Journal of Computations & Modelling, vol.3, no.4, 2013, 115-136
ISSN: 1792-7625 (print), 1792-8850 (online)
Scienpress Ltd, 2013

Chaotic Random Bit Generator Realized with a Microcontroller

Christos K. Volos[1]

Abstract

In the last decades an interesting relationship between chaos theory and cryptography has been developed. As a result of this close relationship several chaos-based cryptosystems, which play important role especially in military operations because of the significant strategic advantage that these systems provide, have been put forward. This work, presents a novel Chaotic Random Bit Generator (CRBG), which is realized by the Arduino, an open-source physical computing platform based on a simple microcontroller board. The proposed CPRBG, uses the XOR function, in the bit sequences, that are produced by two Logistic maps with different initial conditions and systems' parameters, for achieving better results concerning the "randomness" of the produced bits sequence. The generated by the proposed CRBG bit sequences are subjected to the most stringent tests of randomness, the FIPS-140-2 suite tests, to detect the specific characteristics which are expected of random bit sequences.

[1] Division of Mathematics and Engineering Studies, Department of Military Science, Hellenic Military Academy, Vari 16673, Greece. E-mail: chvolos@gmail.com

Article Info: *Received* : July 1, 2013. *Revised* : August 30, 2013.
Published online : December 1, 2013.

Keywords: Chaotic random bit generator, chaos, Logistic map, discrete nonlinear map, XOR function, Arduino microcontroller, FIPS statistical tests.

1 Introduction

New rapid developments in communication technologies, such as in Internet, in mobile networks, and in military networks, depend upon the generation of sufficient size, unpredictable quantities for ensuring the information security. Especially, the “randomness” of the generated quantities in the sense that probability of any particular value being selected must be sufficiently small to preclude an adversary from gaining advantage through optimizing a search strategy based on such probability, plays a crucial role for the safety of the communication network. For this reason various techniques including the keystream in the one-time pad, the secret key in the DES encryption algorithm, the primes *p*, *q* in the RSA encryption and digital signature schemes, the private key a in the DSA, and the challenges used in challenge-response identification systems, have been developed and are commercial available.

So, in the last two decades many research teams tried to design devices or algorithms which satisfy the basic demand of generating unpredictable quantities. These devices or algorithms are called Random Bit Generators (RBGs). As a definition one could say that ([1]):
“*A random bit generator is a device or algorithm which outputs a sequence of statistically independent and unbiased binary digits.*”

Until now a great number of RBGs have been proposed. All these can be classified, based on the source of the randomness, into three major types ([2]):

- True Random Bit Generators (TRBGs),
- Pseudo-Random Bit Generators (PRBGs) and
- Hybrid Random Bit Generators (HRBGs).

In the first type, TRBGs require a naturally occurring source of randomness, which comes from an unpredictable natural process in a physical or hardware device. However, designing a hardware device to exploit this randomness and produce a bit sequence that is free of biases and correlations is a difficult task. Additionally, for most cryptographic applications, the generator must not be subject to observation or manipulation by an adversary. Due to the fact that the TRBGs are based on natural sources of randomness are subject to influence by external factors which cause malfunctions.

All the above mentioned difficulties of obtaining uniform random sequences from TRNG lead many research teams to the design of pseudorandom bit generators. A PRBG is a deterministic algorithm which, outputs a binary sequence of length $l >> k$ that “appears” to be random, if a binary sequence of length k is given. The input to the PRBG is called the seed, while the output of the PRBG is called a pseudorandom bit sequence. This bit sequence is not truly random in that it is completely determined by a relatively small set of initial values. Also, PRBGs are very important in practice for their speed in number generation, their portability and their reproducibility, and they are thus central in applications such as in cryptography, in decision making, in simulating natural phenomena and in sampling data ([3, 4]). So, a good PRBG for the previous mentioned applications should possess three very important characteristics: long period, high speed and randomness.

Nevertheless, it is obvious that in PRBGs due to the fact that the output is a function of the seed state, the actual entropy of the output can never exceed the entropy of the seed. Hence, the randomness level of the pseudo-random numbers depends on the level of randomness of the seed. Thus, HRNGs have been proposed to use a random generator as a seed generator and expand it. A seed generator is a hardware-based RNG with or without user’s interaction, such as mouse movements, random keystrokes, or hard drive seek times.

Furthermore, in the last decades, nonlinear systems and especially chaotic systems have aroused tremendous interest because of their applications in many scientific fields, such as in physics, sociology and economic theory but also of many other interesting applications such as in secure communication schemes and cryptography ([5-7]).

Nowadays it is known, that chaos and cryptography have a structural relationship because they have many similar properties (Table 1) ([8]). As a result of this relationship several chaotic cryptosystems have been presented ([9-11]). One of the most interesting way through which chaotic cryptosystems can be realized is via the implementation of Chaotic Random Bit Generator (CRBG). So, several ideas of designing CRBG, by using either continuous or discrete chaotic systems, have been proposed by academia and industry ([12-20]).

Table 1: Properties of Chaos and its analogous properties of Cryptography

Chaos	**Cryptography**
Ergodicity	Confusion
Sensitivity to initial conditions / system parameters	Diffusion with small changes in plaintext / secret keys
Mixing property	Diffusion with a small change within one block of the plaintext
Deterministic dynamics	Deterministic pseudo randomness
Structural complexity	Algorithm Complexity

In the present work a novel CRBG, which is realized by an open-source physical computing platform based on a simple microcontroller board, the well-known Arduino, is discussed. The microcontroller runs side-by-side two Logistic maps, working in different chaotic regimes because of having different initial conditions and systems' parameters. The produced, by the proposed CRBG, bit sequences are results of the XOR function in the outputs of the two chaotic Logistic maps, which are subjected to de-skewing technique to extract unbiased bits with no correlation. Also, the algorithm, in which the microcontroller has programmed, is very simple, so the CRBG is quite fast. The use of two chaotic discrete maps increases the complexity in the random bit generation, as it is confirmed by the statistical test suite, FIPS-140-2, and hence becomes difficult for an intruder to extract information about the system.

This paper is organized as follows. In Section 2, the Logistic map, which is the base of this CRBG, is presented. Section 3 describes the proposed CRBG block by block as such its realization with the Arduino. In Section 4 the results of the statistical tests of FIPS-140-2 which assess the statistical properties of the CRBG, are presented. Finally, Section 5 includes the conclusion remarks of this work.

2 The Logistic Map

Since the discovery of deterministic chaos in the mid 1960s, the field of non-linear dynamical systems has attracted the attention of researchers worldwide. It is known that the most important aspects of chaotic behavior should appear in systems of lowest dimension and especially in discrete-time dynamical systems. Discrete-time dynamical systems are a particular type of non-linear dynamical systems generally described as an iterative map $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}^n$ by the state equation:

$$x_{k+1} = f\,(x_k),\ \ k = 0, 1,\ 2, \dots \tag{1}$$

where n is the dimensionality of the state-space, k denotes the discrete time, $x_k \in \mathbb{R}^n$ is the state of the system at time k, while x_{k+1} denotes the next state.

The Logistic map, a well-known iterative equation, which is described by the following equation:

$$x_{n+1} = rx_n(1 - x_n),\ \ 0 \le x \le 1 \tag{2}$$

is one of the most studied, one-dimensional, discrete chaotic maps because of its simplicity. As pointed out by May, this map may be thought of as a simple idealized ecological model for the yearly variations in the population of an insect species ([21]). This map was also proposed as pseudo-random number generator by Von Neumann ([22]) partly because it had a known algebraic distribution so that the iterated values could be transformed to a uniform distribution. Over the years, many other realizations of random number generators, based on various forms of the Logistic equation, has been proposed ([23-28]).

Furthermore, Fiegenbaum ([29, 30]) reported some of the universal quantitative features (Table 1), which became the hallmark of the contemporary study of chaos. For this reason Logistic map possesses great potential for various cryptographic applications such as image encryption ([31, 32]), public key cryptography ([33]), block cipher ([34]), and hash function ([35]).

In more details, the parameter r in Eq. (2) varies in the interval $[0, 4]$ so that x_{n+1} maps the unit interval into the unit interval. Fig.1 shows the map function of x_{n+1} as a function of x_n, e.g. for $r = 3.99$ and $x_0 = 0.4$. From this plot the symmetry of the Logistic map about the mid point of the interval $[0, 1]$, is concluded. For the Logistic map of Eq.(2) the following regions, depending on the value of r, may be considered.

- For $r < 1$, x decays to a fixed point ($x \to 0$). That is, for any value of the seed x_0 between zero and one the x_n approaches the value of zero exponentially.

- For $1 \leq r < 3$, the previous fixed point loses its stability and another fixed $(x = 1 - 1/r)$ appears.
- For $3 \leq r \leq 4$, the Logistic map presents a more complex behavior (such as repeated period doubling, appearance of odd periods etc) which finally is leading to chaos.

In Figure 2 this rich dynamical behavior in the third region, which is mainly characterized by the very interesting period-doubling route from periodic to chaotic behavior, is illustrated. This so-called "Bifurcation diagram" of Figure 2 is also a very common perspective in nonlinear dynamics, being in this case a plot of the steady-state behavior of Eq. (2) with respect to the bifurcation parameter *r*.

The first bifurcation, as it is shown in Figure 2, occurs at the value of $r = 3$, followed by further doublings at shorter and shorter intervals of r until the period goes to infinity at $r_\infty = 3.5699$ ...., signifying chaos. Also, various periodic windows interspersed beyond r_∞, is observed, in which the behavior returns to a normal periodic one, quickly followed again by bifurcations to an infinite period.

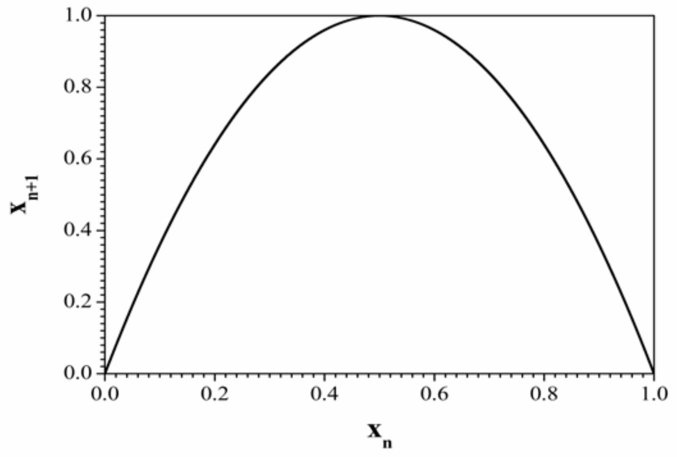

Figure 1: The map function of the Logistic equation (1), for $r = 3.99$ and $x_0 = 0.4$.

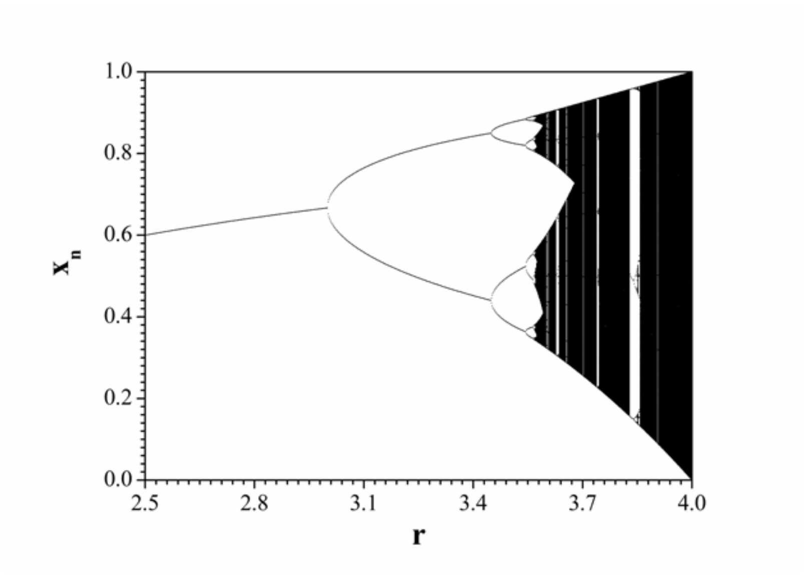

Figure 2: Bifurcation diagram of x_n vs. parameter r,
illustrating the period-doubling route to chaos

So, for $r_\infty > 3.5699$ ... the Logistic map shows a strange complex behavior (the so-called chaotic behavior) where map function never repeats its history. This is evident from Fig.3 where no periodicity arises, for $r = 3.99$ and $x_0 = 0.4$.

Finally, in Fig.4 the well-known Lyapunov exponent:

$$\lambda = lim_{n\to\infty} \sum_{i=1}^{n} ln|f'(x_i)| \tag{3}$$

$$f'(x) = r - 2rx \tag{4}$$

as a function of parameter r, is displayed. As it is known from the nonlinear theory a positive Lyapunov exponent indicates chaos. So, Figure 4 confirms the Logistic map's dynamical behavior as found from the bifurcation diagram (Figure 2).

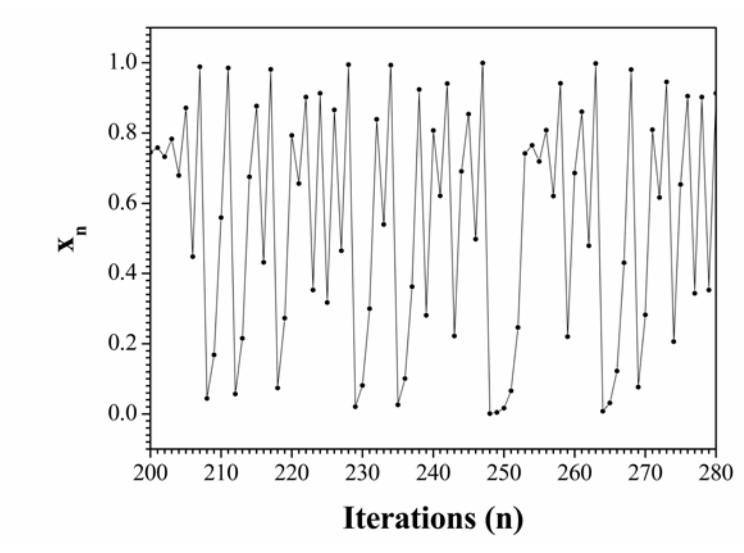

Figure 3: Variable x vs. n, for $r = 3.99$ and $x_0 = 0.4$

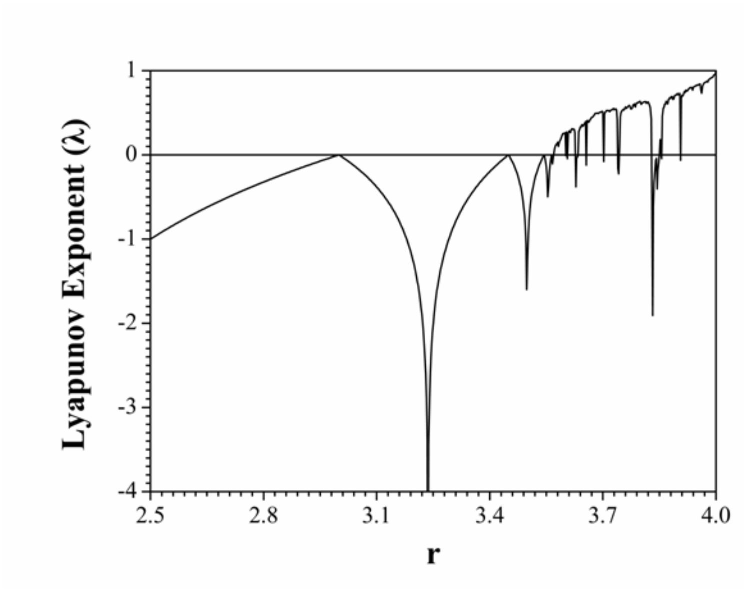

Figure 4: Lyapunov exponent (λ) vs. parameter *r*

3 The Chaotic Random Bit Generator

The main feature of the larger number of chaotic random bit generators, which have been presented, is that they work only in a software environment, where everything is ideal. For this reason, in this work, a step forward to the realization of a CRBG via a microcontroller is done. In general, the microcontroller combines the software programming with the hardware (processor) for doing a specific job. So, it is very interesting to find out the strengths and the weaknesses of the specific proposal because the microcontroller gives us the opportunity for using CRBGs in many applications, such as in robotics, in cryptography etc.

As it is mentioned the well-known open-source Arduino prototyping platform was used in this work for realizing the proposed CRBG. Arduino was chosen because, among all the other advantages it has, it is probably the most commercial platform which is used in a great number of applications. So, in this section the proposed CRBG is presented block by block by presenting also its realization with the microcontroller.

3.1 The Description of the CRBG

As it is referred in the previous section, various forms of the Logistic map have been used, especially in the last decade, in the design of chaotic random bit generators ([23-28]). Especially, in [28] Li et al. design and analyze a pseudo-random bit generator based on a piecewise-linear map. Authors by choosing the mean of the x_n values assure the generating of the same numbers of bits according to the following formula:

$$b_n = \begin{cases} 0, if\ x_n \leq x_{cr} \\ 1, if\ x_n > x_{cr} \end{cases} \tag{5}$$

where $x_{cr} = \bar{x}$ denotes the mean value and b_n is the bit generated by the *n*-th iteration of the Logistic map.

In this work, the above mentioned technique for producing bit sequences by the Logistic map, with x_{cr} different from ($x_{cr} \neq \bar{x}$), has been adopted. Also, the proposed CRBG (Fig.5) is based on two Logistic Maps (LM) of Eq.(1):

$$\begin{cases} \text{LM1}: x_{n+1} = r_1 x_n (1 - x_n) \\ \text{LM2}: y_{n+1} = r_2 y_n (1 - y_n) \end{cases} \tag{6}$$

having different parameters (r_1, r_2) and (x_{cr1}, x_{cr2}) while maps start from random independent initial conditions: (x_0, y_0) with $x_0 \neq y_0$. So, the two iterative maps of Eqs.(6), consist the first block of the proposed CRBG. By using Eq.(5), with $x_{cr} \neq \bar{x}$, two different chaotic bit sequences (b_1, b_2) are produced.

The second block of the proposed CRBG relies on extracting unbiased bits with no correlation from the two defective Logistic maps. For this purpose one of the most known de-skewing technique, the Von Neumann technique ([36]) has been used. Von Neumann proposed a digital post-processing that balances the distribution of bits. Post-processing converts non-overlapping pairs of bits into output bits by converting the bit pair "01" into an output "0", converting "10" into an output "1", while the pairs "11" and "00" are discarded. This technique is very easily implemented but it decreases throughput of generating approximately 1 bit from 4 bit.

Finally, the third block generates the bit sequence (σ_i) by using the XOR function, in the outputs of the second block $(\sigma_i = f_1 \oplus f_2)$. This technique provides, as it is shown in the next Section, better results concerning the "randomness" of the produced bit sequences by the proposed CRBG.

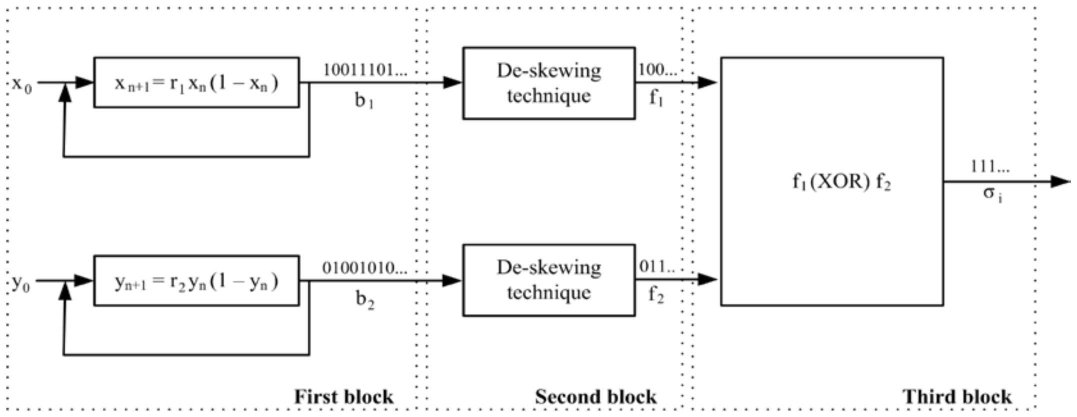

Figure 5: The schematic block diagram of the proposed Chaotic Random Bit Generator

3.2 The Realization of the CRBG with the Microcontroller

The proposed CRBG was realized by using an open-source Arduino prototyping platform made up of an Atmel AVR processor (microcontroller), based on flexible, easy-to-use hardware and software ([37]). This platform has 14 digital input/output pins, 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button (Figure 6). The Arduino is connected to the computer through the USB port and programmed using the language “Wiring” which is similar to C++. The program, which is known as “sketch” is uploaded into the microcontroller using an Integrated Development Environment (IDE).

So, in this work, the microcontroller outputs a random bits sequence by programming it with the sketch (Listing 1), which implements the proposed CRBG. The produced bitstream by the Arduino was captured to a text file by using the HyperTerminal program on Windows (Figure 7). This is the simplest method for writing data from the Arduino to the serial port and save them to a file.

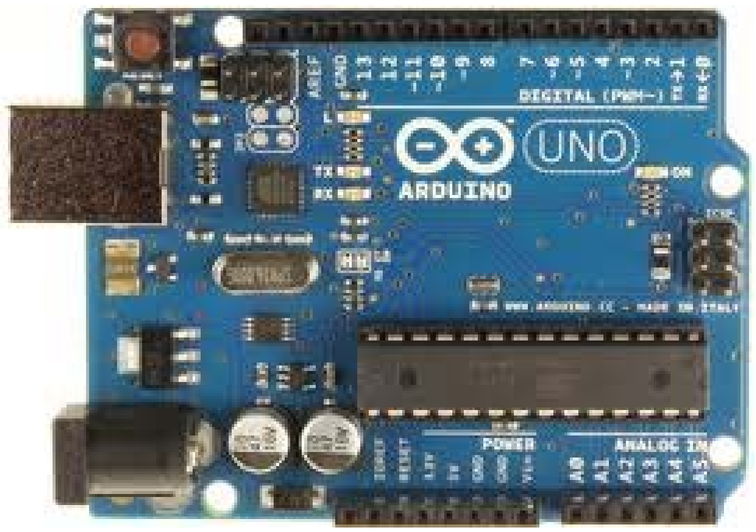

Figure 6: Arduino Uno prototyping platform

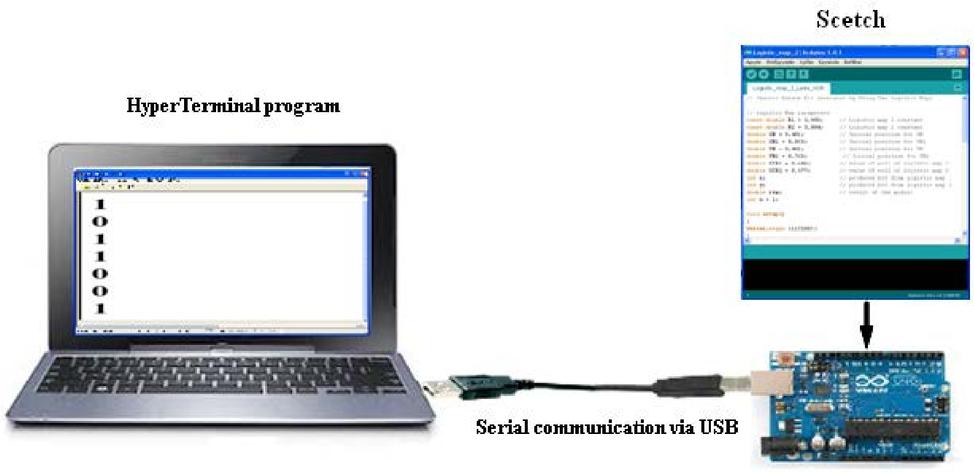

Figure 7: The setup of the proposed CRBG

4 Statistical Tests

The invention of a foolproof source of random numbers it is not an easy task. In order to gain the confidence that a newly developed random bit generator is cryptographically secure, it should be subjected to a variety of statistical tests designed to detect the specific characteristics expected of truly random sequences. There are several options available in the bibliography for analyzing the randomness of the newly developed random bit generators. The four most popular options are:

(i) the FIPS-140-2 (Federal Information Processing Standards) suite of statistical tests of the National Institute of Standards and Technology (NIST) ([38]),

(ii) the DIEHARD suite of statistical tests ([39]),

(iii) the Crypt-XS suite of statistical tests ([40]) and

(iv) the Donald Knuth’s statistical tests set ([41]).

In this work the “randomness” of the produced bit sequences, by the proposed CRBG, is analyzed by using the most stringent tests of randomness: the

FIPS-140-2 suite of statistical tests. The results of the use of the four statistical tests, Monobit test, Poker test, Runs test, and Long run test, which are part of the FIPS-140-2, are presented in details. According to FIPS-140-2, the examined CRBG will produce a bitstream, $b_i = b_0, b_1, b_2, \ldots, b_{n-1}$, of length n (at least 20,000 bits), which must satisfy the following standards.

- Monobit Test: The number n_1 of 1's in the bitstream must be $9725 < n_1 < 10275$.
- Poker Test: This test determines whether the sequences of length n ($n = 4$) show approximately the same number of times in the bitstream. The bounds of this statistic are then $2.16 < x_3 < 46.17$.
- Runs Test: This test determines whether the number of 0's (Gap) and 1's (Block) of various lengths in the bitstream are as expected for a random sequence ([38]).
- Long Run Test: This test is passed if there are no runs longer than 26 bits.

```
// Chaotic Random Bit Generator by Using Two Logistic Maps
// Logistic Map parameters
const double R1 = 3.990;     // Logistic map 1 constant
const double R2 = 3.984;     // Logistic map 2 constant
double XN = 0.400;           // Initial position for XN
double XN1 = 0.800;          // Initial position for XN1
double YN = 0.500;           // Initial position for YN
double YN1 = 0.720;          // Initial position for YN1
double XCR1 = 0.496;         // value of xcr1 of the Logistic map 1
double XCR2 = 0.477;         // value of xcr2 of the Logistic map 2
int x;                       // produced bit from Logistic map 1
int y;                       // produced bit from Logistic map 2
double rem;                  // result of the modulo
int n = 1;
void setup() {
```

```
Serial.begin (9600);          // start serial communication at 9600bps
// De-skewing technique for each system
void loop() {
if ((XN < XCR1) && (XN1 >= XCR1))
x = 0;
if ((XN >= XCR1) && (XN1 < XCR1))
x = 1;
if ((YN < XCR2) && (YN1 >= XCR2))
y = 0;
if ((YN >= XCR2) && (YN1 < XCR2))
y = 1;
// XOR Function
rem = n%2;
if (rem == 1.00) {
if ((x == 0) && (y == 0))
Serial.println(0);              // write bit 0 to the serial interface
if ((x == 1) && (y == 1))
Serial.println(0);              // write bit 0 to the serial interface
if ((x == 0) && (y == 1))
Serial.println(1);              // write bit 1 to the serial interface
if ((x == 1) && (y == 0))
Serial.println(1);              // write bit 1 to the serial interface
}
// The Logistic Maps' functions
XN = XN1;
XN1 = R1 * XN * (1.000 - XN);
YN = YN1;
YN1 = R2 * YN * (1.000 - YN);
n = n + 1;}
```

Listing 1: The sketch for the proposed CRBG

As it is known from information theory, noise has maximum entropy. For this reason, the systems' parameters and initial condition are chosen so as measure-theoretic entropy ([42]) of the CRBG, which is given by the following equation, is maximum.

$$H_n = lim_{n\to\infty}(-\sum_{B^n} P(B^n)\ lnP(B^n)/n) \tag{7}$$

$P(B^n)$ is the probability of occurrence of a binary subsequence B of length n.

Using the procedure described in the previous section, by using $(r_1, r_2) = (3.990, 3.984)$, $(x_{cr1}, x_{cr2}) = (0.496, 0.477)$ and initial condition $(x_0, y_0) = (0.400, 0.500)$, a bitstream of 200,000 bits is obtained which is divided in 10 bit sequences of length 20,000 bits. In Table 2 the measure-theoretic entropy for $n = 3$ and $n = 4$ and the detailed results of the 10 bit sequences which were subjected to the four tests of FIPS-140-2 test suite, are presented. As a conclusion, all the bit sequences produced by the CRBG have numerically verified the specific characteristics expected of random bit sequences.

Table 2: Results of the measure-theoretic entropy and FIPS-140-2 test suite, for 10 bit sequences produced by the proposed CRBG

Tests	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
H_3	0.6928639	0.6927789	0.6930688	0.6927888	0.6928473	0.6928352	0.6928734	0.6930840	0.6930764	0.6928072
H_4	0.6925834	0.6928304	0.6926863	0.6925997	0.6922313	0.6925797	0.6927214	0.6925965	0.6926413	0.6925933
Monobit Test	√	√	√	√	√	√	√	√	√	√
Poker Test	√	√	√	√	√	√	√	√	√	√
Runs Test	√	√	√	√	√	√	√	√	√	√
Long Run Test	√	√	√	√	√	√	√	√	√	√
Result	**Passed**	**Passed**	**Passed**	**Passed**	**Passed**	**Passed**	**Passed**	**Passed**	**Passed**	**Passed**

Finally, by using the proposed CRBG, with the systems' parameters and initial conditions which has been previously mentioned, two bit sequences of length 20,000 bits from the outputs of each of the two Logistic maps, have been obtained. The measure-theoretic entropy and the results of FIPS-140-2 test suite of the two bit sequences produced by each Logistic map and of the XORed bitstream, which is the output of the proposed CRBG, are shown in Table 3. As follows from the comparison of the results of Table 3, the application of the XOR function increases significantly the measure-theoretic entropy and also improves the results of the FIPS-140-2 test suite. Although the bit sequence produced from the first Logistic map (LM1) failed to pass the Poker Test ($x_3 = 140.36064$) and also the bit sequence produced from the second Logistic map (LM2) failed to pass the Poker Test ($x_3 = 119.81938$) and the Runs Test, the final bit sequence of the CRBG passed all the tests of FIPS-140-2 with very good results, as it shown in Table 4.

Table 3: Results of the measure-theoretic entropy and FIPS-140-2 test suite, for the bit sequences of each Logistic map and of the final CRBG

Tests	**LM1**	**LM2**	**LM1 (XOR) LM2**
H_3	0.6893599	0.6901588	0.6930719
H_4	0.6874755	0.6925876	0.6926863
Monobit Test	√	√	√
Poker Test	X	X	√
Runs Test	√	X	√
Long Run Test	√	√	√
Result	**Failed**	**Failed**	**Passed**

Table 4: Results of FIPS-140-2 test, for the CRBG

Monobit Test	**Poker Test**	**Runs Test**	**Long Run Test**
n_1 = 10018 (50.09 %)	3.0021	B_1 = 2469 B_2 = 1268 B_3 = 617 B_4 = 323 B_5 = 161 B_6 = 158	No
Passed	**Passed**	**Passed**	**Passed**

5 Conclusions

In this paper, a chaotic random bit generator, which is based on the Logistic map and implemented by a microcontroller, was presented. The final bit sequence was the result of the application of the XOR function in two chaotic bit sequences that were produced by Logistic maps running side-by-side, with different initial conditions and systems' parameters. The use of the XOR function joint with different values of threshold values (x_{cr1}, x_{cr2}) of each map increased the complexity of the random bit sequence, in regard to other previous works, as it was confirmed by the use of a well-known statistical test suite FIPS-140-2. So, the chaotic random bit generator of this work proved to be very robust against interference from an intruder.

Also, the use of probably the most commercial platform, the Arduino prototyping platform, appointed the proposed chaotic random bit generator suitable in many interesting applications, such as in the design of robot's motion controllers and cryptographic systems. So, in future works, the experimental realization of such systems based in the proposed chaotic random bit generator will be studied.

References

[1] A.J. Menezes, P.C. Van Oorschot and S. A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.

[2] T. Shu, *Uniform random numbers: Theory and practice*, Kluwer Academic Publishers, 1995.

[3] D. Knuth, *The art of computer programming*, Addison-Wesley Publishing Co., Reading, MA, 1981.

[4] S. Park and K. Miller, Random numbers generators: Good ones are hard to find, *Communications of the ACM*, **31**(10), (1988), 1192-1201.

[5] C. Grebogi and J. Yorke, *The impact of chaos on science and society*, United Nations University Press, 1997.

[6] T. Yang, A survey of chaotic secure communication systems, *International Journal of Computational Cognition*, **2**(2), (2004), 81-130.

[7] L. Kocarev, Chaos-based cryptography: A brief overview, *IEEE Circuits and Systems Magazine*, **1**, (2001), 6-21.

[8] G. Alvarez and S. Li, Some basic cryptographic requirements for chaos based cryptosystems, *International Journal of Bifurcation and Chaos*, **16**, (2006), 2129-2151.

[9] J. Wei, X. Liao, K. Wong and T. Xiang, A new chaotic cryptosystem, *Chaos, Solitons & Fractals*, **30**(5), (2006), 1143-1152.

[10] K. Li, Y. C. Soh and Z. G. Li, Chaotic cryptosystem with high sensitivity to parameter mismatch, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **50**, (2003), 579-583.

[11] Ch. K. Volos, I.M. Kyprianidis and I.N. Stouboulos, Experimental demonstration of a chaotic cryptographic scheme, *WSEAS Transactions on Circuits and Systems*, **5**, (2006), 1654-1661.

[12] Ch. K. Volos, I.M. Kyprianidis and I. N. Stouboulos, Fingerprint images encryption process based on a chaotic true random bits generator,

International Journal of Multimedia Intelligence and Security, **1**, (2010), 320-335.

[13] Ch. K Volos, I. M. Kyprianidis and I.N. Stouboulos, Image encryption process based on chaotic synchronization phenomena, *Signal Processing*, **93**, (2013), 1328-1340.

[14] S. Oishi and H. Inoue, Pseudo-random number generators and chaos, *Transactions of the Institute of Electronics and Communication Engineers of Japan E*, **65**, (1982), 534-541.

[15] V.V. Kolesov, R.V. Belyaev and G.M. Voronov, Digital random-number generator based on the chaotic signal algorithm, *Journal of Communications Technology and Electronics*, **46**, (2001), 1258-1263.

[16] T. Stojanovski and L. Kocarev, Chaos-based random number generators - Part I: Analysis, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **48**, (2001), 281-288.

[17] S. Li, X. Mou and Y. Cai, Pseudo-random bit generator based on coupled chaotic systems and its application in stream-ciphers cryptography, In Progress in Cryptology – INDOCRYPT 2001, *Lecture Notes in Computer Science*, **2247**, (2001), 316-329.

[18] L. Kocarev and G. Jakimoski, Pseudorandom bits generated by chaotic maps, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **50**, (2003), 123-126.

[19] S. M. Fu, Z. Y. Chen and Y. A. Zhou, Chaos based random number generators, *Computer Research and Development*, **41**, (2004), 749-754.

[20] L. Huaping, S. Wang and H. Gang, Pseudo-random number generator based on coupled map lattices, *International Journal of Modern Physics B*, **18**, (2004), 2409-2414.

[21] R. M. May, Simple mathematical models with very complicated dynamics, *Nature*, **261**, (1976), 459-467.

[22] S.M. Ulam and J. Von Neumann, On combination of stochastic and deterministic processes, *Bulletin of the American Mathematical Socie*ty, **53**, (1947), 11-20.

[23] R. Ursulean, Reconsidering the generalized Logistic map as a pseudo random bit generator, *Electronika Ir Electrotechnika*, **7**, (2004), 10-13.

[24] V. Patidar, K.K. Sud and N.K. Pareek, A pseudo random bit generator based on chaotic Logistic map and its statistical testing, *Informatica*, **33**, (2009), 441-452.

[25] S. Ahadpour, Y. R. Sadra and Z. ArastehFard, A novel chaotic encryption scheme based on pseudorandom bit padding, *International Journal of Computer Science Issues*, **9**, (2012), 449-456.

[26] J. Liu, Design of a chaotic random sequence and its application, *Computer Engineering*, **31**, (2005), 150-152.

[27] L. Wang, F. P. Wang and Z. J. Wang, Novel chaos-based pseudo-random number generator, *Acta Physica Sinica*, **55**, (2006), 3964-3968.

[28] X. M. Li, H. B. Shen and X. L. Yan, Characteristic analysis of a chaotic random number generator using piece-wise-linear map, *Journal of Electronics Information Technology*, **27**, (2005), 874-878.

[29] M.J. Feigenbaum, The universal metric properties of nonlinear transformations, *Journal of Statistical Physics*, **21**, (1979), 669-706.

[30] M. J. Feigenbaum, Universal behaviour in nonlinear systems, *Los Alamos Science*, **1**, (1980), 4-27.

[31] J. Fridrich, Symmetric ciphers based on two-dimensional chaotic maps, *International Journal of Bifurcation and Chaos*, **8**, (1998), 1259-1264.

[32] Q. Zhou, K. W. Wong, X. Liao, T. Xiang and Y. Hu, Parallel image encryption algorithm based on discretized chaotic map, *Chaos Solitons & Fractals*, **38**, (2008), 1081-1092.

[33] R. Tenny and L. S. Tsimring, Additive mixing modulation for public key encryption based on distributed dynamics, *IEEE Transactions on Circuits and Systems I*, **52**, (2005), 672-679.

[34] G. Jakimoski and L. Kocarev, Block encryption ciphers based on chaotic maps, *IEEE Transactions on Circuits and Systems I*, **48**, (2002), 163-169.

[35] Y. Wang, X. Liao and K. Wong, One-way hash function construction based on 2D coupled map lattices, Information Sciences, **178**, (2008), 1391-1406.

[36] J. Von Neumann, Various techniques used in connection with random digits, G.E. Forsythe (eds.), *Applied Mathematics Series*, National Bureau of Standards, **12**, (1951), 36-38.

[37] Arduino, www.arduino.cc.

[38] NIST, Security Requirements for Cryptographic Modules, FIPS PUB 140-2, http://csrc.nist.gov/publications/fips/fips140-2/ fips1402.pdf, 2001.

[39] G. Marsaglia, DIEHARD Statistical Tests, http://stst.fsu.edu/pub/diehard, 1995.

[40] H. Gustafson, H.E. Dawson, L. Nielsen and W. Caelli, A computer package for measuring the strength of encryption algorithms, *Journal of Computer Security*, **13**, (1994), 687-697.

[41] D. Knuth, *The art of computer programming: Semiemperical algorithms*, Addison Wesley, Reading, USA, 1998.

[42] A.M. Fraser, Information and entropy in strange attractors, *IEEE Transactions on Information Theory*, **35**(2), (1989), 245-262.