Title: Microcontroller-based Random Number Generator Implementation by Using Discrete Chaotic Maps

Authors: Serdar ÇİÇEK
Recieved: 2020-04-27 03:59:56

Accepted: 2020-06-17 14:47:55

Article Type: Research Article
Volume: 24
Issue: 5
Month: October
Year: 2020
Pages: 832-844

How to cite
Serdar ÇİÇEK; (2020), Microcontroller-based Random Number Generator Implementation by Using Discrete Chaotic Maps. Sakarya University Journal of Science, 24(5), 832-844, DOI: https://doi.org/10.16984/saufenbilder.727449
Access link
http://www.saujs.sakarya.edu.tr/en/pub/issue/56422/727449

Sakarya University Journal of Science 24(5), 832-844, 2020

Microcontroller-based Random Number Generator Implementation by Using Discrete Chaotic Maps

Serdar ÇİÇEK*[1]

Abstract

In recent decades, chaos theory has been used in different engineering applications of different disciplines. Discrete chaotic maps can be used in encryption applications for digital applications. In this study, firstly, Lozi, Tinkerbell and Barnsley Fern discrete chaotic maps are implemented based on microcontroller. Then, microcontroller based random number generator is implemented by using the three different two-dimensional discrete chaotic maps. The designed random number generator outputs are applied to NIST (National Institute of Standards and Technology) 800-22 and FIPS (Federal Information Processing Standard) tests for randomness validity. The random numbers are successful in all tests.

Keywords: Chaotic map, Random number generators, NIST 800-22, FIPS, Microcontroller

1. INTRODUCTION

Chaos theory and chaotic systems have typical features such as very much sensitivity to initial conditions, random-like behavior, ergodicity and broadband [1]. Chaotic systems have been used in different disciplines of science in the recent decades [2]. One of these fields is random number generator (RNG). RNG designs are very important because random numbers should be unpredictable in information security, encryption and cryptographic applications [3]. In the literature, various random number generators (RNGs) have been designed with different chaotic and hyperchaotic systems. Wang et al. designed RNG by using a single chaotic system [4]. Ergün et al. designed RNG by using non-autonomous continuous-time chaotic oscillator [5]. Yalçın et al. designed RNG by using double-scroll chaotic system [6]. Vaidyanathan et al. [7] and Liu et al. [1] designed RNG by using 4D hyperchaotic systems. Akgul et al. designed RNG by using integer and fractional chaotic system based on microcomputer (Raspberry Pi) [8].

Besides chaotic and hyperchaotic systems, chaotic maps (CMs) are also used in chaos-based communication, encryption applications and RNG designs. There are two types of CMs: continuous and discrete. The continuous chaotic map (CM) function graph is in an unbroken structure [17]. Discrete CMs are generally obtained by iterations. In fact, discrete CMs are a special version of the continuous system with

* Corresponding Author: serdarcicek@gmail.com ; serdarcicek@nevsehir.edu.tr
[1] Nevşehir Hacı Bektaş Veli University, Vocational School of Hacıbektaş, Department of Electronic and Automation, Nevşehir, TURKEY. ORCID: http://orcid.org/0000-0002-8738-3985

instantaneous states depicted by continuous CM variables. Discrete CMs often have relatively simple algebraic equations than continuous CMs [9]. Also, the number of system state variables required for the hyperchaotic continuous time system is minimum four, but this situation is not necessary for discrete CMs [40]. Digital embedded systems cannot directly support the structure of continuous CMs. Therefore, the continuous signal needs to be discretization for digital embedded systems [10, 35].

Many CMs with different features have been introduced in the literature. Alzaidi et al. [11], Alpar [12], Hua et al. [13] and Liu et al. [14] introduced one-dimensional (1D) chaotic maps. In literature, Henon [15], Chen [16], Barnsley Fern [17], Tinkerbell, Lozi and other two-dimensional (2D) discrete CMs [18-21] and fractional CMs [22, 23] are also introduced.

Discrete chaotic maps are mostly used in chaotic maps based RNG, encryption and communication applications. Alghafis et al. designed encryption scheme based on quantum map and continuous chaotic system [24]. Fridrich designed symmetric ciphers based on continuous two-dimensional chaotic map [25]. Liao et al. designed secure image communication based on Chebyshev map [26]. Papadimitriou et al. proposed two new communication protocols by using CMs [27]. CMs have been widely used in encryption applications as well as communication applications. Zhang et al. presented image encryption design based chaotic map for different image formats [28]. Li et al. introduced a new encryption algorithm for image based on improved logistic map [29]. Pak et al. proposed an image steganography algorithm using a 1D chaotic map [30]. Naseer et al. introduced a new approach using 3D mixed CM to improve multimedia security [31]. Herbadji et al. introduced enhanced quadratic CM based color image encryption scheme [32].

RNGs are generally preferred on the basis of communication and encryption structures for information security. Chaos-based RNGs have also become popular in random number generation nowadays. For chaos-based RNG, chaotic maps have been used such as chaotic and hyperchaotic systems. Dastgheib et al. designed a digital pseudo RNG based on discrete sawtooth CM [33]. Avaroğlu et al. designed a pseudo RNG by using Arnold cat map [34]. Lambic et al. designed a pseudo RNG by using discrete-space CM [35]. Tutueva et al. designed PRNG based on adaptive discrete CMs [36]. In addition to a single chaotic map based RNGs, RNGs are designed using multiple chaotic maps. Garasym et al. introduced new nonlinear chaotic PRNG based on discrete tent and logistic map [37]. Magfirawaty et al. introduced RNG design based on discrete Henon and logistic map [38]. Ergün and Tanrıseven implemented RNG based on discrete-time CM on FPAA (Field Programmable Analog Array) device [39].

The chaotic maps and chaotic map based applications in the literature are generally presented as simulations. The hardware implementation of these chaotic maps and applications is important for various real engineering applications. Hardware realizations can be performed as analog or embedded system designs (such as FPGA - Field Programmable Gate Array-, Microcontroller). In analog designs, the values of electronic devices can vary depending on the ambient conditions. This situation is much less in embedded systems. Additionally, updating analog designs is more difficult than embedded designs. Therefore, embedded system designs are more advantageous than analog designs. In the embedded system designs, microcontroller chips are much cheaper than FPGA chips. Also, updating the design is easy with the microcontroller than FPGA. It is advantageous to use microcontrollers in similar applications for the stated reasons.

In this study, first, microcontroller-based design is implemented of Lozi, Tinkerbell and Barnsley Fern discrete CMs and verified by numeric simulation results. Then, microcontroller-based RNG is implemented by using the CMs and test the design.

Other parts of the article are organized as follows: Section-2 presents the microcontroller-based implementation of the discrete CMs verified by simulation results, Section-3 presents the implementation of microcontroller-based RNG design by using the discrete CMs, Section-4 provides NIST 800-22 and FIPS randomness tests results and Section 5 concludes the paper and mentions about future works.

2. MICROCONTROLLER-BASED IMPLEMENTATION OF THE DISCRETE CHAOTIC MAPS

In this section, Lozi, Tinkerbell and Barnsley Fern discrete CMs are introduced. Also the chaotic maps are implemented on microcontroller. In addition, the numerical simulation results with Matlab® program are compared the microcontroller outputs.

Arduino UNO board given in Figure 1 is used in Microcontroller design. Arduino boards and its software are open source. The board has an 8-bit ATmega328p microcontroller. The microcontroller software is programmed according to the flow chart given in Figure 2. In the microcontroller software, microcontroller input settings and CM parameter values are performed first. In the other step, the values of the state variables of the CM are sent to the output via the USB port. Then, the new values of the state variables are calculated and sent to the output. This process continues as long as the system is running.

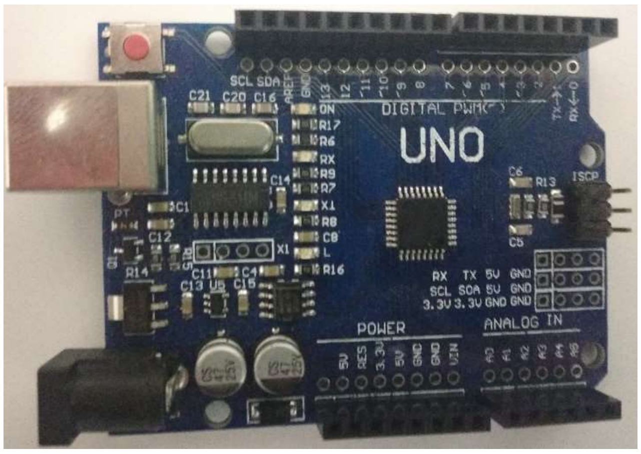


Figure 1 Arduino UNO microcontroller board

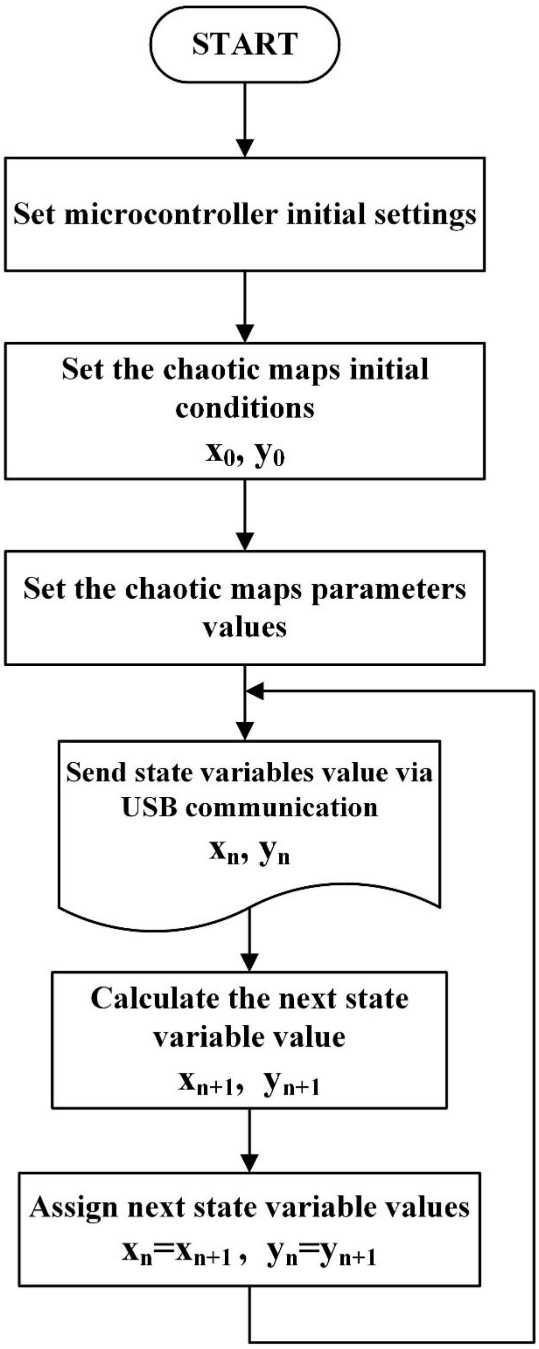

START
Set microcontroller initial settings
Set the chaotic maps initial conditions
x_0, y_0
Set the chaotic maps parameters values
Send state variables value via USB communication
x_n, y_n
Calculate the next state variable value
x_{n+1}, y_{n+1}
Assign next state variable values
$x_n=x_{n+1}$, $y_n=y_{n+1}$

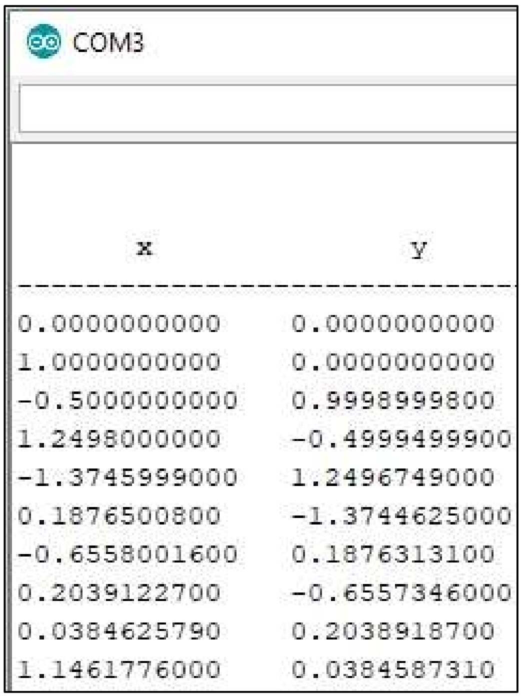
Figure 2 The flow chart of the microcontroller software

2.1. Lozi Discrete Chaotic Map

Lozi two-dimensional (2D) discrete-time CM introduced by Lozi in 1978 [19, 40]. The mathematical model of the Lozi CM is given in Eq. 1 [40]. Different outputs can be obtained by different values of *a* and *b* and initial conditions in the system. *n* is the discrete iteration step.

$$\begin{aligned} x_{n+1} &= 1 - a|x_n| + by_n \\ y_{n+1} &= bx_n \end{aligned} \tag{1}$$

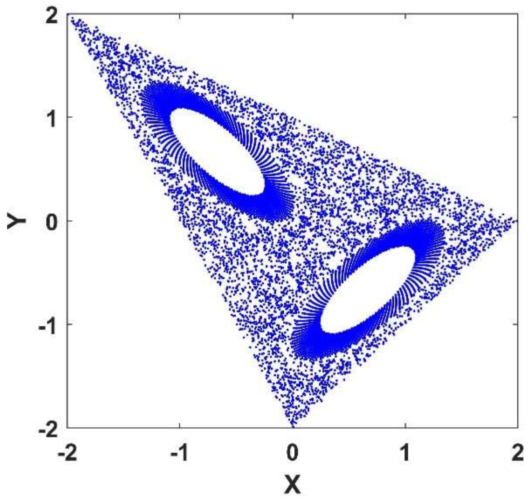
In this study, $a = 1.5$, $b = 0.9999$ and $(x_0, y_0) = (0, 0)$ are taken in [40]. Data received via USB port from the microcontroller output are saved to the file with the computer. Data are received from the microcontroller as given in Figure 3.



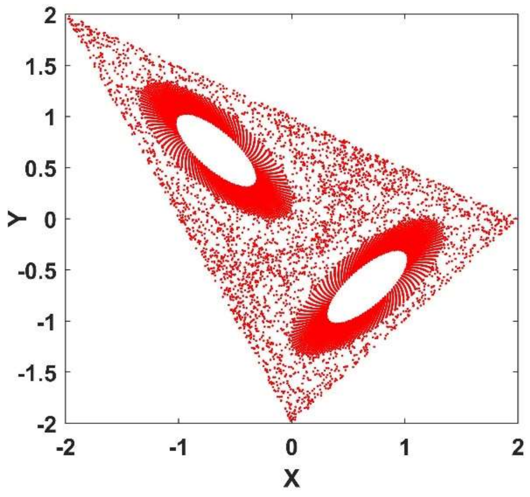
```
COM3

        x                   y
--------------------------------
0.0000000000        0.0000000000
1.0000000000        0.0000000000
-0.5000000000       0.9998999800
1.2498000000        -0.4999499900
-1.3745999000       1.2496749000
0.1876500800        -1.3744625000
-0.6558001600       0.1876313100
0.2039122700        -0.6557346000
0.0384625790        0.2038918700
1.1461776000        0.0384587310
```

Figure 3 Data received via USB port from the microcontroller output

The phase portraits (drawn in points) of the Lozi CM obtained from Matlab® numerical simulation and microcontroller output are given in Figure 4. As seen in Figure 4a and Figure 4b, the numerical calculation result and microcontroller based implementation results confirm each other.



(a)



(b)

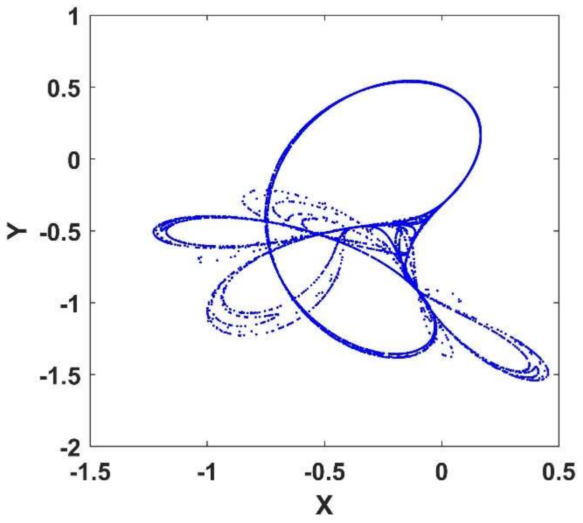
Figure 4 The phase portraits of the Lozi discrete CM (a) Matlab simulation result (b) Microcontroller output result

2.2. Tinkerbell Discrete Chaotic Map

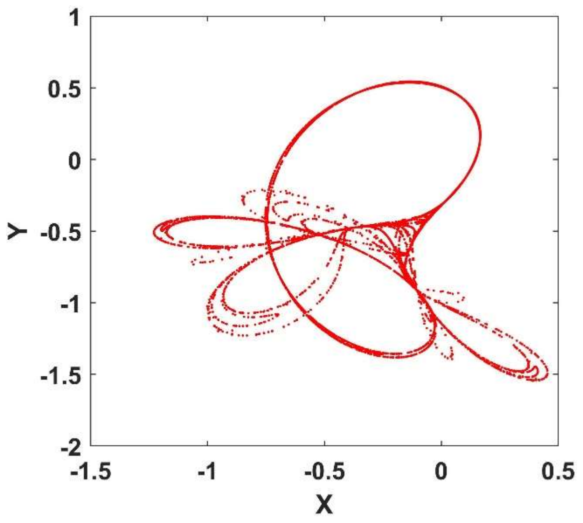
Tinkerbell is a two dimensional discrete time CM. Tinkerbell CM is given in Eq. 2 [18]. In Eq. 2, α, β, γ, and δ are system parameters.

$$\begin{aligned} x_{n+1} &= x_n^2 - y_n^2 + \alpha x_n + \beta y_n \\ y_{n+1} &= 2x_n y_n + \gamma x_n + \delta y_n \end{aligned} \tag{2}$$

In this study, $\alpha = 0.9$, $\beta = -0.6013$, $\gamma = 2$, $\delta = 0.5$ and initial conditions $(x_0, y_0) = (-0.72, -0.64)$ are taken [18]. The phase portraits (drawn in points) of the Tinkerbell CM obtained from Matlab® numerical simulation and microcontroller output are given in Figure 5. As seen in Figure 5 the numerical simulation result and microcontroller based implementation results confirm each other.



(a)



(b)

Figure 5 The phase portraits of the Tinkerbell discrete CM (a) Matlab simulation result (b) Microcontroller output result

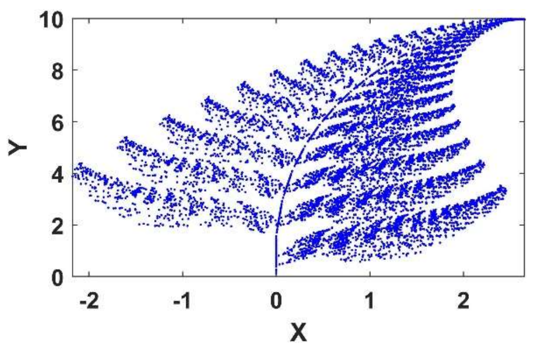
2.3. Barnsley Fern Discrete Chaotic Map

Barnsley Fern two-dimensional (2D) discrete-time CM introduced by Michael F. Barnsley [17]. Phase portrait of this CM is similar to fern. The mathematical expression of the Barnsley Fern CM is given in Eq. 3. In Eq. 3, *a*, *b*, *c*, *d*, and *e* are parameters. In the calculation of the CM, the values to be taken by the parameters differ according to the value taken by the random number p. In Eq. 4, the values of the parameters are given according to the value of the random number p which in range [0, 1] [17]. Initial conditions of the CM are $(x_0, y_0) = (0.1, 0.9)$. In this study, p and the other parameters values are taken as given in Eq. 4.

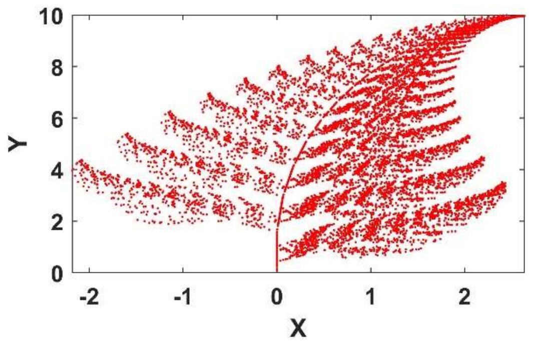
$$\begin{aligned} x_{n+1} &= ax_n + by_n + e \\ y_{n+1} &= cx_n + dy_n + f \end{aligned} \tag{3}$$

$$\begin{cases} a, b, c, e, f = 0, d = 0.16 & p < 0.01 \\ a = 0.2, b = -0.26, c = 0.23, & p < 0.08 \\ d = 0.22, e = 0, f = 1.6 & \\ a = -0.15, b = 0.28, c = 0.26 & p < 0.15 \\ d = 0.24, e = 0, f = 0.44 & \\ a = 0.85, b = 0.04, c = -0.04 & other\ p \\ d = 0.85, e = 0, f = 1.6 & values \end{cases} \tag{4}$$

The phase portraits (drawn in points) of the Barnsley Fern CM obtained from Matlab® numerical simulation and microcontroller output are given in Figure 6. As seen Figure 6 the numerical simulation result and microcontroller based implementation results confirm each other.



(a)



(b)

Figure 6 The phase portraits of the Barnsley Fern discrete CM (a) Matlab simulation result (b) Microcontroller output result

3. MICROCONTROLLER-BASED RNG IMPLEMENTATION BY USING THE DISCRETE CHAOTIC MAPS

Microcontroller-based RNG design is implemented by using the three different discrete two-dimensional CMs which are Lozi, Tinkerbell and Barnsley Fern in this part. For RNG design, the outputs of the *x* and *y* state variables of the CMs are first multiplied by 1000. Thus, the values of the variables are magnified. The output of the *x* and *y* state variables of the CMs are first converted to 32-bit single floating point binary number as in Figure 7.

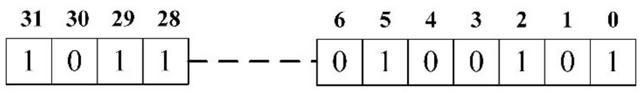


Figure 7 32-bit single floating-point binary format

In the RNG design, the first 24 bits (0,1,2, 3,…, 23) of the 32 bit floating point binary number obtained from CM outputs are used for pre-processing algorithm. The pre-processing algorithm is given schematically in Figure 8 and Figure 9. In the first case, the EXOR operation is performed mutually from the 0th bit of the *x* state variable of Lozi CM and the 23th bit of the *x* state variable of Tinkerbell CM (0 →23, 1 →22, 2 →21, …). The result obtained here is applied to EXOR processing with the 0st bit of the *x* of the Barnsley CM. The other bits are also applied to the same processing in order. In this way, the first 24-bit RNG output is obtained.

Then, used the similar algorithm, as given in Figure 9 for second 24-bit RNG output. For the second 24-bit RNG output, *y* state variables of the CMs are used this time. First, the EXOR operation is performed mutually from the 23th bit of the *y* state variable of Lozi CM and the 0th bit of the *y* state variable of Tinkerbell CM (23 →0, 22 →1, 21 →2, …). The result obtained here is applied to EXOR processing with the 23st bit of the *y* state variable of the Barnsley CM. The other bits are also applied to the same processing in order. In this way, the second 24 bit RNG output is obtained. With the algorithm given in Figure 8 and Figure 9, 48-bit RNG output is obtained.

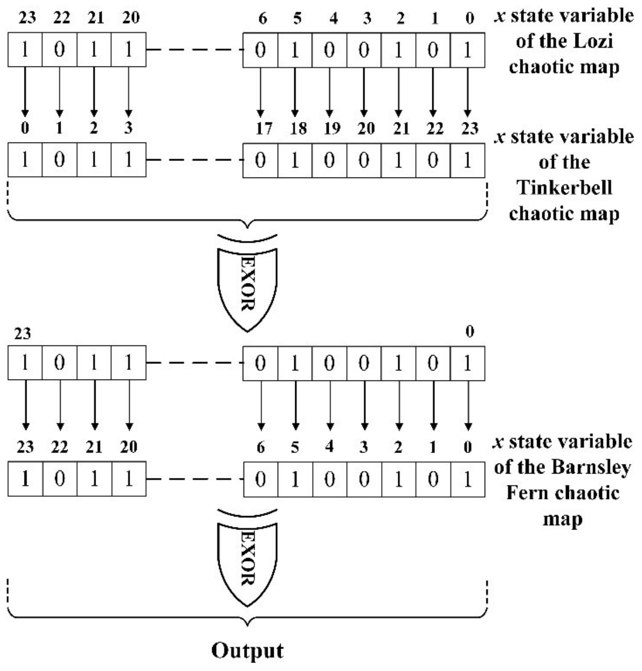

23 22 21 20
6 5 4 3 2 1 0
x state variable of the Lozi chaotic map
0 1 2 3
17 18 19 20 21 22 23
x state variable of the Tinkerbell chaotic map
EXOR
23
0
23 22 21 20
6 5 4 3 2 1 0
x state variable of the Barnsley Fern chaotic map
EXOR
Output

Figure 8 First 24-bit algorithm of the RNG output

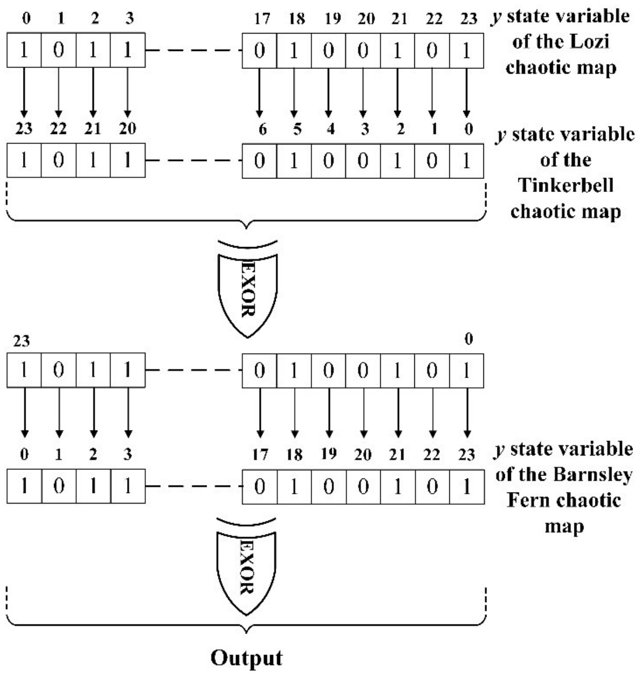

y state variable of the Lozi chaotic map
y state variable of the Tinkerbell chaotic map
EXOR
y state variable of the Barnsley Fern chaotic map
EXOR
Output

Figure 9 Second 24-bit algorithm of the RNG output

The subsequent 48-bit sequences are obtained from the chaotic maps output values obtained after every 95 iteration calculations. In this way, different values are obtained instead of close numbers that are produced consecutively.

The implemented RNG design is run on the microcontroller and obtained random numbers are tested for validity. Figure 10 shows the measurement set-up of microcontroller based RNG output signals with a computer-based oscilloscope. Signal sample of the random numbers obtained from the microcontroller output is given in Figure 11.

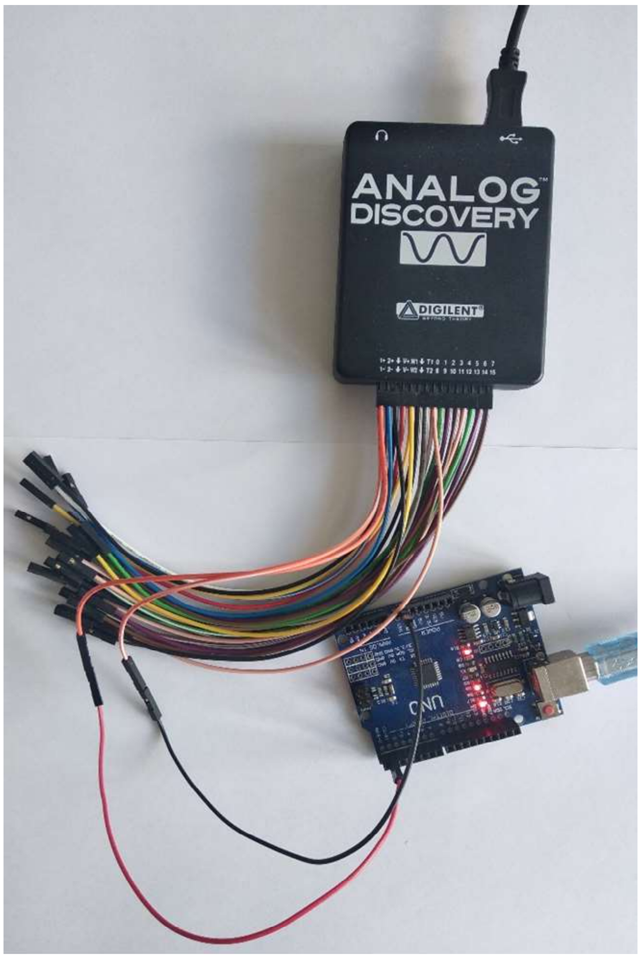


Figure 10 Measurement set-up of microcontroller based RNG output signal with a computer-based oscilloscope

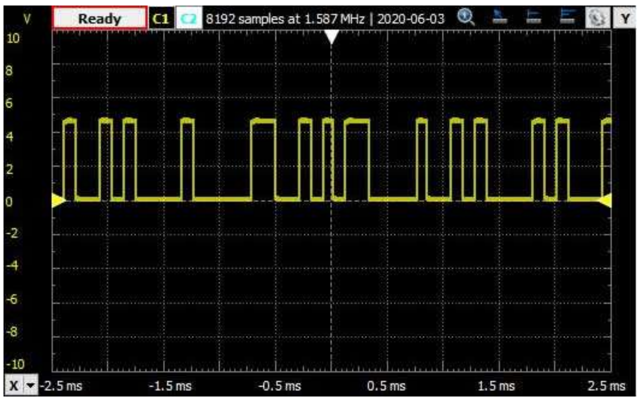


Figure 11 Signal sample of the random numbers obtained from the microcontroller

4. RANDOMNESS TESTS OF THE MICROCONTROLLER-BASED RNG

Various statistical tests are available to determine RNG success. Diehard, TestU01, AIS 31, FIPS and NIST 800-22 tests can be used to determine RNG performance. Diehard test package was introduced in 1996 by George Marsaglia. Diehard includes 15 statistical tests, but it has drawbacks and limitations, such as the sample sizes are not too large and the numbers to be tested must be in the form of 32-bit integers in the binary file [42]. AIS 31 test consists of 8 statistical tests. Today, this test is not very common in the literature to determine RNG performance [43]. TestU01 is empirical test package to determine randomness for RNGs. The TestU01 test also has a limitation that accepts only 32-bit entries [42].

Instead of Diehard, TestU01 and AIS 31 tests, the NIST 800-22 test is commonly used to determine RNG success in the literature. The NIST test suite covers many tests in other test packages. Besides the NIST test, FIPS test is also used in the literature. Therefore, in this study, NIST and FIPS tests were preferred to determine the random success rate of the implemented microcontroller-based RNG outputs.

4.1. NIST 800-22 Test

NIST SP800-22 is a commonly used statistical test package that measures the randomness of sequences produced by RNGs. The NIST 800-22 test consists of a total of 15 statistical randomness tests [44, 45]. The NIST 800-22 statistical test is a procedure that generates two hypotheses, H_0 (data random) or H_a (data not random). The significance level of the test is indicated by α. In the test, α is the probability that the test is not random when the sequence is really random [45]. The value of α is selected between [0.001 - 0.01]. In the NIST tests, the p value points out the degree of randomness and must be greater than the specified α parameter value in the NIST test. If $p \geq \alpha$ the sequence is random, if $p < \alpha$ the sequence is not random. The p takes a value between 0 and 1. If p is closer to 1, the degree of randomness of the sequence is better [41, 45]. NIST tests are perform with standard normal and chi-square (χ^2) distribution reference. The standard normal distribution is used to compare the test statistic value obtained from RNG with the expected value. The chi-square (χ^2) distribution is used to compare the value of observed frequencies of a sample sequence measure to the expected frequencies of distribution [45]. The calculation formulas of all these tests can be found in detail in Ref [45]. For some of the NIST 800-22 tests, the number of sequences obtained from the RNG output should be sufficient. The number of sequences to be tested must be between 10^3 and 10^7 [41, 45].

NIST 800-22 test was applied to the implemented RNG. In the test, α value is taken as 0,01. For the test, 1,000,000 bits are taken from the RNG output. The NIST test results are given in Table 1. When Table 1 is examined, it is seen that the RNG outputs are successful in all NIST tests. Because p value was obtained greater than 0,01 in all tests. In the "Random-excursions" and "Random-excursions-variant" tests, only the results of at x = -4 and x = -9 are given in Table 1, in order for the table to not too long. But in all the tests, the results were successful in all values of x.

Table 1
NIST tests result of implemented microcontroller-based RNG by using the discrete chaotic maps

Test name	*p* value	Result
Frequency	0,9729	Valid
Block frequency	0,7410	Valid
Runs	0,9077	Valid
Longest run	0,6196	Valid
Rank	0,9686	Valid
Discrete Fourier transform	0,2402	Valid
Nonoverlapping template matching (B=001111111)	0,2040	Valid
Overlapping template matching	0,5170	Valid
Universal statistical	0,0668	Valid
Linear complexity	0,7258	Valid
Serial	0,7412	Valid
	0,9191	Valid
Approximate entropy	0,9163	Valid
Cumulative sums	0,6718	Valid
Random-excursions-test (x= –4)	0,8830	Valid
Random-excursions-variant test (x = –9)	0,4597	Valid

4.2. FIPS Test

In addition to the NIST 800-22 test, the FIPS test was also applied for the RNG success determine. The FIPS test consists of four separate tests: Monobit test, Poker test, Run test and Long Run test. For every FIPS test, 20,000 bits (1 and 0) are required. If any of the four tests fail, the RNG does not pass the FIPS test. Each test result has a required range to succeed [46]. The purpose of the "monobit" test is to determine the distribution of '0' and '1 'in the bit sequence. The purpose of the "poker" test is to determine if the number of repetitions of the specified block pieces is within the required range. The purpose of the "run" test is to determine if all consecutive bit sequences of all '1' or '0' values that are part of the 20,000 bits stream are within the desired range. The purpose of the "long run" test is to determine if there are more than 34 consecutive bits in the "run" test [46, 47].

For the FIPS test, 20,000 bits are taken from the implemented RNG output. The FIPS test requirements and results are given in Table 2. When Table 2 is examined, it is seen that the RNG outputs are successful in all FIPS tests.

Table 2
FIPS tests result of implemented microcontroller-based RNG by using the discrete chaotic maps

Test name	Required condition	Test result value	Result
Monobit	9654 < X <10346	9963	Valid
Poker	1.03 < X < 57.4	51	Valid
Run	Consecutive 1 and 0 block length = 1 2267 < X < 2733	2479	Valid
	Consecutive 1 and 0 block length = 2 1079 < X < 1421	1190	Valid
	Consecutive 1 and 0 block length = 3 502 < X < 748	681	Valid
	Consecutive 1 and 0 block length = 4 223 < X < 402	314	Valid
	Consecutive 1 and 0 block length = 5 90 < X < 223	156	Valid
	Consecutive 1 and 0 block length = 6 and more 90 < X < 223	144	Valid
Long Run	All consecutive blocks of "1" and "0" must be less than 34.	No consecutive blocks greater than 34 were found.	Valid

5. CONCLUSIONS AND FUTURE WORK

In this study, firstly, two-dimensional (2D) Lozi, Tinkerbell and Barnsley Fern discrete CMs are implemented based on microcontroller. When Figure 4, Figure 5 and Figure 6 are examined, the numerical simulation results performed in the Matlab® program and microcontroller output results confirm each other. Then, microcontroller based random number generator with the pre-processing algorithm given in Figure 8 and Figure 9 is implemented by using the three different discrete CMs. The RNG outputs are applied to the NIST 800-22 and FIPS tests and passed the all tests.

Consequently, implemented microcontroller-based RNG by using CMs can be used in a variety of engineering applications. In future works, chaos based encryption and communication applications can be realized with the RNG outputs obtained in this study. Similarly, the RNG outputs can be used for random number needs in various optimization algorithms. In this way, optimization success can be increased.

Acknowledgements

The author would like to thank the SAUJS editors and reviewers who reviewed the study.

Funding

The author received no financial support for the research, authorship, and/or publication of this paper.

The Declaration of Conflict of Interest/ Common Interest

No conflict of interest or common interest has been declared by the author.

The Declaration of Ethics Committee Approval

The author declares that this document does not require an ethics committee approval or any special permission.

The Declaration of Research and Publication Ethics

The author of the paper declares that he complies with the scientific, ethical and quotation rules of SAUJS in all processes of the paper and that he does not make any falsification on the data collected. In addition, he declares that Sakarya University Journal of Science and its editorial board have no responsibility for any ethical violations that may be encountered, and that this study has not been evaluated in any academic publication environment other than Sakarya University Journal of Science.

REFERENCES

[1] Y. Liu and X. Tong, “Hyperchaotic system-based pseudorandom number generator”, IET Information Security, vol. 10, no. 6, pp. 433-441, 2016.

[2] İ. Koyuncu and A.T. Özcerit, “The design and realization of a new high speed FPGA-based chaotic true random number generator”, Computers and Electrical Engineering, Vol. 58, pp. 203-214, 2017.

[3] F. Yu, L. Li, Q. Tang, S. Cai, Y. Song and Q. Xu, “A survey on true random number generators based on chaos”, Discrete Dynamics in Nature and Society, Vol. 2019, Article ID: 2545123, 2019.

[4] X.Y. Wang and Y.X. Xie, “A design of pseudo-random bit generator based on single chaotic system”, International Journal of Modern Physics C, Vol. 23, no. 3, pp.1250024-1 – 1250024-11, 2012.

[5] S. Ergün and S. Özoğuz, “Truly random number generators based on non-autonomous continuous-time chaos”, International Journal of Circuit Theory and Applications, Vol. 38, pp. 1-24, 2010.

[6] M.E. Yalçın, J.A.K. Suykens and J. Vandewalle, “True random bit generation from a double-scroll attractor”, IEEE Transactions on Circuit and Systems-I: Regular Papers, Vol. 51, no. 7, pp. 1395-1404, 2004.

[7] S. Vaidyanathan, A. Akgul, S. Kaçar and U. Çavuşoğlu, “A new 4-D chaotic hyperjerk system, its synchronization, circuit design and applications in RNG, image encryption and chaos-based steganography”, The European Physical Journal Plus, Vol. 133, Article number: 46, 2018.

[8] A. Akgul, C. Arslan and B. Arıcıoglu, “Design of an interface for random number generators based on integer and fractional order chaotic systems”, Chaos Theory and Applications, Vol. 01, no. 1, pp. 1-18, 2019.

[9] B.C. Bao, H.Z. Li, X. Zhang and M. Chen, “Initial-switched boosting bifurcations in 2D hyperchaotic map”, Chaos, Vol. 30, no. 3, 033107, 2020.

[10] D. Lambic, “A novel method of S-box design based on discrete chaotic map”, Nonlinear Dynamics, Vol. 87, pp. 2407-2413, 2017.

[11] A.A. Alzaidi, M. Ahmad, M.N. Doja, E.A. Solami and M.M.S. Beg, “A new 1D chaotic map and β-hill climbing for generating substitution-boxes”, IEEE Access, Vol. 6, pp. 55405-55418, 2018.

[12] O. Alpar, “Analysis of a new simple one dimensional chaotic map”, Nonlinear

Dynamics, vol. 78, no. 2, pp. 771-778, 2014.

[13] Z. Hua and Y. Zhou, "One-dimensional nonlinear model for producing chaos", IEEE Transactions on Circuit and Systems-I: Regular Papers, Vol. 65, no. 1, pp. 235-246, 2018.

[14] L. Liu and S. Miao, "A new simple one-dimensional chaotic map and its application for image encryption", Multimedia Tools and Applications, Vol. 77, pp. 21445-21462, 2018.

[15] M. Hènon, "A two-dimensional mapping with a strange attractor", Communications in Mathematical Physics, Vol. 50, pp. 69-77, 1976.

[16] L.Q. Chen, "An open-plus-closed-loop control for discrete chaos and hyperchaos", Physics Letters A, Vol. 281, pp. 327-333, 2001.

[17] M.F. Barnsley, "Fractals Everywhere", Academic Press, USA, pp. 85-91, 1993.

[18] A. Ouannas, A.A. Khennaoui, S. Bendoukha, T.P. Vo, V.T. Pham and V.V. Huynh, "The fractional form of the Tinkerbell map is chaotic", Applied Sciences, Vol. 8, Article ID: 2640, 2018.

[19] R. Lozi, "Un attracteur ètrange (?) du type attracteur de hènon", Journal De Physique, Vol. 39, no. 8, pp. C5-9, 1978.

[20] Y. Xiao, K. Sun and S. He, "Constructing chaotic map with multi-cavity", The European Physical Journal Plus, Vol. 135, Article number: 21, 2020.

[21] Z. Hua, Y. Zhou, C.M. Pun and C.L.P. Chen, "2D sine logistic modulation map for image encryption", Information Sciences, Vol. 297, pp. 80-94, 2015.

[22] Z. Liu, T. Xia and J. Wang, "Fractional two-dimensional discrete chaotic map its applications to the information security with elliptic-curve public key cryptography", Journal of Vibration and Control, Vol. 24, no. 20, pp. 4797-4824, 2018.

[23] Y. Peng, K. Sun, D. Peng and W. Ai, "Dynamics of a higher dimensional fractional-order chaotic map", Physica A, Vol. 525, pp. 96-107, 2019.

[24] A. Alghafis, N. Munir, M. Khan and I. Hussain, "An encryption scheme based on discrete quantum map and continuous chaotic system", International Journal of Theoretical Physics, Vol. 59, pp. 1227-1240, 2020.

[25] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps", International Journal of Bifurcation and Chaos, Vol. 8, no. 6, pp. 1259-1284, 1998.

[26] X. Liao, X. Li, J. Pen and G. Chen, "A digital secure image communication scheme based on the chaotic Chebyshev map", International Journal of Communication Systems, Vol. 17, pp. 437-445, 2004.

[27] S. Papadimitriou, A. Bezerianos, T. Bountis and G. Pavlides, "Secure communication protocols with discrete nonlinear chaotic maps", Journal of Systems Architecture, Vol. 47, pp. 61-72, 2001.

[28] M. Zhang and X. Tong, "A new chaotic map based image encryption schemes for several image formats", The Journal of Systems and Software, Vol. 98, pp. 140-154, 2014.

[29] R. Li, Q. Liu and L. Liu, "Novel image encryption algorithm based on improved logistic map", IET Image Processing, Vol. 13, no. 1, pp. 125-134, 2019.

[30] C. Pak, J. Kim, K. An, C. Kim, K. Kim and C. Pak, "A novel color image LSB steganography using improved 1D chaotic map", Multimedia Tools and

Applications, Vol. 79, pp. 1409-1425, 2020.

[31] Y. Naseer, D. Shah and T. Shah, "A novel approach to improve multimedia security utilizing 3D mixed chaotic map", Microprocessors and Microsystems, Vol. 65, pp. 1-6, 2019.

[32] D. Herbadji, A. Belmeguenai, N. Derouiche and H. Liu, "Colour image encryption scheme based on enhanced quadratic map", IET Image Processing, Vol. 14, no. 1, pp. 40-52, 2019.

[33] M.A. Dastgheib and M. Farhang, "A digital pseudo-random number generator based on sawtooth chaotic map with a guaranteed enhanced period", Nonlinear Dynamics, Vol. 89, pp. 2957-2966, 2017.

[34] E. Avaroğlu, "Pseudorandom number generator based on Arnold cat map and statistical analysis", Turkish Journal of Electrical Engineering & Computer Sciences, Vol. 25, pp. 633-643, 2017.

[35] D. Lambìc, M. Nikolic, "Pseudo-random number generator based on discrete-space chaotic map", Nonlinear Dynamics, Vol. 90, pp. 223-232, 2017.

[36] A.V. Tutueva, E.G. Nepomuceno, A.I. Karimov, V.S. Andreev and D.N. Butusov, "Adaptive chaotic maps and their application to pseudo-random numbers generation", Chaos, Solitons and Fractals, Vol. 133, 109615, 2020.

[37] O. Garasym, I. Taralova and R. Lozi, "New Nonlinear CPRNG Based on Tent and Logistic Maps". In: J. Lü, X. Yu, G. Chen, W. Yu (eds) Complex Systems and Networks. Understanding Complex Systems, Springer, Berlin, Heidelberg, pp.131-161, 2016.

[38] Magfirawaty, M.T. Suryadi, K. Ramli, "On the design of henon and logistic map-based random number generator", IOP Conference Series: Journal of Physics: Conference Series, Vol. 893, 012060, 2017.

[39] S. Ergün, S. Tanrıseven, "Random number generators based on discrete-time chaotic maps", IEEE EUROCON 18th International Conference on Smart Technologies, pp. 1-4, 2019.

[40] L. Moysis and A.T. Azar, "New discrete time 2D chaotic maps", International Journal of System Dynamics Applications, Vol. 6, no. 1, pp.77-104, 2017.

[41] A.M. Garipcan, E. Erdem, "Implementation and performance analysis of true random number generator on FPGA environment by using non-periodic chaotic signals obtained from chaotic maps", Arabian Journal for Science and Engineering, Vol. 44, pp. 9427-9441, 2019.

[42] P. L'ecuyer, R. Simard, "TestU01: A C library for empirical testing of random number generators", ACM Transactions on Mathematical Software, Vol. 33, no. 4, 22, 2007.

[43] R. Santoro, O. Sentieys and S. Roy, "On-line monitoring of random number generators for embedded security", IEEE International Symposium on Circuit and Systems, pp. 3050-3053, 2009.

[44] D. Lihua, Z. Yong, J. Ligang and H. Xucang, "Study on the pass rate of NIST SP800-22 statistical test suite", IEEE Tenth International Conference on Computational Intelligence and Security", pp. 402-404, 2014.

[45] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray and S. Vo, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", National Institute of Standards and Technology (NIST), 2010.

[46] L. Min, T. Chen and H. Zang, “Analysis of FIPS 140-2 test and chaos-based pseudorandom number generators”, Chaotic Modeling and Simulation, Vol. 2, pp. 273-280, 2013.

[47] İ. Koyuncu, “Kriptolojik uygulamalar için FPGA tabanlı yeni kaotik osilatörlerin ve gerçek rastgele sayı üreteçlerinin tasarımı ve gerçeklenmesi”, PhD Thesis, Sakarya University, Institute of Science, Sakarya, 2014.