less than the noise at its input. In this case, the noise reduction factor by the filter that was realized in Matlab: $Cm = Ci / Co = 60 / 35 \approx 1.7$.

Step 2 – Arduino-based (real) part of the lab work. The experiment involved two controllers Arduino Uno. The first controller is required to generate the input signal with the set parameters, the second controller – for implementing the filter. ATmega328 Microcontroller, on which Arduino Uno controllers are implemented, has the supply voltage $Vcc = 5$ V. The 8-bit DAC with R-2R structure is connecting to the parallel ports of each controller. An example of DAC construction and its connection to the Arduino Uno controller is in the resource [18]. Input and output filter signals are controlled by the DSO-2090 USB dual-channel oscilloscope.

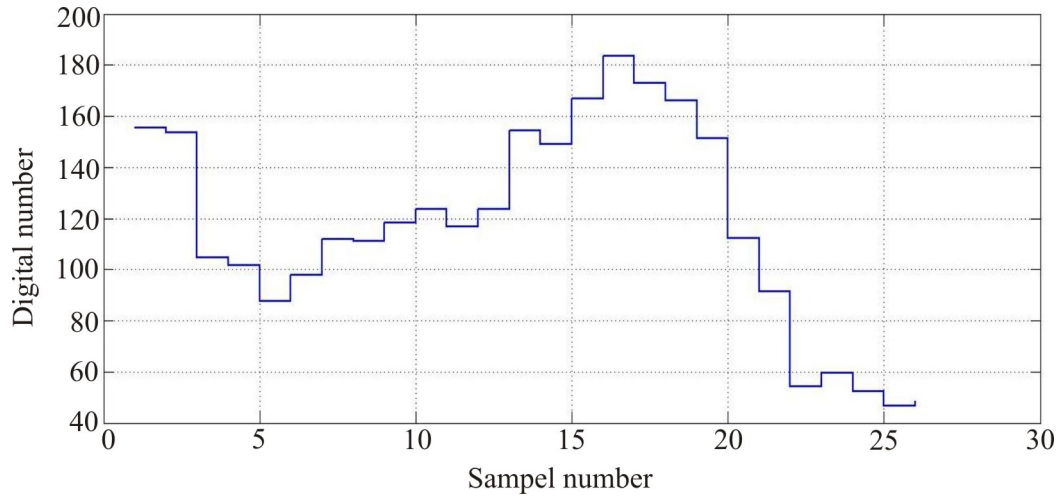

Fig. 4. **Filter output signal**

The first controller's RNG is tuned to generate a random number between 0 and 255. Each random number is output by this controller in the form of 8-bit binary code to a parallel port. If binary code that is equivalent to 255 is fed into the port, then the maximum voltage level ≈ 5 V on the DAC output is generated. The minimum voltage level ≈ 0 V is generated at the DAC output when binary code of zero is fed to the input port. If the value of a random number is changed by one unit, then DAC output voltage is changed to the value determined by the expression:

$$Vcc / 255 = 5 / 255 \approx 19.6 \text{ mV}.$$

The signal at the DAC output of the first controller, i.e. at the filter input, is displayed by the CH1 channel in Fig. 5. The signal at the DAC output of the second controller, i.e. at the filter output, is displayed by the CH2 channel in the same figure.

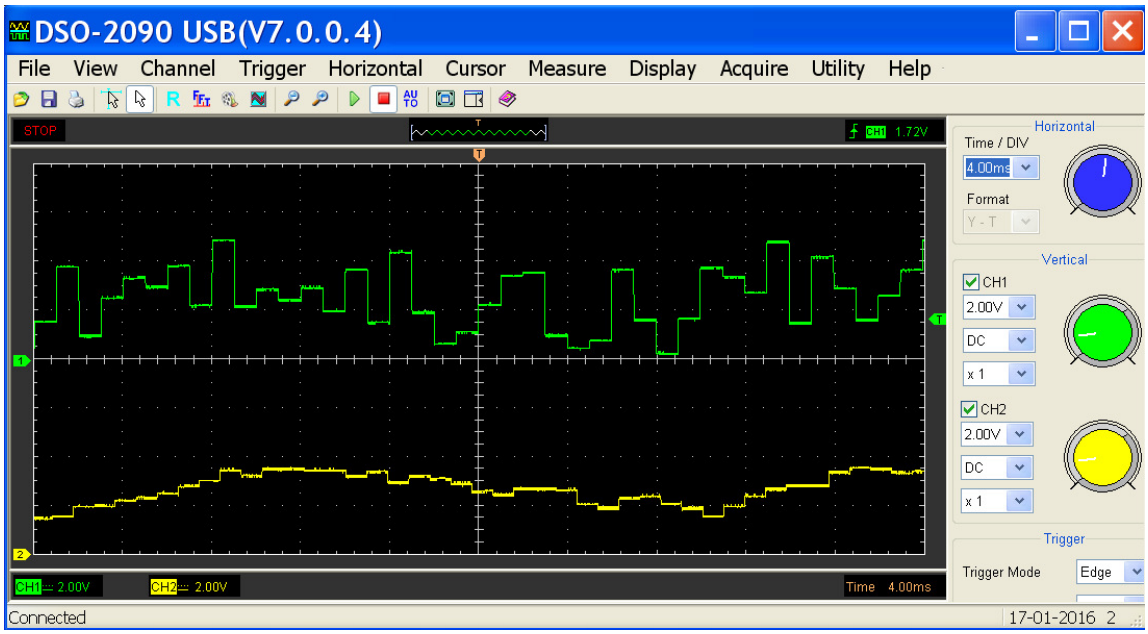

Fig. 5. **Oscillogram of signals:** CH1 - the filter input signal; CH2- the filter output signal

The following Fig. 6 shows the program 2 fragment for a 4 point moving average filter that is realized on the Arduino Uno controller.

We form the sample values of the input and output filter signals according to oscillograms and process them in Matlab. As a result, we obtain for the input signal of the filter the mean value $\mu_i \approx 2.7$ V and the standard deviation $\sigma_i \approx 1.4$ V; for the output signal, respectively $\mu_o \approx 2.5$ and $\sigma_o \approx 0.5$ V. We define the CV for the input and output signals of the filter: $C_i = \sigma_i / \mu_i = (1.4 / 2.7) \cdot 100\ \% \approx 52\ \%$;

$C_o = \sigma_o/\mu_o = (0.5/2.5)\cdot 100\ \% \approx 20\ \%$. In this case, the noise reduction factor by the Arduino-based filter is $C_a = C_i / C_o = 52/20 \approx 2.5$.

```
{               // Loop Start
x1 = x2;        // Send the second sample into a memory cell of the first sample
x2 = x3;        // Send the third sample into a memory cell of the second sample
x3 = x4;        // Send the fourth sample into a memory cell of the third sample
x4 = analog Read(ADC0);       // Read from ADC the conversion result of the next sample
                              // and send its value into a cell memory of the fourth sample
x4 = x4/4;              // Convert 10-bit binary code to an 8-bit
y = (x1+x2+x3+x4)/4;    // Calculate the average value of the last four samples
PORTD = (y);            //and output this value to the parallel port PD
}                       // Go to the top of the loop
```

Fig. 6. **Fragment of Program 2**

Results and discussion

The important thing is that in both cases we have the noise reduction factor close to a definite value. It is known that the noise reduction factor of the moving average filter is equal to the square root of the samples number involved in the averaging. For example, a 100 point moving average filter reduces the noise by a factor of 10 [1]. Consequently, for our 4 point filter the noise reduction factor should in ideal case be 2. This enables to conclude that both study steps of the moving average filter – simulation and experiment are correct.

This pedagogical project of implementing the virtual-real method to organization of DSP laboratory classes using Arduino is in its infancy. There is only one study Arduino platform as the means of DSP training [7]. It is not unusual, because, on the one hand, the DSP discipline was formed as a separate course of radio engineering faculties in Russia only in the last decade, on the other hand, the developing Arduino software has just begun to adopt as training means.

Conclusions

Arduino platform can and should be used to study the DSP course. The results presented in this paper prove that there is an opportunity of developing real DSP systems on the base of this platform. This fact contributes to realization of the important in educational process scientific and didactic principle – from theory to practice. The benefits of real lab work compared with virtual ones are obvious and do not require evidence, sufficient to recall the words of Confucius: “I hear and I forget, I see and I remember, I do and I understand.” Reasonable integration in lab works the virtual and real techniques is contributing to deeper understanding of the studied issues.

References

1. Stewen W.S. The Scientist and Engineer’s Guide to Digital Signal Processing. Second edition. San Diego: California Technical Publishing, 1999. 650 p.
2. TIOBE Index for February 2016 [online] [12.02.2016]. Available at: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
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. Солонина А.И., Клионский Д.М., Меркучева Т.В., Перов С.Н. Цифровая обработка сигналов и MATLAB: учеб. пособие (Digital Signal Processing and MATLAB: textbook). SPb.: BHV-Petersburg, 2014. 512 p. (In Russian).
7. Hochgraf C. Using Arduino To Teach Digital Signal Processing. Reviewed Paper of ASEE Northeast Section Conference, March 14-16, 2013, Norwich University. [online] [12.02.2016]. Available at: http://asee-ne.org/conferences/aseene/2013/index.php/aseene/aseene2013/paper/view/235/24.
8. Хартов Э. Ф. Микроконтроллеры AVR. Практикум для начинающих : учеб. пособие (AVR Microcontrollers. Workshop for beginners: tutorial). Moscow: Publ. house Bauman Moscow State Technical University, 2013. 280 p. (In Russian).

9. Белов А.В. Разработка устройств на микроконтроллерах AVR (Development of devices on AVR microcontrollers). SPb.: Science and Technique, 2013. 528 p. (In Russian).
10. 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).
11. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «ATMEL» (TinyAVR and MegaAVR Microcontrollers of ATMEL's company). Moscow: Publ. House "Dodeka-XXI», 2015. 560 p. (In Russian).
12. Arduino Support from MATLAB. [online] [15.09.2015]. Available at: http://uk.mathworks.com/hardware-support/arduino-matlab.html.
13. Arduino programming with Atmel Studio. [online] [10.10.2015]. Available at: http://www.visualmicro.com/page/Arduino-for-Atmel-Studio.aspx.
14. 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.
15. Вахтина Е., Вострухин А. Интеграция «реального» и «виртуального» в лабораторном эксперименте (Integration of the "real" and "virtual" in a lab experiment). Vysshee obrazovanie v Rossii, 2008, No 6. pp. 77-81. (In Russian).
16. Вахтина Е.А. Дидактический проект лабораторного эксперимента (Didactic project of lab experiment). Pedagogicheskij zhurnal Bashkortostana, 2009, No 3 (22). pp. 147-155. (In Russian).
17. Vakhtina E., Palkova Z. Didactic designing of Learning Objects. Proceedings of 14th International Scientific Conference "Engineering for Rural Development", May 20-22, 2015, Jelgava, Latvia, pp. 661-668. [online] [31.05.2015]. Available at: http://www.tf.llu.lv/conference/proceedings2015/Papers/107_Vakhtina.pdf.
18. Arduino Waveform Generator. [online] [02.12.2015]. Available at: http://www.instructables.com/id/Arduino-Waveform-Generator/?ALLSTEPS.