Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Random number generator based on a memristive circuit

Juan Polo [a], Hans López [a], Cesar Hernández [b,*]

[a] Faculty of Engineering, Universidad Distrital Francisco José de Caldas, Bogotá D.C., Colombia
[b] Technological Faculty, Universidad Distrital Francisco José de Caldas, Bogotá D.C., Colombia

ARTICLE INFO

Keywords:
Attractor
Chaotic circuit
Long-range dependence
Fractal
Random number generator
Memristor

ABSTRACT

In this paper we discuss the details, limitations, and difficulties of the implementation in hardware of a memristor-based random number generator that exhibits monofractal/multifractal behavior. To do so, the components and selection criteria of a reference memristor and one proposed by the authors, the chaotic circuit leveraging them, and the processing that is performed on the chaotic signals to achieve the random discrete sequences are described. After applying the estimation tools, findings indicate that more than 60% of the proposed combinations allow generating random discrete sequences, with long-range dependence, and that both monofractal and multifractal behaviors can also be obtained. Consequently, a hardware system was achieved that can be used as a source of entropy in future synthetic biological signal generators.

1. Introduction

1.1. General context

Euclidean geometry fails to represent nature because geometric shapes like circles, spheres, and cones, among others, cannot represent elements of nature such as clouds, coasts, trees, and mountains. Moreover, it falls short in describing phenomena such as the path traveled by lightning, the structure of the circulatory and nervous systems, the shape of galaxy clusters, or the arrangement of the feathers of a bird in flight [1]. Instead, fractal geometry enables a new way of seeing nature, in which it seeks to discover an order between systems that seem random. Under this perspective, new representations arise with the purpose of modeling natural processes, looking for relationships between the random variables that compose them; these are usually represented from discrete sequences obtained from a Random Number Generator (RNG) [1,2].

RNGs are elements frequently employed in the field of cryptography to guarantee security in the information sent, as well as for the modeling of stochastic processes and mathematical simulations. Nonetheless, those used in the modeling of natural processes tend to have some type of dependency that differentiates them from those utilized in cryptography. Some examples of the processes that have been modeled are climate behavior [3], financial market behavior, and neural cells, among others [4,5].

Another behavior of interest that is observed in the processes of nature corresponds to the Long-Range Dependence (LRD), a property that has been registered in models that seek to represent the flow of traffic in cities, market behavior, electrocardiogram (ECG) signals, and so forth. Consequently, it is essential to model these random variables from RNGs that exhibit fractal and LRD behavior.

In [1] a random number generator model based on the memristor is proposed, for which monofractal and multifractal behavior is

* Corresponding author.
E-mail addresses: jcpolos@udistrital.edu.co (J. Polo), hilopezc@udistrital.edu.co (H. López), cahernandezs@udistrital.edu.co (C. Hernández).

https://doi.org/10.1016/j.heliyon.2024.e26635
Received 12 August 2023; Received in revised form 11 February 2024; Accepted 16 February 2024
Available online 17 February 2024

registered, together with LRD that can be used in the modeling of natural processes. However, this was performed only in simulation, and the problems, difficulties, constraints, and additional considerations that may appear in a hardware implementation are unknown. Guided by this, the question arises: how is the performance of a random number generator based on a hardware-implemented memristive circuit?

In accordance with the above, this work aims to evaluate the performance of a random number generator, based on a memristive circuit and implemented in hardware, for the synthesis of data with long-range dependence.

1.2. Contributions and scope

The carried-out work is the first hardware implementation of a random number generator with long-range dependency and mono/multifractal behavior. The contribution is significant insofar as it opens the tangible possibility of implementing biological signal generators that naturally present such behavior. For example, it can be used to adequately display the distribution of times between: R peaks in ECG signals; firing of neurons in an electroencephalogram (EEG) signal; or action potentials of motor neurons in an electromyography (EMG) signal. To date, it has been found in the specialized literature that the generation of random numbers has been carried out with statistical independence (for instance, with the inverse method), even when it has been evidenced that the long-range dependence occurs in the cases just mentioned and in other fields outside electronic engineering (as in the particular case of rainfall time series).

1.3. Literature review

Chaotic systems are characterized by a lack of periodicity and being unpredictable and sensitive to initial conditions, which result in a high degree of randomness compared to noise and jitter when used in RNGs [6]. Among the implementations made with chaotic systems is the use of the Jerk system, as reported in Ref. [7]. Diehard tests are used to evaluate the performance of the resulting generator. The output of the generator corresponds to values between 0 and 1. In Ref. [6] the use of the Jerk system in True Random Number Generators (TRNG) is also reported, which satisfactorily passes the National Institute of Standards and Technology (NIST) tests.

Another type of implementation corresponds to the use of a Field-Programmable Gate Array (FPGA) to represent the chaotic system, as is the case with [6]. This implementation presents an improvement by being more flexible in prototyping. A chaotic oscillator is presented from the RK-4 algorithm in Very High speed integrates circuit Hardware Description Language (VHDL).

There have also been some implementations of TRNG based on some well-known chaotic systems, such as the Chua circuit. In Ref. [8] cubic nonlinearity is used as a nonlinear function and the output of the chaotic circuit is converted into a number within the range of 0 and 15. The performance of the resulting generator is evaluated using the Monobit randomness tests and the Chi-square test.

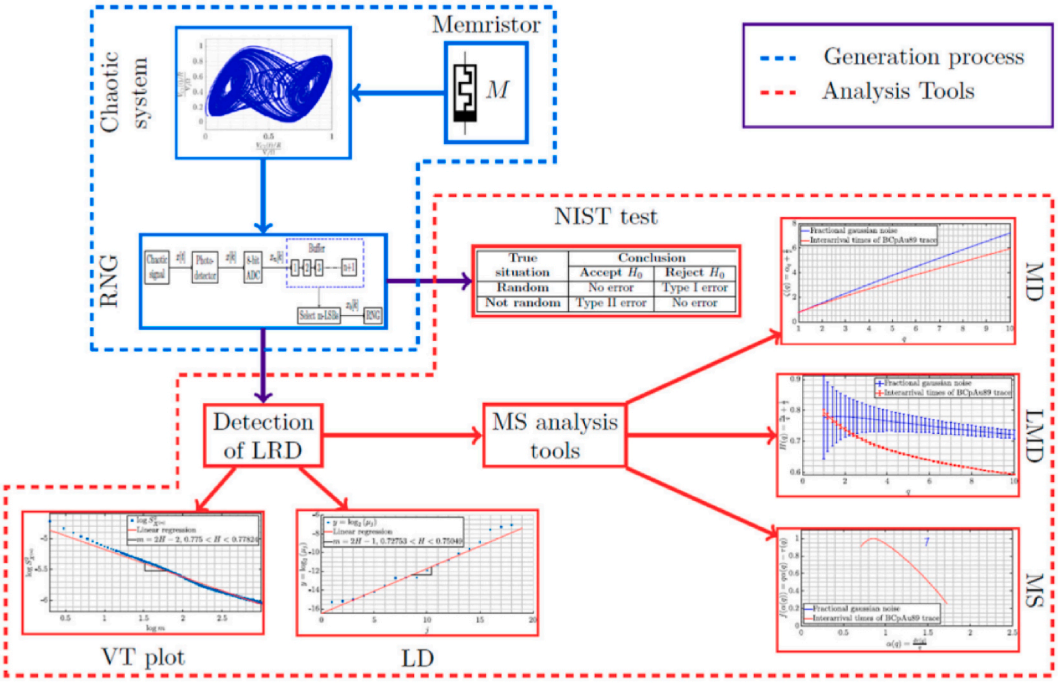

Fig. 1. Diagram of the developed process.
Source: Taken from Ref. [1].

In Ref. [6] an RNG based on the Chua circuit is reported, which uses a negative resistance and a CMOS circuit with 12 transistors along with an analog-to-digital converter (ADC) and linear feedback shift register (LFSR).

Another example of the use of chaotic systems is recorded in Ref. [1], in which the development of an RNG that makes use of chaotic systems as a source of entropy is carried out. The chaotic systems used are circuits based on the memristor. Fig. 1 summarizes the process carried out during the research. The part enclosed in blue corresponds to the model used to generate the random numbers, and the part enclosed in red corresponds to the tests conducted by the researchers to evaluate the model. The purpose of the authors was to observe the multifractal behavior and the LRD of the generator proposed. For this purpose, we applied the variance-time diagram and the Log-scale diagram to calculate the Hurst parameter, which us allows to determine if the sequence of symbols presents the LRD characteristic. For multifractal behavior, we use the multiscale diagram, the linear multiscale, and the multifractal spectrum. Finally, we evaluated the generator using conventional NIST tests.

In this vein, we detected, through simulation tools, the existence of the monofractal/multifractal behavior of an RNG based on memristor. Despite the novelty of the discovery, the logical step to follow requires the hardware implementation of the memristor and the chaotic circuit that uses it to investigate the details, limitations and difficulties, which are not contemplated in the advanced simulations. With this purpose in mind, a systematic review of five memristor emulator circuits was carried out by the authors, comparing them in terms of cost and operating frequency, and a memristor emulator circuit was proposed that uses a wave rectifier based on operational amplifiers, as a non-linear element. This memristor and a reference one was used in a hardware implementation of a Chua chaotic circuit. The modifications made, together with the measurements made in the laboratory, provide evidence that the chaotic system obtained satisfies the necessary conditions to be used in the truly random number generator, implemented in hardware, that exhibits monofractal or multifractal behavior.

1.4. *Organization of the document*

This work is organized and presented in five sections. Section 2 describes the fundamentals. Section 3 describes the proposed model. Section 4 presents the results obtained. Section 5 gives the general conclusions of the work.

2. Conceptual foundations

Below, the most relevant concepts for the development of this work are formally defined, such as the memristor, chaotic systems, and random number generation.

2.1. *Memristor*

In 1968, Fano proposed the existence of four basic elements through the theory of electromagnetism. At the time, only three of these elements were known (resistors, capacitors, and inductors); however, the last proposed element was still unknown [9]. Years later, in 1971, Chua proposed a two-terminal device called the memristor that allows to relate the variables described by Fano for the fourth element, flow and electric charge [10]. In this paper, two types of classifications are proposed according to the variable that controls the resistance, one controlled by charge, resulting in the behavior described by Equation (1), or controlled by flow, corresponding to Equation (2).

$$v(t) = M(q(t))i(t)\ M(q) = \frac{d\varphi(q)}{dq}\ p(t) = M(q(t))i^2(t) \tag{1}$$

$$i(t) = W(\varphi(t))v(t)\ W(\varphi) = \frac{dq(\varphi)}{d\varphi}\ p(t) = W(\varphi(t))v^2(t) \tag{2}$$

To determine if a device is considered a memristor, Chua proposes three characteristics that it must meet [10], these are:

1. The device must have a pinch on the hysteresis loop generated on the voltage-current plane for some period of an excitation signal.
2. The area of pinched hysteresis lobes should decrease monotonically with an increase in frequency.
3. The pinch hysteresis loop should be reduced to a simple function when the frequency tends to infinity.

In 2008, HP Labs reported the discovery of the memristor from a device composed of titanium dioxide (TiO2), and since then, numerous patents have been filed for memristor devices composed of graphene oxide (GO), silicon oxide (SiO2), polymers, among others [10]. As of now, the device is not commercially available, and several approaches have been developed to emulate its behavior using operational amplifiers (OP AMP), analog multipliers, positive second-generation current conveyors (CCII+), and other active components [5,11,12].

2.2. *Chaotic systems*

Chaotic systems consist of a set of differential or difference equations with seemingly unpredictable behavior, yet they do not usually have a very complex structure [13], which makes it possible to implement them in hardware with relative ease. One way to implement these types of systems corresponds to the implementation of their equations from OP AMP configured as integrators and amplifiers; an example of this corresponds to the Lorenz chaotic system, which has been implemented from multipliers and CCII + [14, 15], they have also been implemented from OP MPAs and multipliers [16,17].

Another way to implement chaotic systems is through second-order circuits, capacitors, and inductors. Some examples are the Chua circuit with memristor [18], the canonical Chua circuit [19], or the so-called simplest chaotic circuit [20,21].

2.3. RNG

Random Number Generators (RNGs) can be classified into two categories: pseudo-random number generators (PRNGs) and true random number generators (TRNGs). For PRNGs, a seed is typically utilized, which is the basis of the sequence of symbols generated from an algorithm; in TRNGs, a source of physical entropy is used as a basis [6].

The use of a physical entropy source allows for greater randomness and unpredictability in symbol sequences, which is why the use of TRNG is preferred in applications such as cryptography. Among the most commonly used sources is the noise coming from some electronic components such as resistors and capacitors [6], or active elements such as diodes [22], as well as phase jitter in oscillating signals and chaotic systems. This classification is summarized in Fig. 2.

The TRNGs can be divided into 3 modules, as shown in Fig. 3: an entropy source, a digitization stage and a post-processing stage, and the performance of tests to evaluate the performance of the generator.

Randomness in TRNGs with noise-based entropy sources from electronic elements is affected by the nonlinearity of amplifiers and variations in power supplies [24]. For jitter, a CMOS ring oscillator is commonly employed, but low randomness has been reported in these implementations [6].

3. Proposed model

The development of this work is based on the results achieved in a prior study that examined, analyzed, proposed, implemented, and evaluated the generation of an entropy source from the hardware implementation of a chaotic circuit based on a memristor. In this work, the generation of random numbers from the results achieved was proposed as future research. This project was carried out by the same authors of this paper. In this first work, the authors determined the reference memristor, the proposed memristor and the chaotic circuit that generates the entropy source, which are presented below.

The reference memristor is given by the proposal of [25] in which a variable memductance is proposed from analog multipliers and a quadratic function. This circuit is illustrated in Fig. 4.

The authors of [25] use an input signal whose amplitude ranges from 1 V to 4 V and whose frequency ranges from 100 Hz to 2.2 kHz. The expression corresponding to the memductance produced is shown in Equation (3).

$$W(\varphi)=\frac{1}{R_\alpha}-\frac{1}{R^2C^2}\frac{R_{\beta 1}+R_{\beta 2}}{100R_\alpha R_{\beta 1}}\varphi^2(t) \tag{3}$$

From the proposal of [25] it can be seen that the nonlinear function used by the authors corresponds to a quadratic function from analog multipliers, so to propose an emulator it is necessary to change the block corresponding to the nonlinear function.

As a proposed memristor, it was founded upon the proposal of [25], and full wave rectification was applied as a nonlinear function since it was not used in the references consulted, and this can be implemented with general purpose OP AMP and low-cost fast switching diodes, resulting in the circuit of Fig. 5.

The expression of the memductance produced from the relationship between the voltage and the current of the emulator is presented in Equation (4), and the expression of the emulator memductance can be observed in Equation (5).

$$i_m=v_m\left(\frac{1}{R_\alpha}-\frac{R_{\beta 1}+R_{\beta 2}}{10R_\alpha R_{\beta 1}RC}|\varphi(t)|\right) \tag{4}$$

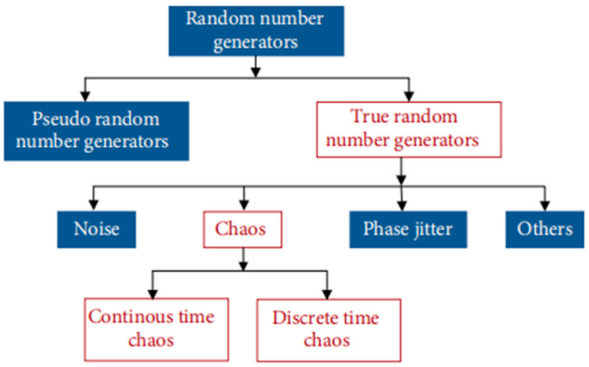

Fig. 2. RNG classification.
Source: Taken from Ref. [6].

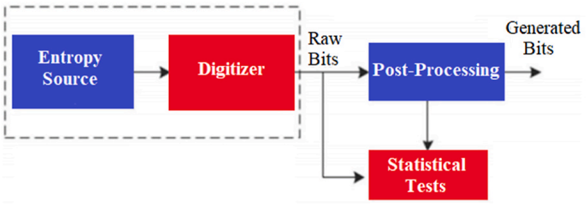

Fig. 3. Block diagram of a TRNG.
Source: Taken from Ref. [23].

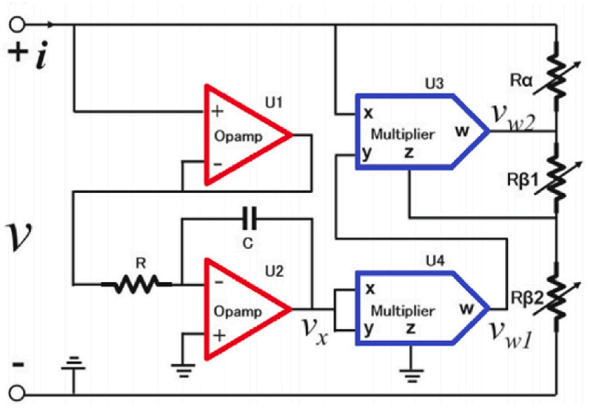

Fig. 4. Reference memristor.
Source: Taken from Ref. [25].

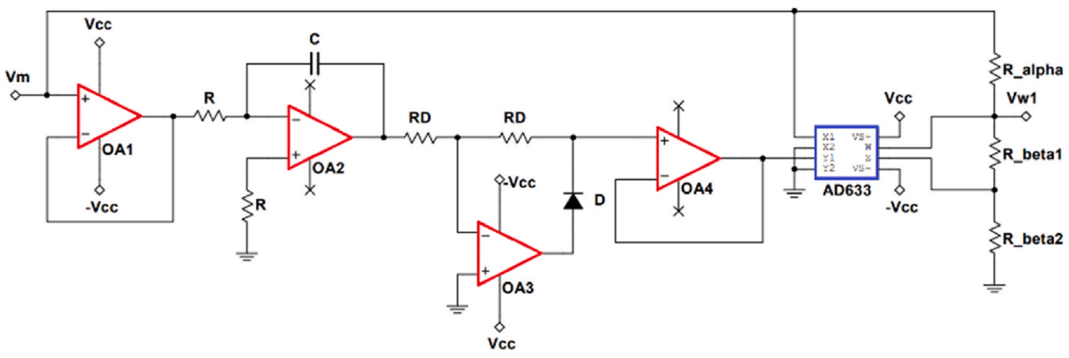

Fig. 5. Proposed memristor (with full wave rectification as a nonlinear function).
Source: The author.

$$W(\varphi) = \frac{1}{R_\alpha} - \frac{R_{\beta1} + R_{\beta2}}{10 R_\alpha R_{\beta1} RC} |\varphi(t)| \tag{5}$$

Now the conducted implementation of the memristor is used to perform a chaotic circuit and generate a source of entropy. To achieve this, an exploration of the five circuits documented in Ref. [1] that have been used for this task was carried out with the purpose of selecting a circuit that can be implemented. The five proposals are evaluated using as evaluation criteria the type of memristor employed, the characteristics of the elements used and the number of elements.

According to the evaluation, circuit 5 is selected as the chaotic circuit to be used as a source of entropy as it proved to be the most feasible option for implementation based on the aforementioned criteria. The necessary conditioning is applied to the proposed memristor, resulting in the circuit depicted in Fig. 6. Equation (6) presents the system of equations that describes the behavior of the chaotic circuit.

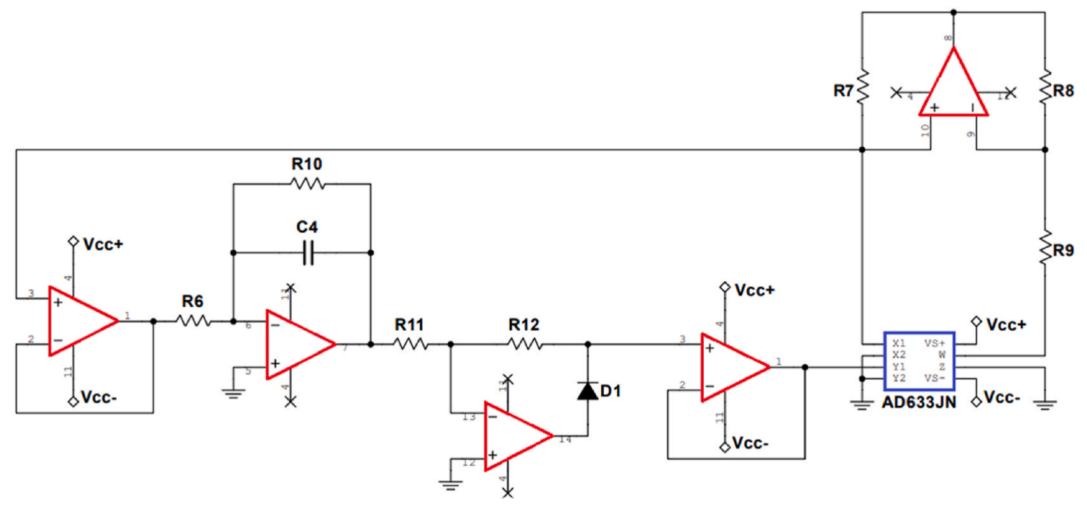

Fig. 6. Chaotic circuit with a modified proposed memristor.
Source: The author

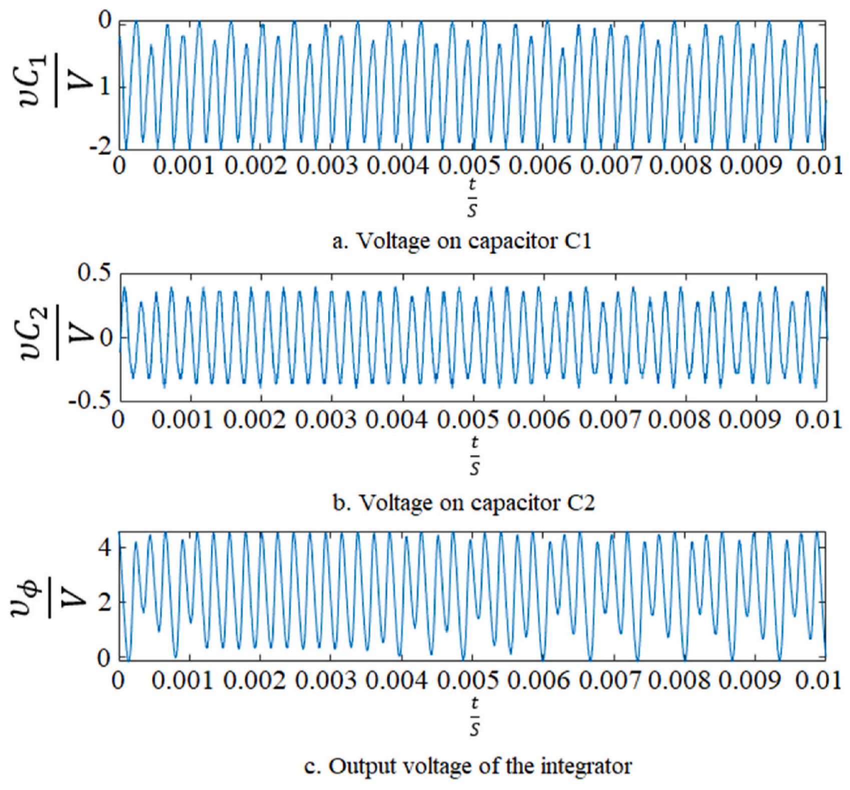

Fig. 7. Signals obtained in the implementation of chaotic circuit 5 with the modified proposed memristor: a. Voltage on capacitor C1. b. Voltage on capacitor C2. and c. Output voltage of the integrator.
Source: The author

$$\begin{aligned}\frac{dv_1}{dt} &= \frac{v_2 - v_1}{RC_1} + \frac{\left(G_a - G_b v_0^2\right) v_1}{C_1} \\ \frac{dv_2}{dt} &= \frac{v_1 - v_2}{RC_2} - \frac{i_3}{C_2} \\ \frac{di_3}{dt} &= \frac{v_2}{L} \\ \frac{dv_0}{dt} &= -\frac{v_1}{R_1 C_0} - \frac{v_0}{R_2 C_0}\end{aligned} \tag{6}$$

Finally, it is observed that the implemented circuit allows for to generation of signals and chaotic attractors resembling those obtained in the simulation, so that the implementation is deemed successful and can be used as a source of entropy. Figs. 7 and 8 show the system signals and attractors obtained in the implementation.

3.1. Implementation of the RNG model

With an entropy source defined, the process for implementing a hardware-based RNG is reduced to employing a suitable algorithm for generating random numbers. This project uses the algorithm proposed by the authors of [1] for the generation of random numbers, since it is desired to determine the conditions that must be taken into account to conduct the implementation and the differences between the results obtained between the simulation and those obtained in an implementation with respect to LRD and multifractal behavior.

3.2. Analysis of the random number generation model

According to Ref. [1], the study demonstrates that the random number generation algorithm consists of two stages. In the first stage, each signal from the chaotic circuit is processed to transition from a continuous-time to a discrete-time representation. The second stage involves the implementation of the algorithm to generate random numbers based on the previously processed signals from the chaotic circuit. Fig. 9 illustrates the process applied to one of the chaotic signals within the system.

In this stage, the signal is sampled from an ADC module of n-bits at a constant sampling rate of Ts. With the sampled signal, the selection of the least significant m bits (LSB) is made; this is because the greatest dependence occurs in these, as mentioned by the authors of [26]. The diagram in Fig. 9 illustrates the utilization of a subsampler, whose purpose is to reduce the sampling rate of the simulation carried out by the authors of [1] due to its high rate; this scenario does not occur in a real implementation since the sampling rate is limited by the processing tool, which is usually low due to the behavior of passive elements such as the capacitors of the ADC modules.

Fig. 10 shows the stage of generating random numbers from the processed chaotic circuit signals. Initially, a sequence is generated from the m LSBs of three of the signals of the chaotic circuit, and this is used to determine the output of the b-bit digital-to-analog converter (DAC) module, whose normalized output generates a decimal number between 0 and 1.

3.3. Characteristics of the elements of the processing tool

According to the analysis of the model for the generation of random numbers, it is observed that the resolution of the ADC and DAC

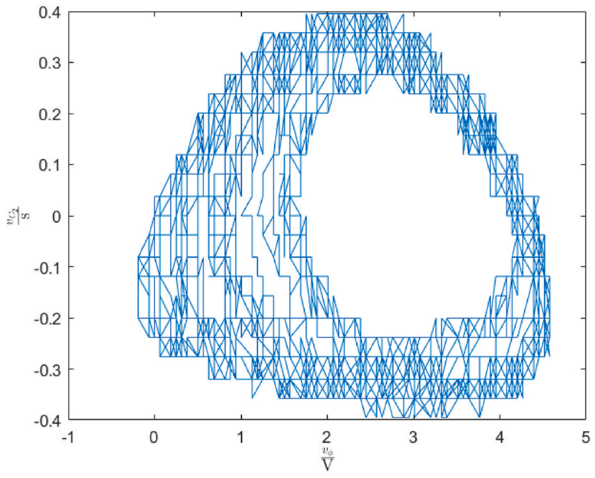

Fig. 8. Attractors obtained in the implementation of the chaotic circuit 5 with the modified proposed memristor.
Source: The author

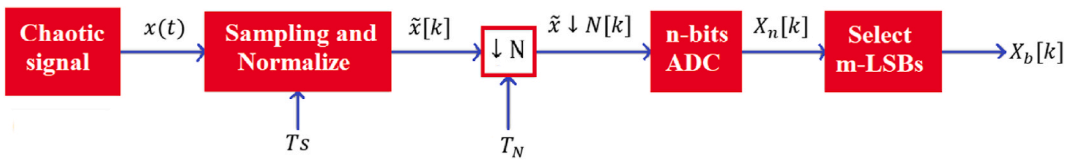

Fig. 9. Chaotic signal processing stage.
Source: Taken from Ref. [1].

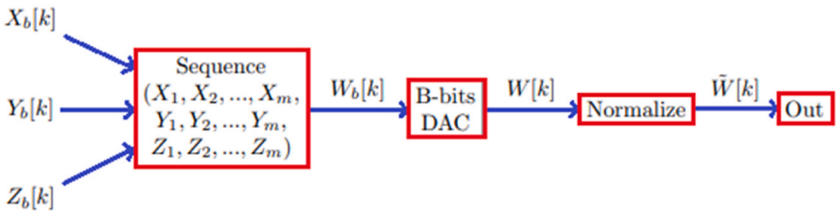

Fig. 10. Random number generation stage.
Source: Taken from Ref. [1].

modules must be configurable to be used in the implementation of the model [1]. In the case of the ADC module, it is noticeable that the signals used are those corresponding to the voltage since they are easier to sample and have the necessary number of signals.

The sampling of chaotic circuit signals can be carried out in two ways. The first style is carried out using a single ADC module, and the second corresponds to the use of three independent ADC modules. In the case of the first way, it is known that it is required to use an analog multiplexer that is responsible for changing the signal to be sampled; this means that only one signal can be sampled at a time, and the conversion time of the ADC module affects the sampling time. Another feature of this type of implementation is the need to use a dead time between conversions since interference problems can occur when dealing with the capacitors in charge of the conversion.

For the second form, it is observed that the limitations mentioned above do not occur in this type of sampling since there is only one conversion time in each channel. This time is usually less than the sampling time and only occurs once in the channel for each sample. The dead time between conversions ends up being the remaining time of the sampling.

Based on the above mentioned, it can be observed that it is more convenient to use three ADC modules, so the tool used must have at least three ADC modules with configurable resolution. The resolution values for the ADC are 8 bits, 10 bits, and 12 bits as recorded in Ref. [1]. In the case of the DAC module, only the use of a single module is required since it has a single output; this module must have the ability to set its resolution to the values of 0 bits, 8 bits, 10 bits, 12 bits, and 16 bits as recorded in Ref. [1].

3.4. *Selection of the processing tool*

In the market, there are multiple options from various manufacturers that can comply with the characteristics of the ADC. For this reason, the processing tool is selected according to familiarity with the manufacturer's microcontrollers and their availability in the domestic market to avoid high implementation costs and reduce the time needed to implement the algorithm.

Searching the STM32 catalog and verifying that there is availability in the national market, it is perceived that there are some proposals that meet the conditions of the ADC in the STM32F4 family of microcontrollers; however, none of them meets the conditions of the DAC.

Exploring other microcontrollers, it would not be possible to find any that could meet the characteristics required for the DAC. Thus, the function of the DAC is analyzed in order to find solutions that do not require the use of this module.

From the DAC, it is observed that it performs two tasks. The first task corresponds to the generation of the random number, and the second one is the generation of the output of the system. The random number can be obtained from its calculation by means of the expression of the DAC, which has a configurable resolution.

With respect to the output, it is possible to perform it without the use of the DAC module by using a serial protocol that allows sending the random number generated to a receiving terminal.

In the STM32F4 family of microcontrollers, it is noticeable that among the serial communication protocols that can be used, the fastest is the USB communication protocol. By means of the aforementioned, the STM32 nucleo-F446RE development board is selected since it meets the desired conditions for sampling, works with the USB protocol, and is in the domestic market. It is decided to use a development board instead of the microcontroller to be able to use it without having to worry about adding external components for microcontroller power, USB communication, and the use of ADC modules.

3.5. *Implementation of the algorithm*

Next, the sampling time, signal conditioning, sequence generation, random number generation, and random number generator output are defined.

3.5.1. Sampling time

Before processing chaotic signals, it is necessary to define the value of the sampling rate. For this procedure, the MATLAB system is used for the purpose of determining the most important frequency components of the signals of the chaotic circuit with reference memristor and the proposed one. Fig. 11 exhibits the frequency spectrum for the chaotic circuit signals with a reference memristor and Fig. 12 shows the frequency spectrum of the signals of the proposed memristor circuit.

In Figs. 11 and 12 it can be seen that the maximum frequency of the components of interest among all the signals is below 5 kHz, which makes it necessary to sample these signals with at least twice this frequency. Since the ability to use a higher frequency is convenient, the signals are sampled at a frequency of 100 kHz.

3.5.2. Signal conditioning

With the sample rate defined, it is now required to capture the signal by means of the ADC module. This element has a range of operations that is given by the manufacturer. In the case of the chosen microcontroller, it has to be from 0 V to 3.3 V.

Since the signals of the chaotic circuit are outside the operating range, it is necessary to perform a signal conditioning stage, which is responsible for adjusting the signal to take advantage of the operating range of the ADC module.

3.5.3. Sequence generation

The sequence is generated by concatenating the m LSBs of each of the sampled signals. Therefore, the size of the sequence is 3 m bits; if a resolution in the DAC is smaller than this size, a number that operates outside the DAC's account range would be acquired. For this reason, it may be necessary to adjust the sequence so that the sequence value is within the DAC's account range.

The process performed for this adjustment is to evaluate whether the value of the sequence is greater than the maximum DAC count. If it is larger, the sequence is modified by applying the operation of the module to the previous value of the sequence, using as a divisor the number of accounts of the DAC. If the condition is not met, the sequence value is not changed.

In [1] an order is not defined to generate the sequence, so it is of interest to identify if there is any effect on the way in which the sequence is generated when evaluating the output of the generator.

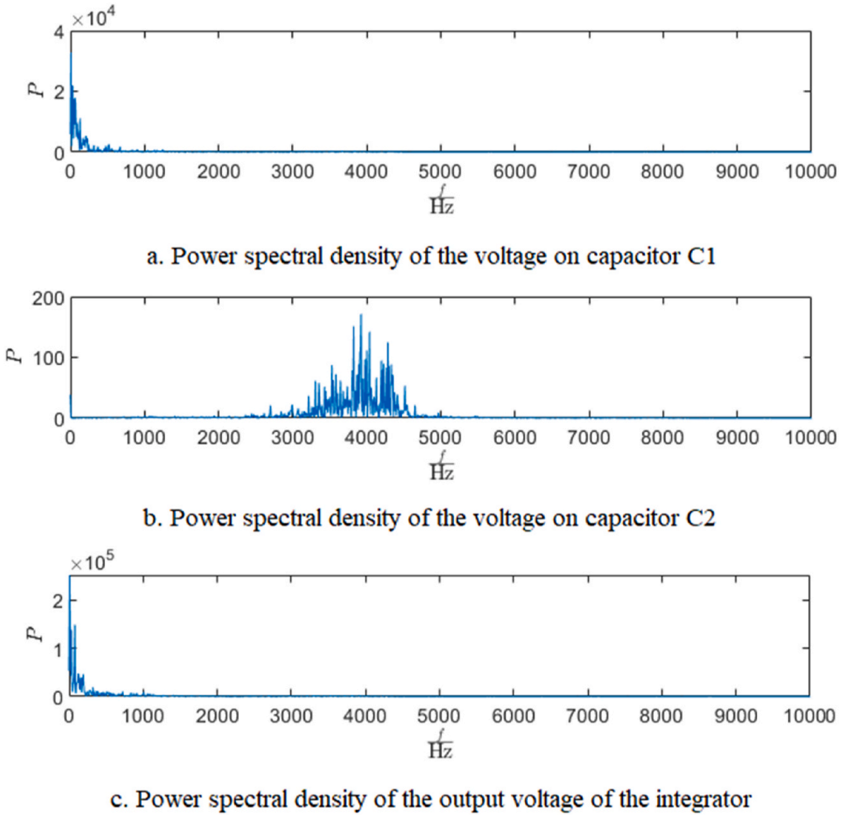

Fig. 11. Frequency spectrum of circuit signals with the reference memristor: a. Power spectral density of the voltage on capacitor C1. b. Power spectral density of the voltage on capacitor C2. c. Power spectral density of the voltage of the integrator.
Source: The author.

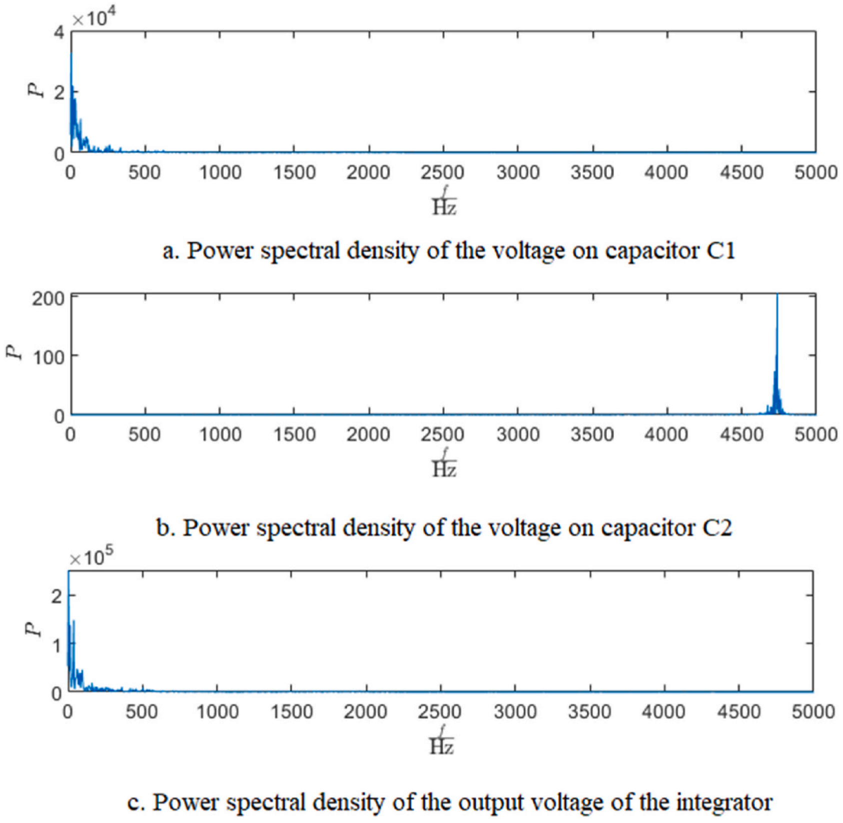

Fig. 12. Frequency spectrum of circuit signals with the proposed memristor: a. Power spectral density of the voltage on capacitor C1. b. Power spectral density of the voltage on capacitor C2. c. Power spectral density of the voltage of the integrator.
Source: The author.

3.5.4. Generation of random numbers

Once the appropriate input parameter of the DAC is obtained, the output is generated from the expression of the DAC, which is presented in Equation (7), using the desired resolution value.

$$v_{DAC} = \frac{v_{REF}}{2^b} \tag{7}$$

3.5.5. Random number generator output

To send the generated number, it must be taken into account that in order to send the decimal value obtained, it is necessary to convert this number to characters and then send it. The conversion to characters is a slow process that manages to negatively affect the sampling time; for this reason, the number obtained in this way is not sent.

It is chosen to send integer values since they are encoded in 8-bit packets and require less time to be sent. Since the number is decimal, a multiplier of 1000 is used to send the generated number to three decimal places; this number is stored in a 16-bit variable.

Once the integer is acquired, it is separated into groups of 8 bits to obtain a group with the least significant bits and another with the most significant bits. This process is repeated for 100 numbers, and this group of numbers is sent.

The receiving terminator has the task of joining the separated bits and obtaining the generated number again.

4. Results

4.1. LRD and fractal behavior testing

With the model implemented and using each of the entropy sources implemented, discrete sequences of size 10ˆ6 are generated for the combinations of n bits (ADC), m bits (LSB) and b bits (DAC) registered in Ref. [1]. LRD and fractal behavior are evaluated for the sequences obtained by the 10 combinations used.

4.2. LRD dependency tests

The detection of LRD is performed using the Hurst parameter (H), which can be determined from the variance-time diagram (VT) and the log-scale diagram (LD) [1]. The use of the variance-time diagram is selected as a tool to estimate the value of the Hurst parameter for each discrete sequence obtained.

Figs. 13 and 14 display an example of the resulting VT diagram using discrete sequences with and without LRD respectively.

Table 1 records the combinations and the Hurst parameter obtained by the authors of [1]. These results are compared with those obtained in the implementation when using both emulators of the memristor.

4.2.1. Reference memristor

Using the chaotic circuit with the reference memristor as an entropy source, multiple discrete sequences are generated for the combinations recorded in Table 1. For each combination, the sequences described in Table 2 are used in the algorithm.

Table 3 records the estimated values of the Hurst parameter from the time variance-diagram, for each combination and sequence used.

It can be identified that the Hurst parameter is indeed affected by the order used to generate the sequence. It can also be seen that if the combinations 7, 8, and 10, are excluded, the rest of the combinations of sequences of numbers that exhibit behavior with LRD are obtained.

On the other hand, the sixth combination ($b = 8$, $n = 12$ and $m = 10$) can be identified as the most interesting option; since, by simply changing the order of the concatenated signals, it is possible to select some significant values of the Hurst parameter, which belong to the interval (0.5,1). If another H is required, the other options summarized in Table 3 could be considered.

4.2.2. Proposed memristor

The same procedure performed previously is performed using the proposed memristor as a source of entropy in the chaotic circuit. Table 4 indicates the values obtained from the estimation of the Hurst parameter for the discrete sequences generated for the combinations described in Table 1 and the sequence orders in Table 2.

From Table 4 and it can be observed that the order in which the sequence is generated still affects the Hurst parameter. In contrast to the reference memristor, the number of combinations that present LRD ($0.5 < H < 1$), is scarcer.

Despite this, with combinations 2 and 3 various values of H that present LRD can be covered.

4.3. Fractal behavior

To evaluate the fractal behavior of number sequences, the multifractal diagram with multiscale (MD), the multiscale linear diagram (MLD), and the multifractal spectrum (MS) can be used [1]. The application of the multifractal spectrum is chosen since it allows for visually determining the fractal behavior, and with the help of the MATLAB system, it can be easily determined since it has been implemented in the toolboxes available as a multifractal analysis tool. Figs. 15 and 16 show an example of the resulting multifractal spectrum by using discrete sequences with monofractal behavior and multifractal behavior, respectively.

Table 5 records the results obtained by the authors of [1] for different combinations concerning fractal behavior. The symbol ($\cap$) refers to a multifractal behavior and (•) to a monofractal behavior.

4.3.1. Reference memristor

For the entropy source composed of the reference memristor, its fractal behavior is determined by the multifractal spectrum. The same symbols are used to identify multifractal behavior and monofractal behavior. This result can be seen in Table 6.

Table 6 shows that, when considering those combinations that present LRD, sequences with both monofractal and multifractal behavior are attained from the chaotic circuit with the reference memristor.

Additionally, it is possible to configure the same Hurst parameter but with different fractal behavior: as in the case of sequences 2 and 3, of combination 6. This attests to the versatility of the proposed generator.

4.3.2. Proposed memristor

The same procedure is performed with the discrete sequences obtained when using the chaotic circuit with the proposed memristor. The results obtained with respect to fractal behavior can be seen in Table 7.

It is possible to notice that, when considering those combinations that present LRD, the behavior is similar despite the fact that the memristor is different. Thus, it is possible to obtain discrete sequences from the proposed memristor with both monofractal and multifractal behavior using the chaotic circuit 5.

In this way, the hardware implementation of a random number generator with long range dependence is achieved from different chaotic input signals, being possible the setting of H and the fractal behavior, through the variation of the RNG parameters (such as the number of bits in the ADCs and DACs, the number of least significant bits taken from the ADC, and the order of the concatenated sequences). In fact, the combinations proposed (Tables 4 and 5) mostly present long-range dependence, with the possibility of selecting monofractal/multifractal behaviors (Tables 6 and 7). This last characteristic is differential, if compared to algorithms based on multiplicative cascades, such as MWM or MFHSW, which only allow generating discrete multifractal sequences of length 2ˆn. Furthermore, this generator has the advantage of generating the numbers sequentially up to the desired length k; in contrast, in multiplicative cascades we must wait to obtain a trace of length 2ˆn and discard the remaining 2ˆn – k synthetized values. These

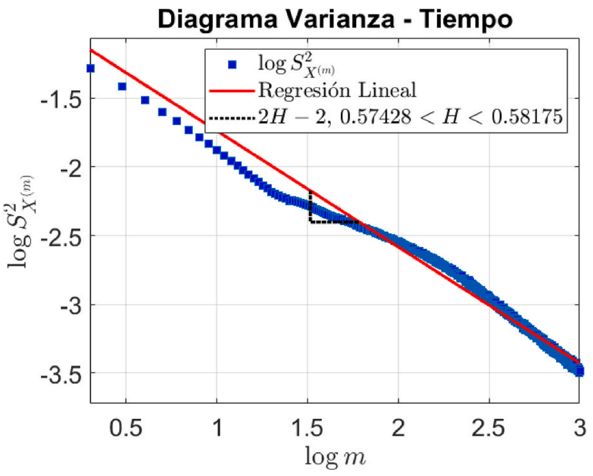

Fig. 13. VT diagram for a discrete sequence with LRD (H = 0.5780).
Source: The author.

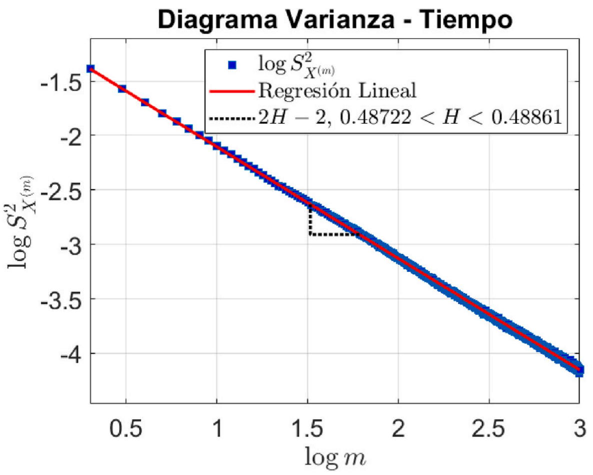

Fig. 14. VT diagram for a discrete sequence without LRD (H = 0.4879).
Source: The author.

Table 1
Combinations and Hurst parameter obtained by the authors of [1].

	Combination									
	1	2	3	4	5	6	7	8	9	10
b	8	16	12	16	10	8	0	8	0	0
n	8	8	8	10	10	12	12	10	10	10
m	4	7	7	9	8	10	9	10	10	9
H	0.5218	0.5756	0.6510	0.6757	0.6913	0.7196	0.7738	0.7958	0.8171	0.8253

characteristics suggest that the generator can be used as a reference framework for possible applications in the financial, geophysical, texture modeling, climate prediction, characterization of social behavior and growth of organisms, among others. Particularly, we are interested in using it, in a first application that allows establishing the time between R peaks of a synthetic ECG signal generator.

5. Conclusions

This project aims to observe the problems, conditions, and limitations that arise when implementing the model for the generation of random numbers with LRD and fractal behavior, starting from the entropy source to the output of the generator.

Table 2
Sequences generated by varying the order of signals.

Identificator	Order concatenated signals		
1	v_{C_1}	v_{C_2}	v_{φ}
2	v_{C_1}	v_{φ}	v_{C_2}
3	v_{C_2}	v_{C_1}	v_{φ}
4	v_{C_2}	v_{φ}	v_{C_1}
5	v_{φ}	v_{C_1}	v_{C_2}
6	v_{φ}	v_{C_2}	v_{C_1}

Table 3
Hurst parameter obtained with a chaotic circuit with the reference memristor for different combinations and sequences.

	Combination									
	1	2	3	4	5	6	7	8	9	10
1	0.7049	0.5654	0.5662	0.5997	0.6020	0.8312	0.4341	0.40	0.7049	0.3896
2	0.6023	0.5489	0.6030	0.7090	0.7514	0.7495	0.4347	0.4795	0.6023	0.4697
3	0.6023	0.5489	0.6030	0.7090	0.7514	0.7495	0.3802	0.4182	0.6023	0.4691
4	0.7013	0.5651	0.5745	0.7081	0.7481	0.6129	0.4344	0.4723	0.7013	0.4678
5	0.6023	0.5489	0.6989	0.7146	0.7121	0.6604	0.3845	0.4182	0.6023	0.4717
6	0.7013	0.5651	0.5695	0.5989	0.5990	0.5152	0.4344	0.4012	0.7013	0.3893

Table 4
Hurst parameter obtained with a chaotic circuit with the memristor proposed for different combinations and sequences.

	Combination									
	1	2	3	4	5	6	7	8	9	10
1	0.4879	0.5198	0.5228	0.4973	0.527	0.4991	0.4473	0.4889	0.4762	0.4625
2	0.578	0.7697	0.8019	0.7765	0.5053	0.4871	0.4439	0.49	0.4791	0.4636
3	0.4945	0.6252	0.6483	0.6478	0.5549	0.5311	0.3347	0.4896	0.4765	0.4633
4	0.833	0.7593	0.7972	0.7719	0.4884	0.4715	0.462	0.4913	0.4787	0.4645
5	0.4985	0.6101	0.6532	0.6568	0.501	0.4784	0.4083	0.4927	0.4816	0.4658
6	0.494	0.5145	0.5195	0.5009	0.4917	0.4925	0.4877	0.4931	0.481	0.4658

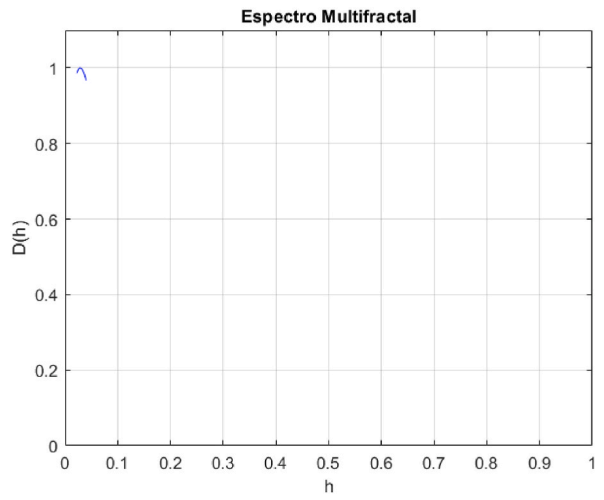

Fig. 15. Multifractal spectrum for sequences with monofractal behavior.
Source: The author.

When performing the numerical simulations, it is observed that it is necessary to use the exact values with respect to capacitors, inductors, and resistance of the chaotic circuit 5 to obtain the chaotic signals and attractors. This is reflected in the implementation because there is a need to use elements with the lowest possible tolerance to obtain results similar to the simulation, which is why the use of the inductance emulator increases the accuracy of the circuit if elements with low tolerance are used.

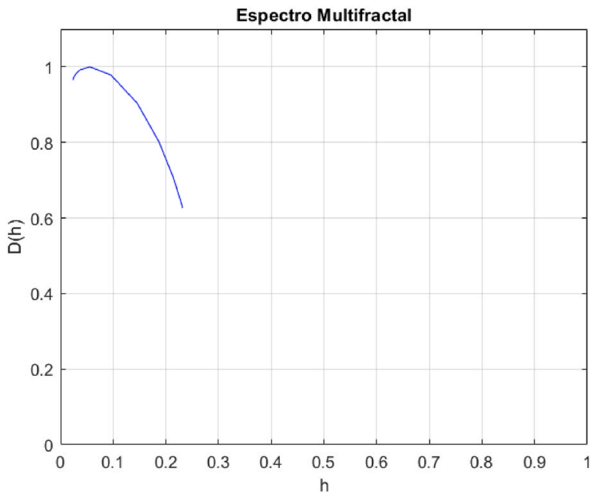

Fig. 16. Multifractal spectrum for sequences with multifractal behavior.
Source: The author.

Table 5
Results regarding fractal behavior obtained by the authors of [1].

Combinations			Results
b	n	m	
8	8	4	∩
16	8	3	•
16	10	9	∩
12	8	7	•
10	10	8	•
8	12	10	∩
0	12	9	•
8	10	10	∩
0	10	8	•
0	10	9	•

Table 6
Results of fractal behavior obtained for sequences generated with a chaotic circuit with the reference memristor.

Sequence order	Combination									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	•	•	•	•	•	∩	•	∩	•	•
2	∩	•	∩	∩	•	•	•	∩	•	•
3	∩	•	∩	∩	•	∩	∩	∩	∩	•
4	∩	•	∩	∩	•	∩	•	∩	•	•
5	∩	•	∩	∩	•	•	•	∩	•	•
6	•	•	•	•	•	∩	•	∩	•	•

Table 7
Results of fractal behavior obtained for sequences generated with a chaotic circuit with the proposed memristor.

Sequence order	Combination									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	•	•	•	•	•	∩	•	•	•	•
2	•	•	∩	∩	•	•	•	∩	•	•
3	∩	•	∩	∩	•	∩ s	•	∩	•	•
4	∩	•	•	•	•	∩	•	•	•	•
5	∩	•	∩	∩	•	•	•	∩	•	•
6	•	•	•	•	•	∩	•	•	•	•

For the signals generated by the chaotic circuit, it is evident that there is a relationship between the v_(C_1) and v_φ signals because, after being processed by the microcontroller, they behave as a negated version of each other within the operating range of the ADC. This makes it possible to sample two of the signals and obtain the third by means of its calculation, giving the opportunity to sample at a higher frequency.

More than 60% of the discrete sequences generated by the hardware implementation contain an LRD and monofractal and multifractal behavior from the variation of the parameters used by the authors of [1], despite the limitations and considerations that this whole process entails.

When comparing the results obtained between the hardware implementation and those proposed by the authors of [1], it is evident that there is no single sequence order that allows similar results to be obtained. However, by including the order of the sequence as a fourth parameter to vary in the model, it is possible to obtain close values when using the reference memristor.

With this project, it is feasible to observe that, despite the considerations and limitations involved in the implementation of the model for the generation of random numbers with LRD and fractal behavior, it is possible to obtain discrete sequences with LRD and fractal behavior by varying the parameters of the model.

Future work

By implementing the hardware, the need for calculation is eliminated and reduced to the handling of serial communication between the processing tool and the computer. This concedes to obtaining sequences with a user-defined size or limited by computer memory. Thus, exceeding the limits of the simulation. Being able to obtain discrete sequences with a user-defined size enables its use for the representation of variables that present LRD.

In the case of biological signals, the behavior of LRD in ECG, EMG, and EEG signals has been recorded. By using discrete sequences to represent the behavior of LRD in these signals, it is viable to obtain a synthetic signal generator that more accurately represents a real signal.

Other fields, such as finance, climate prediction, and social behavior, have registered the presence of LRD. The use of discrete sequences with LRD in predictive models can generate results that are closer to reality.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data availability statement

Data will be made available on request.

Additional information

No additional information is available for this paper.

CRediT authorship contribution statement

Juan Polo: Writing – review & editing, Validation, Software, Methodology, Formal analysis. **Hans López:** Writing – review & editing, Validation, Formal analysis, Data curation, Conceptualization. **Cesar Hernández:** Writing – review & editing, Writing – original draft, Project administration, Methodology, Formal analysis.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:Cesar Hernandez reports administrative support and equipment, drugs, or supplies were provided by District University Francisco José de Caldas.

Acknowledgments

The authors would like to thank the Center for Research and Scientific Development of Universidad Distrital Francisco Jose de Caldas for supporting in this investigative project.

References

[1] M. Téllez, J. Mejía, H. López, C. Hernández, Random number generator with long range dependence and multifractal behavior based on memristor, Electronics 9 (10) (2020), https://doi.org/10.3390/electronics9101607.

[2] L.M. Tuberquia-David, H.I. López-Chávez, C. Hernández, *A Multifractal Model for Cognitive Radio Networks,"* Primera Edición. Bogotá, Editorial UD, Colombia, 2019.

[3] A. Ramanathan, A.N.V. Satyanarayana, M. Mandal, Theoretical Predictability limits of spatially anisotropic multifractal processes: implications for weather prediction, Earth Space Sci. 6 (7) (2019) 1067–1080, https://doi.org/10.1029/2018EA000528.
[4] M. Tuberquia-David, F. Vela-Vargas, H. López-Chávez, C. Hernández, A multifractal wavelet model for the generation of long-range dependency traffic traces with adjustable parameters, Expert Syst. Appl. 62 (2016) 373–384, https://doi.org/10.1016/j.eswa.2016.05.010.
[5] L. Chua, G.C. Sirakoulis, A. Adamatzky, Handbook of memristor networks, Handbook of Memristor Networks (2019) 1–1368, https://doi.org/10.1007/978-3-319-76375-0.
[6] F. Yu, L. Li, Q. Tang, S. Cai, Y. Song, Q. Xu, A survey on true random number generators based on chaos, Discrete Dynam Nat. Soc. 2019 (2019), https://doi.org/10.1155/2019/2545123.
[7] R.C. Harrison, B.K. Rhea, A.N. Ramsey, R.N. Dean, J.E. Perkins, A true random number generator based on a chaotic Jerk system, Conference Proceedings - IEEE southeastcon (2019), https://doi.org/10.1109/SoutheastCon42311.2019.9020442. *April*-December.
[8] F. Ozkaynak, A novel random number generator based on fractional order chaotic Chua system, Elektronika Ir Elektrotechnika 26 (1) (2020) 52–57, https://doi.org/10.5755/j01.eie.26.1.24221.
[9] M. Hemmati, V. Rashtchi, A. Maleki, S. Toofan, A Configurable Memristor-Based Finite Impulse Response Filter, April 2019.
[10] A.G. Radwan, M.E. Fouda, On the mathematical modeling of memristor, memcapacitor, and meminductor, in: Proceedings of the International Conference on Microelectronics, ICM, 2015.
[11] A.G. Alharbi, M.H. Chowdhury, Memristor Emulator Circuits, Memristor Emulator Circuits, 2021, https://doi.org/10.1007/978-3-030-51882-0.
[12] Z. Li, Y. Zeng, M. Ma, A novel floating memristor emulator with minimal components, Act. Passive Electron. Components 2017 (2017), https://doi.org/10.1155/2017/1609787.
[13] E.M.N. López, La naturaleza, las matemáticas, la ingeniería y el caos, Ingeniare Rev. Chil. Ing. 18 (1) (2010) 5–7, https://doi.org/10.4067/S0718-33052010000100001.
[14] H.I. Deniz, Z.G. Cam Taskiran, H. Sedef, An analog chaotic Lorenz circuit based on CCII+ and multiplier, in: *41st International Conference on Telecommunications and Signal Processing*, TSP, 2018, https://doi.org/10.1109/TSP.2018.8441256.
[15] X. Ye, Y. Wang, X.Y. Tang, H. Ji, B. Wang, Z. Huang, On the design of a new simulated inductor using a contactless electrical tomography system as an example, Sensors 19 (no. 11) (2019), https://doi.org/10.3390/s19112463.
[16] P. Saha, D.C. Saha, A. Ray, A.R. Chowdhury, Memristive non-linear system and hidden attractor, Eur. Phys. J.: Spec. Top. 224 (8) (2015) 1563–1574, https://doi.org/10.1140/epjst/e2015-02480-1.
[17] Y. Shaohui, R. Yu, S. Zhenlong, S. Wanlin, S. Xi, A memristive chaotic system with rich dynamical behavior and circuit implementation, Integration 85 (2022) 63–75, https://doi.org/10.1016/j.vlsi.2022.03.003, 2022.
[18] B. Muthuswamy, Implementing memristor based chaotic circuits, International Journal of Bifurcation and Chaos 20 (5) (2010) 1335–1350, https://doi.org/10.1142/S0218127410026514.
[19] B.C. Bao, Z. Liu, H. Leung, Is memristor a dynamic element? Electron. Lett. 49 (24) (2013) 1523–1525, https://doi.org/10.1049/el.2013.2788.
[20] B. Muthuswamy, L.O. Chua, Simplest Chaotic Circuit, January, 2014, https://doi.org/10.1142/S0218127410027076.
[21] Y. Guo, Q. Wu, X. Wang, A circuit implementation of random number generator utilizing memristor Stochasticity, in: International Conference on Advanced Computational Intelligence (ICACI), 2021, pp. 341–345, https://doi.org/10.1109/ICACI52617.2021.9435904.
[22] J. Melgarejo, A. Pirajan, Diseño y realización hardware de un generador de números aleatorios, Ingenieria 7 (2) (2002) 97–100, https://doi.org/10.14483/23448393.2825.
[23] E. Avaroğlu, T. Tuncer, A.B. Özer, B. Ergen, M. Türk, A novel chaos-based post-processing for TRNG, Nonlinear Dynam. 81 (1) (2015) 189–199, https://doi.org/10.1007/s11071-015-1981-9.
[24] N. Nguyen, G. Kaddoum, F. Pareschi, R. Rovatti, G. G Setti, A fully CMOS true random number generator based on hidden attractor hyperchaotic system, Nonlinear Dynam. 102 (4) (2020) 2887–2904, https://doi.org/10.1007/s11071-020-06017-3.
[25] V.H. Nguyen, K.Y. Sohn, H. Song, On-printed circuit board emulator with controllability of pinched hysteresis loop for nanoscale TiO 2 thin-film memristor device, J. Comput. Electron. 15 (3) (2016) 993–1002, https://doi.org/10.1007/s10825-016-0862-x.
[26] F. Corinto, O.V. Krulikovskyi, S.D. Haliuk, Memristor-based chaotic circuit for pseudo-random sequence generators, in: Proceedings of the 18th Mediterranean Electrotechnical Conference: Intelligent and Efficient Technologies and Services for the Citizen, Melecon, April 2016, pp. 18–20, https://doi.org/10.1109/MELCON.2016.7495319.