MÜHENDİSLİK ve DOĞA BİLİMLERİ FAKÜLTESİ

Uluslararası Mühendislik Araştırma ve Geliştirme Dergisi
International Journal of Engineering Research and Development

UMAGD, (2024) 16(2), 668-678.
10.29137/umagd.1392479

Cilt/Volume:16 Sayı/Issue:02 Haziran/June 2024

Araştırma Makalesi / Research Article

Generation of Random Numbers on a Microcontroller Platform

Rastgele Sayıların Mikrodenetleyici Platormunda Üretilmesi

Mustafa Sanlı [1]

[1]*Aselsan, Ankara, Türkiye*

Başvuru/Received*: 17/11/2023* ***Kabul / Accepted****: 05/03/2024* ***Çevrimiçi Basım / Published Online:*** *30/06/2024*
Son Versiyon/Final Version*: 30/06/2024*

Abstract
Microcontrollers are widely used in everyday applications as a result of their cheap and versatile nature. Recent advances in the fields of Internet of Things and Artificial Intelligence further increased the application areas of microcontrollers. A major problem in most microcontroller applications is the generation of random numbers with the limited hardware resources available. Existing methods which use the jitter in different clock sources or incorporate dedicated random number generators either lack operation speed or need addition of expensive hardware components. This paper uses the avalanche breakdown uncertainty in a transistor to generate random numbers on a microcontroller platform. In the context of this study, a hardware platform is designed to generate random numbers and generated data is analyzed through statistical methods. The presented solution is quite fast and cost effective in terms of both design budget and hardware resources.

Key Words
"Microcontroller, RNG, IoT, computer hardware"

Öz
Mikrodenetleyiciler ucuz ve çok yönlü olmaları nedeniyle günlük uygulamalarda yaygın olarak kullanılmaktadır. Son yıllarda Nesnelerin İnterneti ve Yapay Zeka alanlarında yaşanan gelişmeler mikrodenetleyicilerin uygulama alanlarını daha da artırmıştır. Mikrodenetleyici uygulamalarının en büyük sorunu, mevcut sınırlı donanım kaynaklarıyla rastgele sayıların üretilmesidir. Farklı saat kaynaklarındaki seğirmeyi kullanan veya özel rastgele sayı üreteçleri içeren mevcut yöntemler, ya çalışma hızından yoksundur ya da pahalı donanım bileşenlerinin eklenmesini gerektirir. Bu makale, bir mikrodenetleyici platformunda rastgele sayılar üretmek için bir transistördeki çığ kırılma belirsizliğini kullanmaktadır. Bu çalışma kapsamında rastgele sayılar üretecek bir donanım platformu tasarlanmış ve üretilen veriler istatistiksel yöntemlerle analiz edilmiştir. Sunulan çözüm hem tasarım bütçesi hem de donanım kaynakları açısından oldukça hızlı ve uygun maliyetlidir.

Anahtar Kelimeler
"Mikrodenetleyici, RNG, IoT, bilgisayar donanımı"

***Responsible Author:** msanli@aselsan.com.tr*

1. Introduction

Today, many technological applications rely solely on microcontrollers. These application areas range from consumer electronics to industrial automation. Microcontrollers' cheap and versatile nature opens the gate of endless use cases. Many microcontrollers cost less than a few dollars and they offer quite useful amount of RAM (Random Access Memory) and efficient number of input-output pins compared to this price.

As a consequence of advances in technological fields such as Internet of Things and Artificial Intelligence, the application areas of microcontrollers increased in recent years drastically. It is expected that more than 40 billion connected devices will be spread over the world by 2025 (Al-Sarawi et al., 2020). These devices contain sensors, actuators and communication interfaces. Many of these devices rely on microcontrollers for serving the required computational power, reading sensors over several interfaces such as SPI, I2C, RS232 and RS422, timing and control activities of actuators and many more logical operations.

Besides the advantages in terms of cost, availability and functionality, microcontrollers have inherent limitations in terms of the amount of computing power, speed, memory and input-output pins. Most microcontrollers operate at several MHz clock rates. This is more than enough for basic input-output operations such as reading a sensor and interfacing an external integrated circuit. However, when there is need for computationally complex algorithms, the microcontrollers fail to provide the desired speed. Also, memory is another concern for microcontrollers. Usually, data can be stored on an external RAM or flash memory and can be accessed by the microcontroller when necessary. However, the available memory on the microcontroller puts a strict limitation on the size of the firmware that contains the operational instructions for the microcontroller. Today, microcontrollers need to connect to sensors, memory chips, oscillators and many other interfaces at the same time. The number of input-output pins limits the maximum number of peripherals a microcontroller can communicate. As a consequence of these limitations, microcontroller design has inherent difficulties and concerns which is unique to the microcontroller world. Extensive care must be provided to firmware size and computational power budget in microcontroller-based solutions.

Random numbers are required in many microcontroller applications. They are used as cryptographic keys, initialization vectors, nonces and padding values. Random numbers are also necessary in robotic decision-making algorithms. The following list summarizes the role of random numbers in microcontroller applications.

- Security and Encryption: Random numbers are used for producing security keys, initialization vectors, and other security-related parameters in microcontrollers. Random numbers are used to create secure communication channels, such as in SSL/TLS protocols, ensuring data confidentiality and integrity (Tehranipoor et al., 2023).

- Randomization: Microcontrollers are used in gaming applications where random numbers are needed for gameplay, character generation, and more (Kneusel, 2018).

- Authentication: Random numbers are often used in authentication protocols to challenge users or devices to prove their identity securely (Tkacik, 2003).

- Noise Generation: In applications involving sensors and measurements, random noise is added to analog signals to improve resolution and accuracy (Tehranipoor et al., 2023).

- Machine Learning: In machine learning applications, random numbers are used for shuffling training data to improve model generalization and performance (Bhargava, 2019).

- Random Delays: In traffic signal control systems, random delays can be introduced to mimic real-world traffic variations and reduce congestion (Johnston, 2018).

- Quality Control: Random numbers are used in quality control processes for selecting random samples of products for inspection (Kneusel, 2018).

- Randomized Algorithms: Some sorting algorithms use random numbers for pivot selection, such as QuickSort and Randomized QuickSelect (Kneusel, 2018).

- Power Management: Microcontrollers can use random numbers to determine when to enter sleep mode and when to wake up, optimizing power consumption (Chung and Yung, 2011).

- Randomized IoT Applications: In IoT (Internet of Things) devices, random numbers can be used for secure device pairing, data encryption, and randomized reporting intervals to reduce network congestion (Rangarajan et al., 2021).

- Firmware Testing and Debugging: Random numbers can be used to simulate unpredictable conditions during testing, helping to identify and debug issues in firmware (Tehranipoor et al., 2023).

- Resource Allocation: In multi-user or multi-device scenarios, random numbers can help allocate resources fairly or randomly, preventing resource contention (Johnston, 2018).

The main difficulties in the generation of random numbers is finding a reliable source for randomness and applying a method to generate entropy which is the measure of disorder of a variable (Ferguson et al., 2010). Low entropy indicates that values can easily be predicted, while truly random data possesses high entropy. The source of randomness usually depends on analog sources. This source should not provide repeating results. However digital systems such as microcontrollers and FPGAs (Field Programmable Gate Array) are predictable and precise by their design nature.

The generation of random numbers with limited available resources is a major problem for microcontrollers. Most existing methods either use jitter from different clock sources or incorporate dedicated random number generators that lack operation speed or require the addition of expensive hardware components.

In this paper, an efficient method of generating random numbers on a microcontroller platform using the avalanche breakdown noise signal as a source of entropy is presented and implemented on a hardware platform. The proposed solution is quite fast and cost effective in terms of both design budget and hardware resources. The proposed solution can be used as a subsystem of the larger microcontroller applications that require random number generation. The remainder of the paper is organized as follows: Section 2 gives a summary of the related work about random number generation on hardware platforms. A method for generating random numbers using avalanche breakdown uncertainty of a transistor is presented in Section 3. Section 4 gives the test and evaluation results. Finally, Section 5 gives a conclusion and summarizes the paper.

2. Background

The idea of using avalanche noise as a source of entropy to generate random numbers was first proposed in (Lampert et al., 2016). However, the circuit used for the generation of random numbers included several op-amps, a boost converter, as well as comparator logic, which made the design heavily dependent on external components. While the proposed circuit may be useful in larger designs, many IoT applications require low-cost and simple solutions to generate random numbers.

In (Kwok and Lam, 2006), the researchers tackled the problem of random number generation on a Xilinx Virtex II Pro FPGA platform. In this work, the jitter of the Xilinx Digital Clock Manager (DCM) is used as a source of randomness. The slow clock signal generated by DCM is sampled by an accurate higher frequency clock signal to generate a sequence of bits. To achieve this, the high frequency clock is connected to the data input of a D type flip flop and the slower clock is fed into the clock input of the flip flop. In this design, the frequency of the fast clock needs to be selected as the same order of magnitude as the jitter of the DCM. This solution depends on two clock sources and DCM. Consequently, it is not possible to apply this design approach to microcontroller domain. Additionally, this work does not address the predictability of the source of randomness.

In (Tsoi et al., 2007), this problem is identified and an alternative solution is proposed. The researchers use a sequence of inverters and a feedback loop to create ring oscillation. It is proposed that with the addition of a delay element and an XOR gate, the ring oscillation can be doubled and the predictability of the ring is further decreased. The output of the ring is connected to the input of a D type flip flop and the system clock is fed into the clock input of this flip flop. It is argued that this solution is an improvement to the method presented in (Lampert et al., 2016) as the dependency on two separate clock sources is removed. However, this solution involves a number of sequential logic operations that are carried on the ring. While these consecutive operations may seem trivial for an FPGA platform, it is not suitable for implementation on microcontrollers. FPGAs are capable of doing many logical operations in parallel in a single clock cycle. However, microcontrollers can perform instructions one by one.

The researchers introduce another approach that gathers entropy from the jitter in different clock signals on a CPU in (Mowery et al., 2013). This approach is quite slow and cannot be used on embedded devices that use only a single clock source.

Different from these solutions which heavily depend on the clock signal, the study in (Tezuka and L'Ecuyer, 1991) proposes another approach that depends on predefined seeds. These seeds are fed into a Tausworthe Uniform Random Number Generator which initiates the shift registers with given seeds and rotates the content of the registers as long as the system is not reset. In this solution, the shift registers need to be operated with different clock frequencies to achieve better unpredictability. This results in dependency on multiple clock sources.

A commonly adopted approach for generating random numbers on FPGA is using linear feedback shift registers with feedback loops. This approach is preferred because it does not requires special hardware or complicated processing. However, the simple design has problems in reaching high speeds. The faster implementations require parallel processing (Schneier, 1996). Although this approach is easily implemented on FPGAs, thanks to the parallel execution capabilities of FPGAs, microcontrollers' sequential execution behavior makes it inappropriate for microcontrollers.

Using SRAM in microcontrollers as a source of entropy for generating random numbers is proposed in (Herrewege et al., 2013). In this study, the noise in the SRAM characteristics of two different types of microcontrollers, a STM32F100R8 and a PIC16F1825 are measured and evaluated. After analyzing over 1.000.000 measurements, it is concluded that PIC16F1825 is not suitable for securely generating seeds. This limits the applicability of the approach to only some types of microcontrollers.

There are studies that use unpredictable physical properties as a source of randomness such as independent oscillators fed into a shift register (Petrie and Connelly, 2000) and noise on the least significant bits of analog to digital converters (Moro et al., 2006). The main problems with these solutions are the low throughput of the random number generator and predictability/repeatability of the physical property.

There are several other approaches for generating random numbers (Bhattacharjee and Das, 2022; L'ecuyer, 2004; Datcu et al., 2020; Dridi et al., 2023; Tutueva et al., 2021; Levina et al., 2022; Gupta and Chauhan, 2021; Arciuolo and Elleithy, 2021; Pazos et al., 2023). These approaches either need dedicated hardware or complicated processing which make them inappropriate for microcontroller implementation.

3. Method

In this work, we use avalanche noise in a reverse-biased p-n junction to generate random numbers on an Atmega328 microcontroller platform. When a transistor is reverse biased, only a small current flows through its junction and the transistor can be said to behave like an open circuit. When the reverse bias voltage is increased further, finally at some point the current suddenly increases. This point is known as the breakdown point of the connection and the voltage at this point is known as the breakdown voltage (Monk, 2017). This voltage is the maximum amount of voltage that can be applied to a transistor in reverse bias operation. There are two possible types of breakdown: Avalanche breakdown and zener breakdown. The former occurs in lightly doped pn-junctions, while the latter is observed in highly doped junctions. In an avalanche breakdown, a strong electric field increases the kinetic energy of the carriers and they collide with neighboring atoms, creating new electron-hole pairs.

The process of avalanche breakdown is similar to a tiny particle accelerator. Within this accelerator, thermal noise is one of the mechanisms which enables electrons to travel to the depletion region of a PN junction. Here, they are accelerated by an electrical field until they form new pairs of electrons and holes. These high-energy electrons create an amplification cascade that has minimal repeatability when they collide with atoms in the depletion area. If this process turns into an avalanche effect, random noise occurs. This random noise that occurs when a junction is operated under an avalanche breakdown is called avalanche noise (Monk, 2017). The noise signal is generated by electrons passing through the silicon junction and their behavior is unpredictable. The base-emitter junction of an npn transistor can be used to create avalanche noise (Vasilescu, 2010). Figure 1 shows the basic transistor configuration we used to generate avalanche noise. The image of the avalanche noise produced by this circuit on the oscilloscope screen is given in Figure 2.

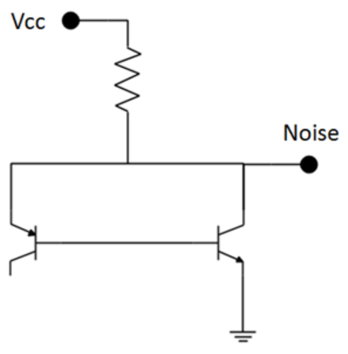

Figure 1. The transistor configuration that is used to create avalanche noise. Adapted from (Url-1).

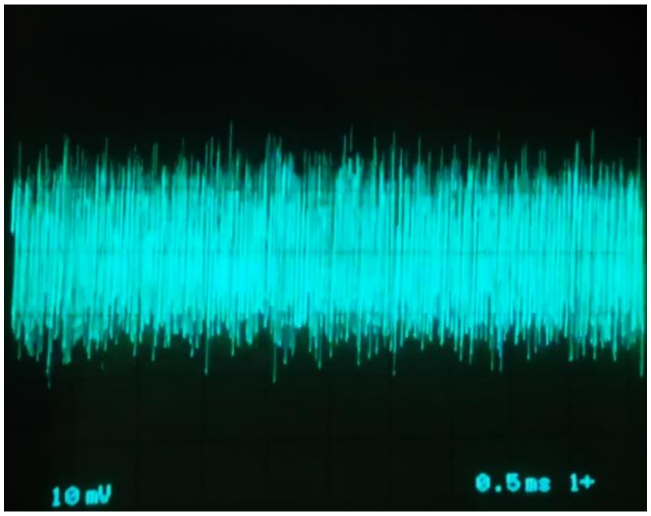

Figure 2. The image of the avalanche noise on the oscilloscope.

In order for the noise signal to be used as a source of randomness, it must be amplified and its level adjusted. The entire circuit used to generate the random analog signal is given in Figure 3. The circuit elements and values used in design are presented in Table 1. The circuit uses a total of only 3 transistors, 2 capacitors and 4 resistors. In the figure, T1 and T2 transistors are used to produce the avalanche noise. Transistor T3, together with capacitor C1 and resistor R4, is used to amplify the noise signal. Resistors R2 and R3 form a voltage divider and feed the signal to the analog input pin of the microcontroller. Like most commercial microcontrollers available today, the ATMega328 has built-in analog-to-digital converters (ADC) that are connected to some input pins. The resolution of the ADC is 10 bits. The sampling frequency is 9.6 Hz. We used one of these pins for this application. After digital conversion, the microcontroller receives the value of noise measurement in digital format. In our design, the microcontroller takes readings from the ADC at a frequency of 8 Hz. This allows the microcontroller to generate a random number every 125 ms. It is possible to increase the readout frequency up to 9.6 Hz, which is the limit set by the ADC's sampling frequency for 5 V operation and 125 kHz ADC sampling clock (Url-4). When faster random number generation speed is needed, a faster external ADC can be used to increase the generation speed.

The analog noise signal used for random number generation is usually biased. That is, the probability of the number 0 or 1 occurring at any given moment may not be equal. Moreover, this behavior may change over a long time depending on physical conditions. Before generating a random 0 or 1 output using this noise measurement, there is a need for removing any possible bias in this process. To achieve this, after every 1024 readings the microcontroller calculates a mean. This new mean value is used to fully digitize the analog signal. If the value of the noise signal is greater than the mean, the random number generator outputs 1, otherwise 0.

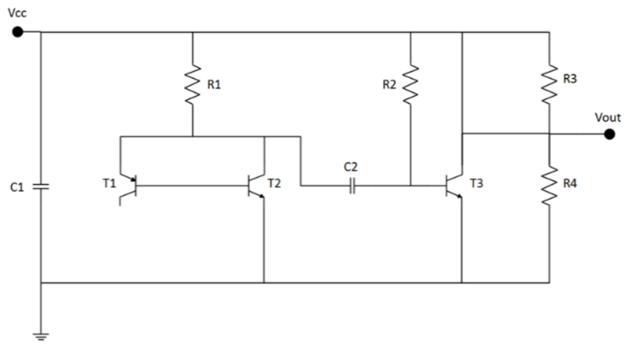

Figure 3. The circuit used in the generation of the random analog signal. Adapted from (Url-2)

Table 1. Circuit elements and values used in design

Component	Value
R1	5.6 KΩ
R2	5.6 KΩ
R3	10 KΩ
R4	1.8 MΩ
C1	100 µΩ
C2	100 nF
T1	2N3904
T2	2N3904
T3	2N3904

We designed and manufactured a circuit board specifically for the hardware implementation of this method. The hardware platform includes an ATMega328 microcontroller, random analog signal generation circuitry, and various programming and communication interfaces for data acquisition via the computer. The printed circuit board (PCB) is designed with special care to minimize electromagnetic interference (EMI) from surrounding circuits on the noise line. To achieve this, we used advanced PCB design techniques such as electromagnetic shielding, trace separation, and utilizing solid ground planes when designing the PCB. Figure 4 shows the PCB before electronic components are placed on it.

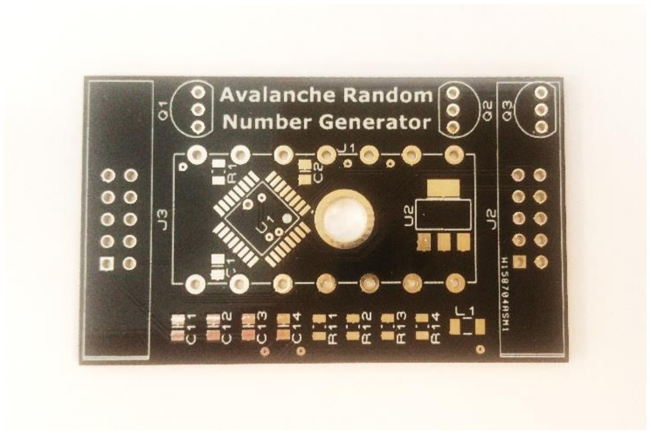

Figure 4. Printed circuit board specially designed for random number generator.

Although similar circuits have been used for avalanche noise generation in previous random number generator studies such as (URL-1; URL-2), our solution, which includes a simple bias removal technique and EMI-oriented PCB design, stands out as a suitable solution for IoT applications.

4. Results and Discussion

To analyze the efficiency of the proposed method, we need to examine the uniformity in the random number generation process. To achieve this, we first tested our random generator using Monte Carlo simulation. We then applied another statistical test, the chi-square test, to the generated random output. After, we visually observed the randomness in the output using bitmap images. Finally, we tested the randomness in the sequence of numbers produced by the random number generator at different temperatures with the dieharder testing tool.

4.1 Monte Carlo Simulation

We used an application of Monte Carlo simulation to test for uniformity in the random distribution. Monte Carlo simulation is a statistical technique used to test the uniformity or randomness of a distribution because it allows random samples to be generated and their properties to be examined. This simulation method is particularly useful in cases where it is difficult or impossible to prove the uniformity of a distribution analytically or when dealing with complex systems or data. Monte Carlo simulations involve generating a large number of random samples from a given distribution. If the distribution is truly uniform or random, the generated samples should show features consistent with uniformity.

Monte Carlo methods are used to provide numerical approximations to problems that are not possible or feasible to solve exactly. In Monte Carlo simulations, a number of independent random trials are used for approximation. The results of the independent trials are averaged to get an approximation to the solution. The Law of Large Numbers states that as the number of independent trials is increased, the approximation converges to the exact result (Howes and Thomas, 2007).

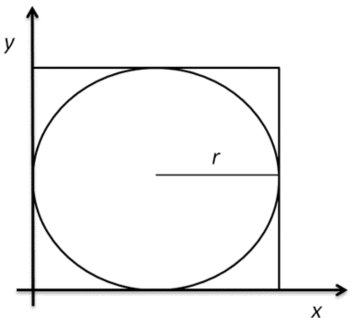

Figure 5. A circle drawn inside a square in the first quadrant of the x-y plane.

Figure 5 shows a circle drawn inside a square in the first quadrant of the x-y plane. Let *r* be the radius of the circle and *p* be the ratio of the circle area to square are. Then, *p* is calculated as:

$$p = \frac{\pi r^2}{4r^2} \qquad (1)$$

$$p = \frac{\pi}{4} \qquad (2)$$

For a Monte Carlo simulation where random numbers are used to generate *x* and *y* coordinates of the points with coordinate pairs (*x*, *y*), some fraction of the randomly generated points will lie inside the circle. Let *n* be the number of points generated in a simulation and *k* be the number of points lying inside the circle. Then this fraction can be calculated as:

$$p = \frac{k}{n} \qquad (3)$$

Then, using Equation 2 and Equation 3, π can be approximated as

$$\pi \approx \frac{4k}{n} \qquad (4)$$

In Monte Carlo simulation, Equation 4 is used to get an approximation to π. The degree of convergence to π indicates the degree of uniformity in randomness.

In our tests, we generated 100.000 random numbers on our hardware random generator and transmitted these numbers to our test PC through the RS-232 interface. We prepared a computer program in Visual C# to apply Monte Carlo simulation to the generated random numbers. In the first test case, we run the Monte Carlo simulation with the first 5000 random numbers in our list of 100.000 numbers. In the second test case, we run the simulation with the first 25.000 numbers. Finally, in the third case, we run the simulation with all of the 100.000 numbers. The test results are given in Table 2.

Table 2. Monte Carlo simulation results

Test case	Number of points	Approximation value
1	5000	3.1245
2	25000	3.1327
3	100000	3.1458

The simulation results show that the approximation closely matches the exact value, π, in all three cases. This indicates the uniformity in random distribution. Additionally, as the number of points increases, the accuracy improves, giving better estimates of π.

4.2 Chi-Square Test

The chi-square test is used to measure randomness in a set of data. The test calculates an absolute number and a ratio that indicate the level to which the tested sequence is supposed to be non-random. A ratio higher than 99% or lower than 1% indicates that the sequence is definitely not random. A ratio between 99% - 95% or 1% - 5% means the sequence is questionable. A ratio between 90% and 95% and between 5% and 10% indicates that the series is almost questionable. We used SPSS software to apply the chi-square test to our 100,000-bit long output sequence. The calculated distribution is 0.74 and the percentage value is 50%. These results show that our sequence passes the chi-square test and shows a high degree of randomness.

4.3 Random Bitmaps

Finally, after rigorously testing the proposed random number generator statistically with Monte Carlo and chi-square tests, we tried to visually observe the randomness. To achieve this, we created a bitmap image from the bit sequence generated by the random number generator. In this bitmap image, each pixel represents one bit. We created 3 bitmap images from three different sequences of different lengths. Bitmap images generated from random sequences of length 10000, 40000 and 90000 bits are presented in Figures 6, 7 and 8, respectively. Although this test does not provide any statistical information like Monte Carlo or chi-square tests, it does allow visual inspection of the randomness in the generated bit sequences.

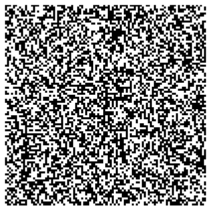
Figure 6. A bitmap image created from a random bit sequence of length 10000 where the bits are arranged in a 100x100 matrix.

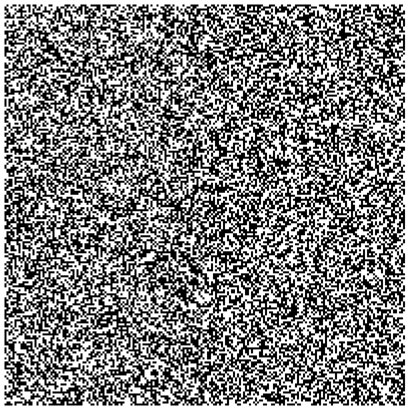
Figure 7. A bitmap image created from a random bit sequence of length 40000 where the bits are arranged in a 200x200 matrix.

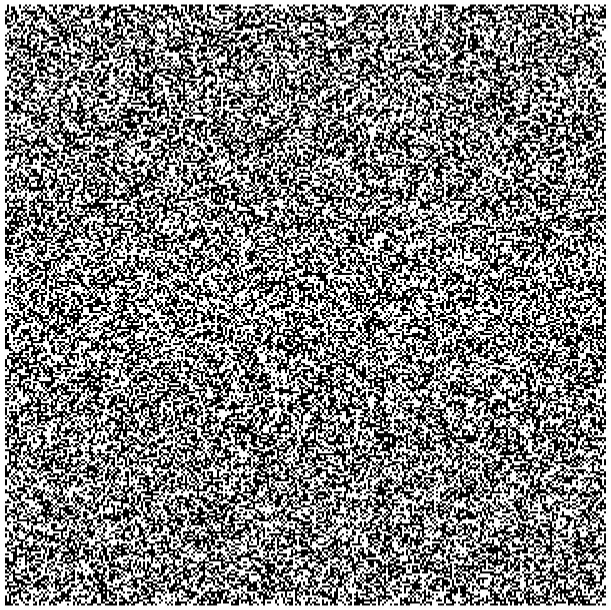

Figure 8. A bitmap image created from a random bit sequence of length 90000 where the bits are arranged in a 300x300 matrix.

4.1. Dieharder Test

Dieharder is a tool used to test randomness in a sequence of numbers (Url-3). The Dieharder tool requires a long sequence of random numbers to test for randomness in the data. We obtained test data by connecting a computer to the microcontroller via the UART interface and running the random number generator to obtain a random sequence 100 Mbit long. After collecting the data, we ran the Dieharder test on the PC and performed the statistical tests included in the Dieharder tool at room temperature. These tests include monobit test, execution test and serial test. The monobit test counts the number 1 in a sequence of bits and evaluates the result against the anticipated number. The runs test keeps track of the number of consecutive 0 and 1 bits in the given sequence. Serial test gathers the frequencies of overlapping bits in n-bit groups. While the monobit and run tests are run only once, the serial test is repeated many times with different n values. Table 3 summarizes the test results obtained from the statistical tests of the Dieharder tool. The calculated p values are used by the tool to arrive at a test result. A "Passed" test result confirms the randomness in the tested sequence of numbers.

Table 3. Results obtained from the statistical tests of the Dieharder tool at room temperature

Test Type	Calculated p-value	Result
STS Monobit Test	0.78	Passed
STS Runs Test	0.59	Passed
STS Serial Test (multi iteration)	-	Passed

We repeated the test at three different temperatures (5°C, 20°C, 35°C) on a temperature-controlled box to observe the effect of changes in environmental conditions, such as temperature, on randomness. Table 4 summarizes the test results.

Table 4. Results obtained from the statistical tests of the Dieharder tool at different temperatures

Temperature	Test Type	Calculated p-value	Result
5°C	STS Monobit Test	0.63	Passed
5°C	STS Runs Test	0.51	Passed
5°C	STS Serial Test (multi iteration)	-	Passed
20°C	STS Monobit Test	0.73	Passed
20°C	STS Runs Test	0.48	Passed
20°C	STS Serial Test (multi iteration)	-	Passed
35°C	STS Monobit Test	0.69	Passed
35°C	STS Runs Test	0.53	Passed
35°C	STS Serial Test (multi iteration)	-	Passed

As a result of the bias removal technique we applied to get rid of the effects of possible environmental or hardware sources that may affect the noise signal, the test results confirm that the random number generator can reliably provide random sequences at different operating temperatures.

5. Conclusion

In an environment of constantly evolving technology, the need for high quality random numbers is continuously increasing. This paper explores the fundamental task of generating random numbers on microcontrollers. The limited computational resources of microcontrollers cause many challenges in performing this task.

In this paper, we use the unpredictability in avalanche breakdown of a transistor to generate random numbers on a microcontroller. Within the scope of this work, we designed a microcontroller based electronic board and implemented the presented approach on this hardware platform.

Practical testing and verification confirmed the reliability of our system, which can serve as an essential tool for applications requiring secure, unpredictable data, such as cryptographic protocols and secure key generation. Statistical analysis of test results by Monte Carlo simulation and chi-square test confirms the randomness in the generated data. Statistical tests performed using the Dieharder tool also verify random behavior in the generated sequence of numbers.

It is clear that generating random numbers on a microcontroller platform is very important in ensuring data security and increasing the robustness of various embedded systems. In today's world where digital privacy and security are crucial, our contribution helps ensure more secure and reliable data processing and communications.

References

Al-Sarawi, S., Anbar, M., Abdullah, R. & Al Hawari, A. B. (2020). Internet of Things market analysis forecasts, 2020–2030. Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 27-28 July 2020, London, UK.

Arciuolo, T.F., & Elleithy, K.M. (2021). Parallel, True Random Number Generator (P-TRNG): Using Parallelism for Fast True Random Number Generation in Hardware. 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), 0987-0992.

Bhargava C. (2019). AI Techniques for Reliability Prediction for Electronic Components, IGI Global, Hershey, PA, USA.

Bhattacharjee, K. & Das, S. (2022). A search for good pseudo-random number generators: Survey and empirical studies. Computer Science Review, 45.

Chung Y. & Yung M. (2011). Information Security Applications, Springer, Berlin, Germany.

Datcu, O., Macovei, C., & Hobincu, R. (2020). Chaos Based Cryptographic Pseudo-Random Number Generator Template with Dynamic State Change. Applied Sciences.

Dridi, F., El Assad, S., El Hadj Youssef, W., & Machhout, M. (2023). Design, Hardware Implementation on FPGA and Performance Analysis of Three Chaos-Based Stream Ciphers. Fractal and Fractional.

Ferguson, N., Schneier, B. & Kohno, T. (2010). Cryptography Engineering: Design Principles and Practical Applications. Wiley, New York, NY, USA.

Gupta, M.D., & Chauhan, R.K. (2021). Hardware Efficient Pseudo-Random Number Generator Using Chen Chaotic System on FPGA. J. Circuits Syst. Comput., 31, 2250043:1-2250043:14.

Herrewege, A. V., Leest, V., Schaller, A., Katzenbeisser, S., Verbauwhede I. (2013). Secure PRNG Seeding on Commercial Off-the-Shelf Microcontrollers. Workshop on Trustworthy Embedded Devicesi 4 Nov. 2013, Berlin, Germany.

Howes, L. & Thomas, D. (2007). Efficient random number generation and application using CUDA, Addison Wesley, New York, NY, USA.

Johnston, D. (2018). Random Number Generators—Principles and Practices, De|G Press, Berlin, Germany.

Kneusel, R.T. (2018). Random Numbers and Computers, Springer, Cham, Switzerland.

Kwok, S.H.M. & Lam, E.Y. (2006). FPGA-based High-speed True Random Number Generator for Cryptographic Applications. IEEE Region 10 Conference, 14-17 Nov. 2006, Hong Kong, China.

Lampert, B., Wahby, R. S., Leonard, S., Levis, P. (2016). Robust, low-cost, auditable random number generation for embedded system security. 14th ACM Conference on Embedded Networked Sensor Systems (SenSys), 14-16 November 2016, Stanford, CA, USA.

L'Ecuyer, P. (1994). Uniform random number generation (1994). Annals of Operations Research, 53, 77 – 120.

L'ecuyer, P. (2004). Handbook of Computational Statistics, Springer Verlag, New York, NY, USA.

Levina, A., Mukhamedjanov, D., Bogaevskiy, D., Lyakhov, P.A., Valueva, M.V., & Kaplun, D. (2022). High Performance Parallel Pseudorandom Number Generator on Cellular Automata. Symmetry, 14, 1869.

Monk, S. (2017). Electronics Cookbook, 1st edition, O'Reilly Media, Sebastopol, CA, USA.

Moro, T., Saitoh, Y., Hori, J.& Kiryu, T. (2006). Generation of Physical Random Number Using the Lowest Bit of an A-D Converter. Electronics and Communications in Japan (Part III: Fundamental Electronic Science), 89(6), 13–21.

Mowery, K., Wei, M., Kohlbrenner, D., Shacham, H. & Swanson, S. (2013) Welcome to the Entropics: Boot-Time Entropy in Embedded Devices. IEEE Symposium on Security and Privacy, 19-22 May 2013, Berkeley, CA, USA.

Pazos, S.M., Zheng, W., Zanotti, T., Aguirre, F.L., Becker, T.E., Shen, Y., Zhu, K., Yuan, Y., Wirth, G.I., Puglisi, F.M., Roldán, J.B., Palumbo, F., & Lanza, M. (2023). Hardware implementation of a true random number generator integrating a hexagonal boron nitride memristor with a commercial microcontroller. Nanoscale.

Petrie, C. & Connelly, J. (2000). A Noise-based IC Random Number Generator for Applications in Cryptography. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 47(5), 615 –621.

Rangarajan N., Patnaik S., Knechtel J., Rakheja S.& Sinanoglu O. (2021). The Next Era in Hardware Security, Springer, Cham, Switzerland.

Schneier, B. (1996). Applied Cryptography. John Wiley and Sons, New York, NY, USA.

Tehranipoor M., Pundir N., Vashistha N., & Farahmandi F. (2023). Hardware Security Primitives, Springer, Cham, Switzerland.

Tezuka, S. & L'Ecuyer, P. (1991). Efficient and portable combined Tausworthe random number generators. ACM Transactions on Modeling and Computer Simulation 1(2), 99–112.

Tkacik, T .E. (2003). A Hardware Random Number Generator. In: Kaliski, B.S., Koç, Ç.K., Paar, C. (eds) Cryptographic Hardware and Embedded Systems - CHES 2002. CHES 2002. Lecture Notes in Computer Science, vol 2523. Springer, Berlin, Heidelberg.

Tsoi, K. H., Leung, K. H. & Leong, P. H. W. (2007). High performance physical random number generator. IET Computers & Digital Techniques, 1(4), 349-352.

Tutueva, A.V., Karimov, T.I., Moysis, L., Nepomuceno, E.G., Volos, C.K., & Butusov, D.N. (2021). Improving chaos-based pseudo-random generators in finite-precision arithmetic. Nonlinear Dynamics, 104, 727 - 737.

Url-1. https://web.jfet.org/hw-rng.html Accessed: 1.11.2023.

Url-2. https://robseward.com/misc/RNG2 Accessed: 1.11.2023.

Url-3. https://webhome.phy.duke.edu/~rgb/General/dieharder.php Accessed: 1.11.2023.

Url-4. https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf Accessed: 1.11.2023.

Vasilescu, G. (2010). Electronic Noise and Interfering Signals: Principles and Applications. Springer Berlin& Hiedelberg, New York, NY, USA.