

Hardware random numbers generator based on microcontroller

Denys Ostapets*, Volodymyr Dziuba, and Pavlo Ivin

Ukrainian State University of Science and Technologies, Department of electronic computers, 49010 Dnipro, Lazaryana Street 2, Ukraine

Abstract. The work considers the principles of development and organization of the random number generation system using a hardware generator of true random numbers. An analysis of noise sources for a hardware generator of true random numbers has been carried out. The architecture of the random and pseudo-random number generation system, the structure of the hardware part, and the interaction protocol of the system elements have been developed. Hardware and software of the system are implemented. An analysis of numbers sequences randomness degree obtained in different modes was carried out using a NIST statistical tests suite and visual tests. Recommendations for using the system are given.

1 Introduction

Random numbers are widely used in modern technology and applied sciences. These are areas such as statistics, numerical methods, graphics, computer games, testing, simulation, cryptography, information security, etc. At the same time, there are two main approaches to obtaining random numbers:

- Generation of pseudorandom numbers (PRN) using special algorithms or tables.
- Use of special hardware devices that generate true random numbers (TRN).

Unlike true random numbers, pseudorandom numbers are generated according to a special algorithm using a certain initial setting. Therefore, with the same initial setting, the same sequence of numbers will be obtained. In addition, usually, such sequences are repeated after a certain period. To generate true random numbers, as a rule, special devices are built that use a noise signal from one or more sources, which is digitized. Next, the obtained value is used either for the initial setting of the sequence of certain random numbers, or as another random number. The quality of the solution of the applied task in which they are used depends on the quality of the obtained random numbers. Thus, the development and research of tools for generating random and pseudorandom numbers is an actual problem.

The purpose of the work is to research the quality of random number sequences obtained by means of hardware true random number generators (TRNG) using different noise sources. Tasks to be solved in the work: analysis and selection of a noise signal source for a hardware generator of true random numbers; development of the architecture of the hardware-software system for generating random numbers;

development of the hardware structure and protocol for the interaction of system elements, random number generation modes; system hardware and software implementation; analysis of the degree of randomness of sequences of numbers obtained in different modes using known sets of tests.

2 A brief overview of noise sources

Let's analyze the known sources of noise (entropy) that can be used in a hardware random number generator. Among the known (and practically used) sources of noise, the following should be considered [1 – 5]:

- Shot noise is noise in radio-electronic elements (diodes, transistors, lamps) that occurs when current flows due to fluctuations.
- Thermal noise, or Johnson noise, is the noise in passive elements that occurs due to the random movement of electrons in a resistive environment and increases with increasing temperature and resistance.
- Avalanche noise is the noise that occurs in p-n junctions of Zener diodes or emitter-base junctions of bipolar transistors operating in reverse (avalanche) breakdown mode.
- Noise at the outputs of ring oscillators, each of which consists of an odd number of inverters connected in series, and the outputs of several oscillators are combined by an XOR circuit; noise occurs due to the randomness of oscillator parameters.
- Atmospheric noise, which can be obtained using a radio receiver.
- Digital circuits with an undefined state consisting of a set of a pair of inverters (as connected in a ring) and transistors (which control the inverters).

Avalanche noise in Zener diodes or dedicated noise diodes is a popular and inexpensive source of entropy.

* Corresponding author: odaua@i.ua

Such a noise source is often used in industrial designs of hardware random number generators or signals. Its main drawback is the relatively low speed of generating random numbers. Atmospheric noise, which uses the popular service RANDOM.ORG [4], is also noteworthy. Compared to the previous option, such an entropy source is more expensive (due to the cost of equipment), but the number generation speed is also faster. The paper proposes to perform a comparative analysis of sequences of random numbers obtained using avalanche and atmospheric noise sources.

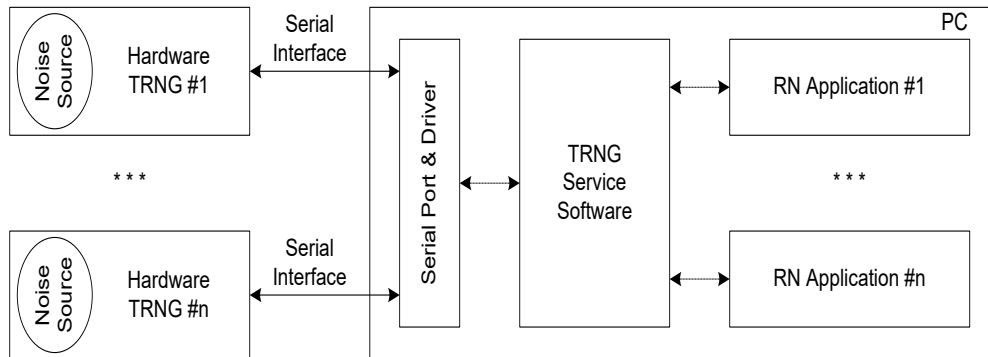


Fig. 1. Generalized system architecture.

Devices can use different sources of noise. Each TRNG device is connected to a computer port using a serial interface. Interaction with each device via the interface and control of its operation on the computer side is performed by specially developed software - the random number generation service (RNG Service). RNG Service issues commands to the device for setting operating modes, generating numbers, and receives generated numbers from the device. Applications to get random numbers access the RNG Service using the API.

The selection of a specific TRNG device with a noise source is done using the RNG Service Software. The RNG Service is also responsible for the representation format of random numbers.

3.2 Principles and modes of TRN generation

To obtain one binary digit of a truly random number, the device converts the analog value of the amplitude of the noise signal and compares the obtained discrete value with a predetermined threshold. If the value is greater than the threshold, then the current bit of the random number is taken equal to 1, otherwise it is 0. Thus, to form a number of bits n , it is necessary to perform n such operations. The threshold value is assumed to be the same constant for all bits (transformation and comparison operations) and is experimentally determined at the device testing stage.

Within the framework of the presented system, with the help of TRNG extensions, it is possible to generate both sequences (sets) of truly random and pseudorandom numbers. As you can see, the process of analog-to-digital conversion is equally correct. So, generating true random numbers takes a significant amount of time (by the standards of a computer system). In the case of generating a sequence of pseudorandom numbers, a true

3 Principles of building a random number generation system

3.1 System architecture

The architecture of a system using a hardware-based true random number generator is shown in Fig. 1.

Several different TRNG hardware devices can be used in the system, including simultaneously.

random number can be used as the initial setting of the generator. The actual process of generating the next elements of the sequence in this case occurs on the side of the RNG Service program. Thus, the process of obtaining a sequence of numbers becomes faster than in the first case: only one true random number is required, a relatively slow serial interface and analog-to-digital conversion are not used to obtain the following numbers, computing operations on a PC are faster than on an external device.

Thus, it is advisable to implement several algorithms for generating sequences of pseudorandom numbers in the system, namely the BBS (Blum-Blum-Shub) algorithm [6] and the LFSR (Linear Feedback Shift Register) algorithm [7]. These algorithms are quite popular and effective according to the authors.

So, a system using a hardware generator can operate in the following modes:

- Generation of the next TRN (one call to the device is required to generate one number).
- Initial setup of the PRN generator according to the BBS algorithm (TRN are used for initialization).
- Generation of the next PRN in sequence according to the BBS algorithm.
- Initial setup of the PRN generator according to the LFSR algorithm (TRNs are used for initialization).
- Generation of the next PRN in sequence according to the LFSR algorithm.

3.3 Communication protocol

A specialized protocol (command system) has been developed for the exchange between a device and a computer. Data is transmitted via a serial interface in ASCII format, which simplifies the analysis and

configuration of the exchange protocol. Commands and responses consist of:

- command/response code (1 character);
- command/response parameter (optional);
- command/response termination indicator (carriage return character "CR" - 0Dh ASCII code).

The device generates 2-byte numbers, which are transmitted as a response parameter in hexadecimal form. To operate the device in the modes described above, three types of commands/responses are sufficient:

- command/response "G" to generate the next TRN (see Fig. 2);
- command/response "R" to reset the device (see Fig. 3).
- response "E" if the command is not recognized by the device (see Fig. 4).

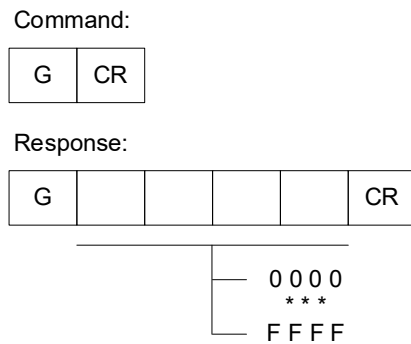


Fig. 2. TRN generation command and response structure.

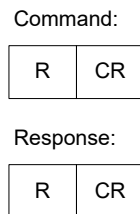


Fig. 3. Reset command and response structure.

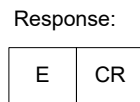


Fig. 4. Error response structure.

3.4 System hardware

The hardware device of the TRNG system is built using a microcontroller (see Fig. 5).

As a source of noise (entropy), as indicated above, a semiconductor element is used - a Zener diode. The microcontroller performs the task of processing the digitized noise signal, which has been previously converted from analog form using an analog-to-digital converter (ADC). Communication with a computer is implemented using a serial port (UART), an RS-232 interface with a converter to USB is used. The ADC and UART blocks are internal subsystems of the microcontroller.

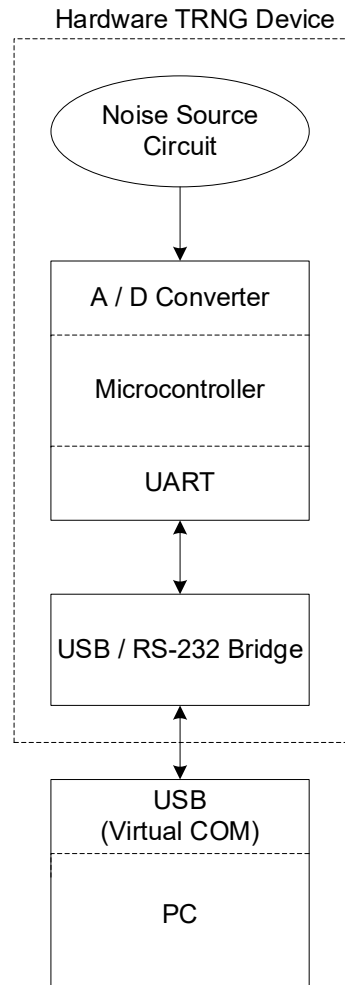


Fig. 5. System hardware structure.

The Atmega328 was chosen as the microcontroller, which meets the requirements for the hardware: it includes a 10-bit ADC device and serial hardware port. The controller is fast enough. As a platform for implementing the device, it is advisable to use the Arduino Uno based on the selected controller, which also includes a USB / RS-232 converter.

A Zener diode 1N746 (KS133G) was used to construct the noise signal generation circuit (see Fig. 6), the signal amplifier was implemented on two inverters of the CD4049 microcircuit (K561LN2). The circuit is based on [8].

4 Research of the randomness of the obtained sequences of numbers

Let's carry out a comparative analysis of the quality of the following sequences of numbers:

- true random numbers obtained using a device with an avalanche noise source based on a Zener diode (developed in this work);
- true random numbers obtained from atmospheric noise using the [4] service;
- sequences of pseudorandom numbers obtained by the BBS and LFSR algorithms with the initial setting of true

random numbers using a device (developed in this work).

The degree of randomness of numbers sequences was assessed using the NIST statistical test suite [9], which is the de facto standard for testing for randomness. Testing

was carried out using the software represented in [10], also it can be used the software represented in [11, 12].

For testing, four sequences were obtained (one of each type indicated above), consisting of 20000 bits each (1250 numbers, 2 bytes each). The results of passing the main NIST tests are shown in Table 1.

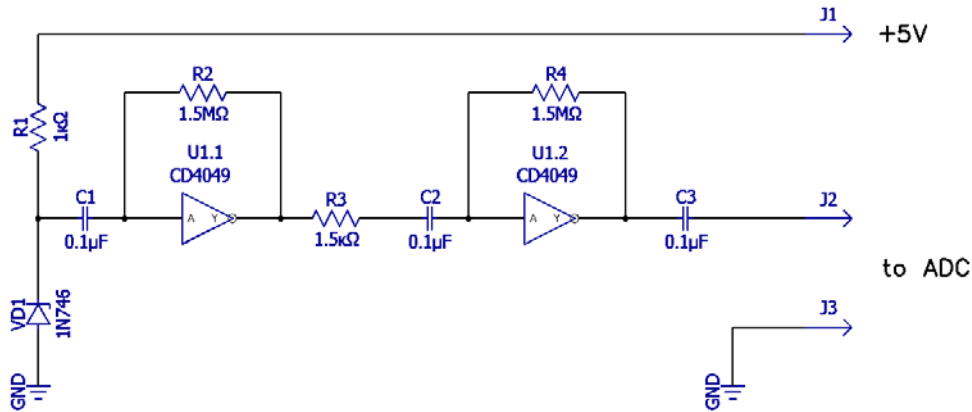


Fig. 6. Noise source circuit.

Table 1. Results obtained using the NIST test suite.

Test name	The way of RN generating			
	Hardware TRNG (with Zener diode)	TRN from atmospheric noise	BBS with initial TRN	LFSR with initial TRN
Frequency (Monobit) Test	+	+	+	+
Frequency Test within a Block	+	+	+	+
Runs Test	+	+	+	+
Test for the Longest Run of Ones in a Block	+	+	+	+
Binary Matrix Rank Test	+	+	+	+
Discrete Fourier Transform (Spectral) Test	+	+	-	+
Non-overlapping Template Matching Test	+	+	+	+
Serial Test	+	+	+	+
Approximate Entropy Test	+	+	+	+
Cumulative Sums (Cusum) Test	+	+	+	+

In total, the set includes 15 tests, but 5 of them cannot be performed due to the length of the input sequence. The results of these tests are not included in the table.

As we can see from the data in Table 1, all relatively short sequences perform well in the main tests.

Also, for a visual assessment of the degree of randomness, the graphic tests proposed in [13, 14] were used. With the help of the test, you can observe the presence (or absence) of visual patterns in binary sequences. For the received sequences of numbers, the image (bitmap) consists of 141*141 pixels. The resulting images are shown in Fig. 7 – 10.

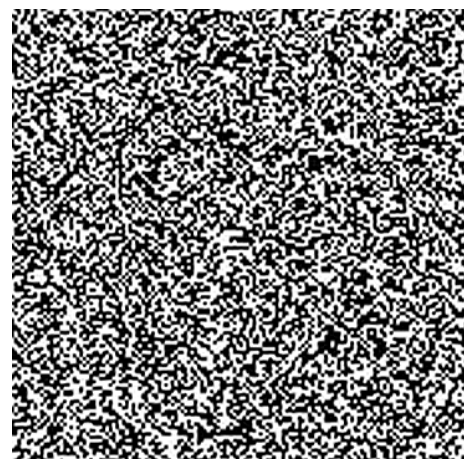


Fig. 7. Hardware TRNG bitmap.

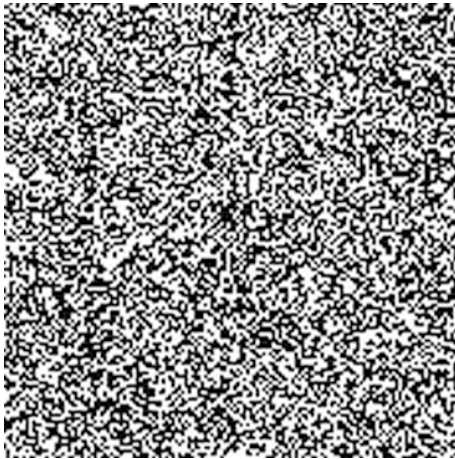


Fig. 8. TRN from atmospheric noise bitmap.

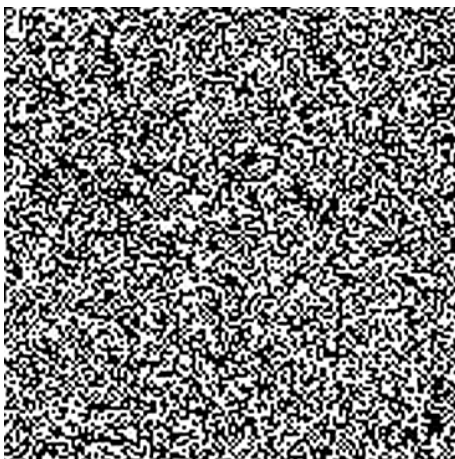


Fig. 9. BBS PRN bitmap.

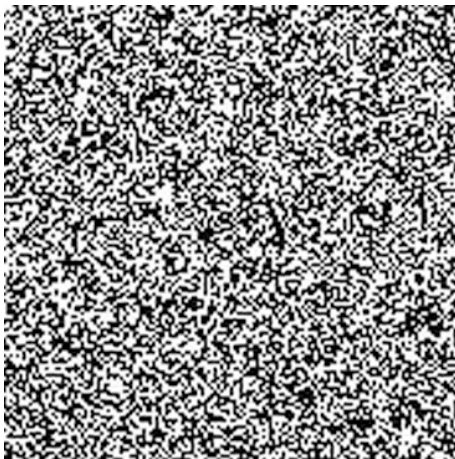


Fig. 10. LFSR PRN bitmap.

On the images Fig. 7 – 10 significant patterns are not observed. Thus, all presented sequences of random numbers successfully pass the visual test.

5 Conclusions

The results of tests from the NIST set and visual tests showed that all the received sequences of numbers have signs of randomness. Thus, in applications with high performance requirements, the use of the proposed system in pseudorandom number generation mode (with a true random initializing) is justified. However, it is clear that to obtain more complete results of assessing the degree of randomness, it is worth operating with longer sequences.

References

1. A. Nakonechnyi, D. Mozola, Automation, measurement and control, **1**, T.1, 19–24 (2019)
2. I. Podorozhnyi, Young scientist, **1** (105), 190–194 (2016)
3. S. Edwards, Electronic components, **11**, 19–25, (2013)
4. Randomness and Integrity Services, <https://www.random.org>
5. G. Taylor, G. Cox, IEEE Spectrum, **48** (9), 32–58 (2011)
6. L. Blum, M. Blum, M. Shub, SIAM Journal on Computing, **15**, 364 – 383 (1986)
7. B. Schneier, Applied Cryptography, 816 (2013)
8. I. Nechaev, Radio, **4**, 55 – 56 (2007)
9. A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, NIST Special Publication 800-22, Revision 1a, 131 (2010)
10. D. Ostapets, Information protection methods and means. Part 1, DNURT, 19 (2011)
11. NIST SP 800-22: Download Documentation and Software, <https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software>
12. S. Kho Ang, NIST Randomness Test suite, https://github.com/stevenang/randomness_testsuite
13. B. Allen, Pseudo-Random vs. True Random, <https://boallen.com/random-numbers.html>
14. Random Bitmap Generator, <https://www.random.org/bitmaps/>