Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 547 – 561

doi:10.3906/elk-2104-67

Cryptographically strong random number generation using integrated CMOS photodiodes for low-cost microcontroller based applications

Baykal Sarıoğlu*
Department of Electrical and Electronics Engineering, Faculty of Engineering and Natural Sciences,
İstanbul Bilgi University, İstanbul, Turkey

Received: 10.04.2021 • **Accepted/Published Online:** 30.07.2021 • **Final Version:** 21.03.2022

Abstract: In this work, we propose a method to generate random numbers for low-cost, low-power, resource-limited low data-rate microcontrollers using integrated CMOS photodiodes. The proposed method utilizes an integrated CMOS photodiode in the photovoltaic mode as the entropy source. The method is based on serially capturing analog values derived from the integrated CMOS photodiode. The entropy of these values increased by a custom algorithm. The proposed random number generator is devised using an integrated CMOS photodiode manufactured in 180 nm standard CMOS technology. The wide applicably of the random number generator is demonstrated by realizing it on a low-cost Arduino UNO board placed in a typical room environment. The implemented random number generator passes NIST-SP800-22 and AIS31 randomness tests at high scores. The proposed method achieved 5.4 Kbps throughput and 7.2% total significance level without any postprocessing. The test results show the high cryptographical strength of the proposed method makes it a promising alternative to the currently used random number generation algorithms in low-cost, low-resources, low-data rate microcontroller-based applications.

Key words: Random numbers, CMOS photodiode, embedded systems

1. Introduction

Random numbers are critical components in any cryptographic process to achieve unpredictability and secure operation. The added unpredictable behavior with the random numbers is one measure to prevent intrusions to sensitive information. The encryption algorithms used in network communications require random numbers to achieve high security in the data transfer. On the other hand, creating such strong random numbers for these secure processes is not a trivial task and can be challenging. In practice, there is a balance between generating random numbers quickly and generating them cryptographically strongly. Quick generation schemes generally result in weak random number sequences. Hence, currently, for the generation of random numbers, there are two main generation methods in cryptographic applications; (1) pseudo-random number generation (PRNG) and (2) true-random number generation (TRNG).

The PRNG development goes back to Jon von Neumann [1]. Von Neumann introduced the idea to generate pseudo randomness using various algorithms. In general, PRNG uses a deterministic algorithm to generate the numbers from a distribution such as Poisson, Gaussian, exponential, and Cauchy. However, in order to generate a random number, a PRNG requires an initial seed number. If this seed number is provided identical, the random number sequence generated in each execution is also identical, which makes them poor in

*Correspondence: baykal.sarioglu@bilgi.edu.tr

terms of secure operations [2]. If the seed and the algorithm are known, the random number can be guessed. However, their fast execution time makes them very suitable for most digital systems [3–7], and therefore they are very popular in most embedded system applications. For example, the highly practiced and standardized C programming language contains a pseudo-random number function called *rand*() in the stdlib library. This algorithm is short-cycle and predictable. Another, better standard C random number generation algorithm in the same library is called *Posixrandom*(), which provides better randomness, however, it is still a PRNG and requires an initial seed. In order to create better randomness, the Park–Miller generator is added in C++ as *minstd* function. *arcrandom*4() supported by BSD is another PRNG, requires BSD support in the embedded system. In the Microsoft Windows platforms *BCryptGenRandom*() can be benefited for generating strong random numbers, however, it requires a powerful embedded system that can run the MS Windows operating system. Another popular programming language Python incorporates a PRNG algorithm called Mersenne Twister [16] for generating random numbers with *random*() in the random.py library. Then again, besides its random function's cryptographically weaker nature, Python is also itself a resource-heavy programming language for implementing on 8-bit and 16-bit microcontrollers with low resources. Similarly, achieving a better random number generation algorithmically in Python or similar other high-level languages can be resource-heavy for low-cost microcontrollers.

In TRNGs, physical entropy sources are employed instead of mathematical models [8]. These generators do not require any initial seed and the generated sequence of numbers are unique and not repeatable. The entropy source is sampled and digitized. Then various statistical tests are performed for measuring the strength of the generator. As the entropy source thermal noise on electrical components [9, 10], phase jitter in oscillators [11–14], chaos [15, 17–20], spintronic [21] or optical sources [22] can be utilized. These implementations require advanced setups that require complex instrumentation and cover a large area. Therefore, they are not suitable for low-cost microcontroller-based applications. Another solution to gather a random number is using a web-based random number generator service. With this method, the necessary complexity of the true random number generation and the usage of extra resources for the generation process are overcome by remote computing solutions [23, 24]. However, for this type of solution, it is essential to provide internet connectivity to the embedded system. This means using additional hardware and software components in the application, and therefore, increasing the overall cost. Additionally, an internet service must also be present in the environment where the system is placed.

The strong number generation is required not only for high-performance systems, but it is also a necessity for low-cost electronic systems where data security is critical; such as data logging, smart locks [25] and low data rate wireless communication systems for IoT [26]. On the other hand, for such low-cost embedded system applications in which the hardware resources are limited, it is a challenge to create a random number generator both strongly secure and resource-efficient [27, 28]. These systems are generally battery-powered and lack high computing power. In this work, a fast, compact, easy-to-implement, ultralow power, cryptographically strong random number generator that is suitable for low-cost microcontroller applications that requires data security is proposed. In the proposed random number generator, an integrated CMOS photodiode manufactured in 180 nm standard CMOS technology is used as the entropy source in a photovoltaic mode that does not consume any power. Moreover, the proposed method enables the utilization of the CMOS photodiode for both signal reception and random number generation in low-power visible light communication systems.

2. Related work

In the literature, using optical components as quantum entropy source for a TRNG is commonly reported [22, 29–33]. For example, an intradyne receiver is utilized to achieve wide noise bandwidth of 11GHz [22]. It is also possible to use the random radiation intensity of superluminescent light-emitting diode (SLED) to generate random numbers [29]. Silicon nanocrystals light source can also be used as an entropy source for optical random number generation [30]. Generating random numbers using the optical heterodyne of two chaotic optical-feedback semiconductor lasers as entropy source is also reported [31]. The dark current noise in active-pixels in CMOS image sensors can also be exploited for generating true random numbers [32]. These implementations provide high performance and strong randomness due to the quantum nature of the entropy source. However, they mostly require advanced setups, high perfromance computing units (DSP [22], FPGA [29, 30], PC CPU [32], and methods that are not easily available for low-cost embedded system implementations.

In the literature, FPGA as an embedded processing unit is mainly preferred for generating true random numbers using optical entropy sources due to its up to 343 Gbps throughput capabilities [34]. However, the number of reported random number generators for low-cost low-power embedded systems are highly limited in the literature [27, 28, 35–37]. These implementations are designed mostly for obtaining high-entropy keys [35]. For obtaining such keys, noisy entropy sources are used such as biometrics [36], quantum information [37]. Yet again, these reported implementations are utilized on RISC-based CPUs where the CPU frequency is in order of 100 MHz. Achieving a random number generator with a resource-limited low-cost low-clock frequency embedded system remains a challenge.

3. System description

Figure 1 shows the block diagram of the proposed random number generation method firmware. An integrated CMOS photodiode is connected to the ADC pin of a microcontroller. It is important to note that the proposed method can be utilized in any embedded system that contains an available ADC input pin. In order to get the random numbers, the photovoltaic mode of the CMOS photodiode which serves as the entropy source for the proposed random number generator is utilized. The proposed method also allows the usage of the integrated CMOS photodiode as an optical signal receiver for various other tasks in the application.

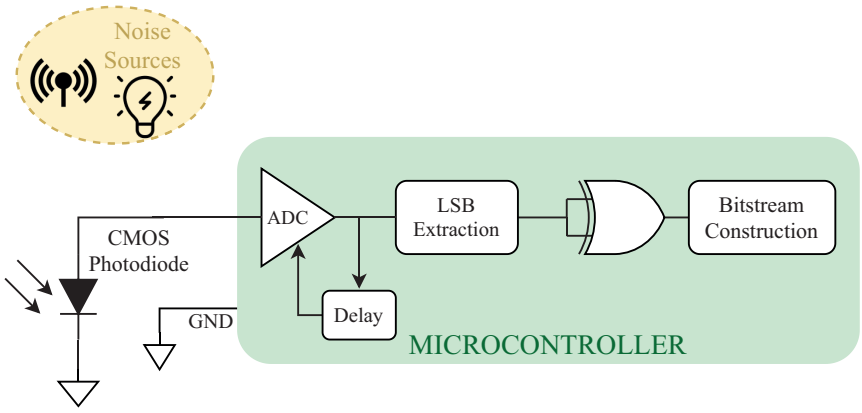


Figure 1. Block diagram of the proposed random number generation method.

3.1. Integrated CMOS photodiode as the entropy source

The proposed method depends on the noise performance of the integrated CMOS photodiode. The overall internal noise of the photodiode is determined by shot noise, thermal noise, flicker noise, and generation-recombination noise factors [38]. In the photovoltaic mode employed in the proposed method, the thermal noise due to the shunt parasitic resistance of the photodiode dominates the overall noise. The noise current, I_j, generated by the thermal noise is given by:

$$I_j = \sqrt{\frac{4kTB}{R_{sh}}}, \tag{1}$$

where k is the Boltzmann constant, T is the temperature, B is the bandwidth, and R_{sh} is the shunt resistor of the photodiode. It is important to note that the thermal noise is internal to the photodiode. There are also external noise sources, such as emission fluctuations of the optical sources nearby the photodiode. Figure 2b shows the equivalent circuit model of a photodiode and the parasitic components in the photodiode model. For gathering the external noise sources, the diode parasitic, C_d, R_d and R_s, are the important parameters. The parasitic capacitance affects the performance of the proposed method significantly. It is important to note that besides the optical noise sources also the electromagnetic noise sources coupled to the system through connected cables and terminals of the photodiode. In a reverse-biased photodiode the capacitance is dominated by the junction capacitance $C_j = \sigma_{Si}\sigma_0 A/W_d$, where A is the photosensitive area, W_d is the depletion region depth. However, since the ADC requires positive voltage values for proper conversion this region is not suitable for the proposed method. In the photovoltaic mode, the photodiode is forward biased. In this mode, the diffusion capacitance ($C_d = d\Delta Q/d\Delta V_d$, where Q is the total charge in the capacitor and V_a is the bias voltage.) is dominant over the junction capacitance. For an integrated photodiode, the capacitance is generally in fF range [39, 40] and it is significantly lower than the capacitance of off-the-shelf photodiodes (in $pF - nF$ range) is due to the much shallower junctions and less total charge (Q) present in the submicron standard CMOS processes (50–300 nm). Therefore, compared to the off-the-shelf counterparts, most CMOS photodiode implementations are unable to efficiently filter out the high frequency optical and electrical noise coupled to their terminals and photosensitive areas. This results in lower spectral responsivity and a higher noise profile than the off-the-shelf solutions. Thus, the integrated photodiodes in the standard CMOS processes are relatively poor optical-to-electrical converters but this situation makes them good entropy sources for random number generation in a noisy environment.

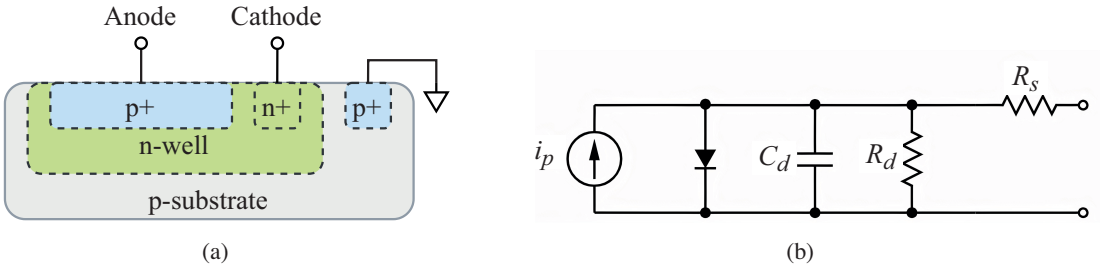


Figure 2. (a) Cross section of an integrated p+/n-well junction photodiode in a standard CMOS process, (b) Equivalent circuit model of a photodiode.

3.2. Random number generation implementation

In the proposed method, the focus is on implementing a random number generator for low-cost microcontrollers. In contemporary implementations of random number generators in microcontrollers, PRNGs are utilized mostly for their efficiencies. The PRNGs mostly require to be provided by an initial random seed to achieve cryptographically strong sequences of random numbers. If the random seed number not provided, however, they are cryptographically strong, the generated sequence of random numbers is always identical. This circumstance causes low-cost microcontrollers, not suitable for most cryptographic applications that require random numbers.

Algorithm 1 Pseudo-code of the proposed random number generation algorithm.

```
1: procedure GENERATE RANDOM NUMBER
2:     Begin configuration of the analog pool registers
3:     Selection of analog input channel
4: loop:
5:     Clear ADC register
6:     Set ADC input channel for photodiode sensor
7:     Start conversion
8:     Get ADC input
9:     Store ADC output as adcout1
10:    Wait for adcout1 amount (for performance purposes microseconds as the time unit should be preferred)
11:    Clear ADC register
12:    Get ADC input
13:    Store ADC output as adcout2
14:    Extract LSB from adcout1
15:    Extract LSB from adcout2
16:    XOR LSBs and store the bit
17:    if Stored number of bits = 8 then Construct Integer by performing bitwise concatenation
18:    Reset variables
```

To achieve cryptographically strong random numbers, the algorithm expressed in Algorithm (1), that exploits the least significant bit (LSB) of the ADC binary output is proposed. The LSB in the ADC binary output is the most sensitive digit to noise and hence it contains the most binary entropy. In the proposed algorithm, the ADC output value is stored as an integer. The LSB is extracted from this stored integer. Then, the stored integer is used to apply a finite delay. The delay time unit can be arranged according to the embedded system in use. However, this delay directly affects the execution time of the random bitstream generation. Therefore, the smallest possible time delay should be preferred for increasing the frequency of the random bit generation. Hardware timer units or simple software delays can be utilized in this step. After the delay period, another integer and related LSB are gathered from the ADC output. The LSBs of the two integers are then subject to bitwise XOR operation to obtain a final bit output. Bitwise XOR operation is preferred over other bitwise operations, i.e. AND, OR, due to the better performance related to the null elements theorem in Boolean algebra. That is, with a bitwise AND operation '0' bit value becomes dominant, while in a bitwise OR operation '1' bit value dominates the output in the proposed serial bitstream generation process. The XOR operation also eliminates the long runs of '1' s in the bitstream, which is an important feature of a strong random bitstream. It is important to note that the proposed algorithm does not belong to the XOR shift family of random number generator [41], due to the lack of an initial seed and shift operations in the process. The bit value obtained from the bitwise XOR operation is stored and then used to generate a bitstream. The number of consecutive sampling and XOR operations can be increased to achieve stronger random numbers. If random

integer numbers are required, integer numbers are created by applying bitwise concatenation to the generated bitstream.

The bit-depth of ADC is another important factor in the proposed method. The method depends on fluctuation on the ADC input hence an analog voltage input value higher than ADC resolution is essential. The resolution of an ADC can be calculated as $VDD \times 2^{-n}$, where VDD is the supply voltage of the microcontroller, n is the bit-depth of the ADC. The most current generation low-cost microcontrollers contain a 10-bit ADC, hence with a 5V supply voltage value, the resolution can be calculated as 4.88mV for these microcontrollers. With a 12-bit ADC, this value drops to 1.22 mV, which is a significant four-fold drop. The analog input of the ADC should at least reach this voltage level to change the LSB. To get a positive voltage on the ADC input pin, the pin can be left floating with a high impedance to act as a monopole antenna and receive EM radiation in the environment. This is the most favorite method to get the initial random seed for PRNGs in microcontrollers. However, it is demonstrated in the experimental results section of this work that the floating ADC input pin method does not bring about strong random numbers by itself. Hence, the seed generation itself is not a strong random number generation, which can lead to security problems for repetitive encryption operations. But then again, this case shows us that it is at most important to distinguish where the noise originated when ADC, cables, and PCB are used together in a random number generator. The wrong assessment can give rise to the faulty assumption that the entropy source is the cause of the observed randomness. The proposed algorithm does not include any sort of loops and complex computations. Hence, the computation complexity of the method can be given as O(n), since it directly depends on the number of bits generated.

3.3. Limitations of the proposed method

It is important to note that the proposed method is designed for low cost, resource-limited embedded systems that require cryptographic operations (e.g., encryption of transmitted data in IR or low data rate wireless communications in IoT) and it is not suitable for high-performance high resource-demanding applications (e.g., blockchain) that may require Gbps throughput. This limitation comes from both CPU speed (that is in 2 to 16 MHz range for most low-cost microcontrollers) and related to this fact, the rather slow sampling time of the ADC in these microcontrollers. As a result of the low-frequency clock input present, the throughput of the proposed method suffers. The overall throughput of the proposed algorithm can be calculated as:

$$f_{rng} = \frac{1}{t_{conv} + t_{exec} + t_{delay} + t_{com}}, \tag{2}$$

where t_{conv} is the conversion time of the ADC, t_{exec} is the execution time of the computations in the proposed algorithm (successive XOR and optional bitwise concatenation), t_{com} is the optional transmission time of the generated numbers over serial communication. If the generated numbers will be used internal operations on the same device, t_{com} can be ignored. If we consider implementing the proposed algorithm on the popular Arduino platform the resulting throughput can be calculated as; for ATmega 328p microcontroller, $t_{conv} = 13/f_{ADC}$ if we ignore the setup of the ADC; $t_{exec} \approx 300/f_{CPU}$; t_{delay} is dependent on the received data; which can be between 0 and 1023. We may assume $t_{delay} = 100\mu s$ as an average for a typical room condition with microseconds range delay is employed. The resulting throughput of the proposed random number generation for this case is then found to be $f_{rng} \approx 5.4Kbps$ which is suitable for most low-cost, low-power embedded system applications with low data rate [42]. Due to the relatively slow throughput of the system, the generation of moderately large files sizes (>10MB) requires a long duration. For instance, with 5.4 Kbps throughput

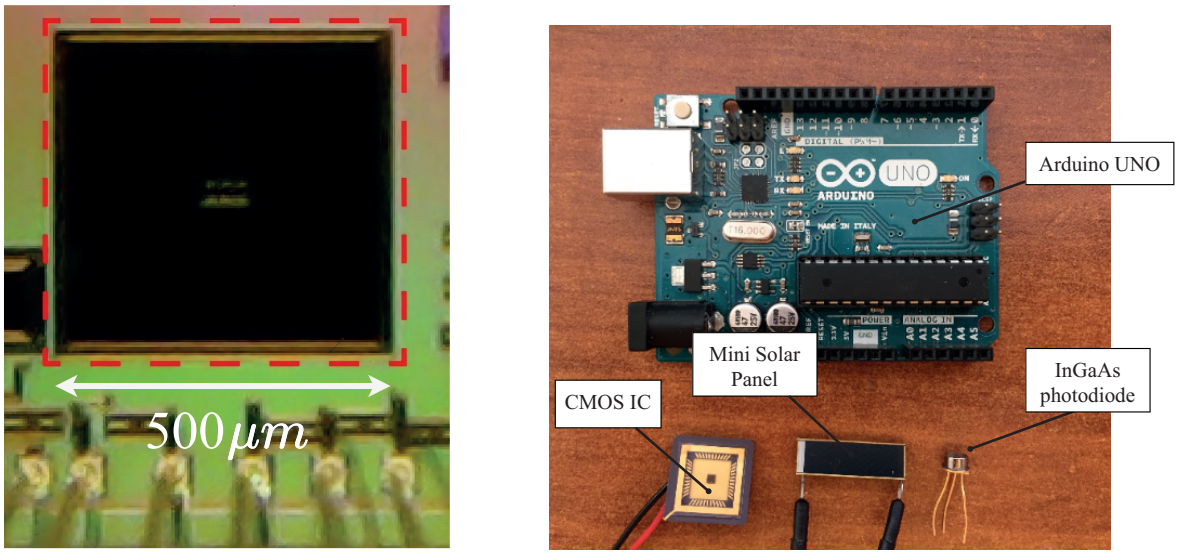
of the system, to generate 1 GB of random numbers, 18 full days are required which is impractical for most embedded system applications. However, for the target of this work, low data rate applications, the required data size is much lower. For instance, for a smart air monitoring system, the air quality index measurements are taken and transmitted in 5 to 15 min intervals where the data is 1 byte long [42–44]. Hence, for one year, the monitoring system only sends 34.2 KB to 102.6 KB of data. Therefore, for the protection of such low data rate communication, the proposed system is suitable for encryption processes.

Another limitation of the proposed method stems from its optical entropy source where changing optical conditions in the environment may lead to weaker random number generation. For example, a strong light source may saturate the optical source and the ADC, which results in less strong random number generation. However, this limitation may be overcome by placing the system in a controlled environment where the optical conditions are stable.

4. Experimental results and discussion

The algorithm was tested in various configurations with various optical sources with the very popular low-cost Arduino UNO Rev. 3 platform. The Arduino UNO was chosen for two main reasons (1) due to its wide availability and (2) for demonstrating the efficiency of the proposed method on a very low-cost platform. Figure 3b shows the test setup. The setup was placed in an environment that reflects the casual room environment that contains various electrical devices. The electrical devices were an HP Prodesk 405 4G desktop PC, Samsung SyncMaster 2233BW monitor. The tests were carried out at different locations in the same room. The room environment is specifically selected in this manner to underline the lack of need for special external noise sources and environments in the proposed method. In order to remove any positive bias towards the integrated CMOS photodiode in the tests and determine the contribution of the CMOS photodiode, various other entropy sources were tested with the algorithm in various lighting conditions. As the entropy sources (1) a CMOS integrated photodiode fabricated in 180 nm UMC technology in CLCC44 package shown in Figure 3a connected through 20 mm cables, (2) IXYS KXOB22-04X3F Mini solar panel (active area 1 cm^2) through 50 mm cables, (3) MTPD1346D InGaAs photodiode (package TO-46, spectral sensitivity 600 nm–1750 nm, active area 0.8 mm^2), (4) floating ADC pin, (5) floating ADC input pin with 20 cm cable as a monopole antenna to pickup EM noise, (6) floating ADC input pin with 20 cm cable and 20 cm cable connected to GND pin of Arduino UNO to act like a dipole antenna. To investigate the noise generated due to the IC package and cables, the following tests were added; (7) the ADC input pin connected through 20 mm cables to the cathode pin of the CLC44 package of CMOS IC, (8) GND of UNO connected to the cathode pin of the packaged CMOS IC via 20 mm cables. The A0 ADC channel of UNO was selected. The light intensity in the room was measured as 110 lux. In the consequent measurements, different light intensities were applied through a white power LED placed 1 mm above the CMOS photodiode to test extreme illumination effects on the algorithm. Various bitstreams containing up to 7,200,000 bits were generated, and from these bitstreams, 8-bit signed integers were formed. Figure 4a shows the generated 8-bit integers with their respective array index. Figure 4c shows the histogram constructed using the generated integers. Furthermore, to visually investigate the randomness, 2D plots were created from the bitstreams in each test using a custom Python script. Figure 5a shows the 2D plot of the algorithm used with the integrated CMOS photodiode in 110 lux room, Figure 5b shows the plot of the ADC output connected to the integrated CMOS photodiode in the same illumination, Figure 5c shows the plot of the ADC value with the floating ADC input pin, Figure 5d shows the algorithm with the floating ADC pin. Even without the statistical tests, it is visually evident that the proposed method results in a significant increase in

randomness even with a floating ADC pin.



(a) (b)

Figure 3. (a) Micrograph of the integrated CMOS photodiode manufactured in 180 nm standard CMOS technology, (b) measurement setup with the test components.

To compare the experimental data, the approximate entropy plots were also derived. The approximate entropy is a fast and reliable measure to test the irregularity and randomness of any time series [45]. The approximate entropy algorithm can be formulated as;

$$\phi_m(r) = \frac{1}{N-m+1} \sum_{i=1}^{N-m+1} C_i^m(r), \tag{3}$$

where N is the number of elements in the sequence, m is a nonnegative integer, r is a positive real number, C^m is a special term calculated using m, r and N. Then $ApEn$ value can be calculated as;

$$ApEn(m,r) = \lim_{N\to\infty} [\phi^m(r) - \phi^{m+1}(r)] = 0. \tag{4}$$

Figure 4b shows the ApEn result for the different number of bits of the single random number generation session. In these runs, m = 1 and r = 0.01 values are used. The test shows ApEn p-value never reaches less than 0.09 and it shows strong randomness for even 5000 bits. This demonstrates the feasibility of the algorithm for applications that require a short or a long stream of numbers.

The monobit test from the NIST-SP800-22r1a suite [46–48] is also used to measure the performance of the generator with various sizes of the generated bitstream. This test focuses on the proportion of zeroes and ones for the entire bitstream. The purpose of this test is to determine whether the number of ones and zeros in the bitstream are approximately the same. If the computed P-value in the test is greater than 0.01, then conclude that the sequence is random. All subsequent tests in the NIST suite are conditioned on having passed the monobit test. In this test, bits in the bitstream are converted $+1$ and -1 and summed together as S_n value. Then p-value is computed with $P = erfc(S_n/\sqrt{2n})$, where $erfc$ is the complementary error function

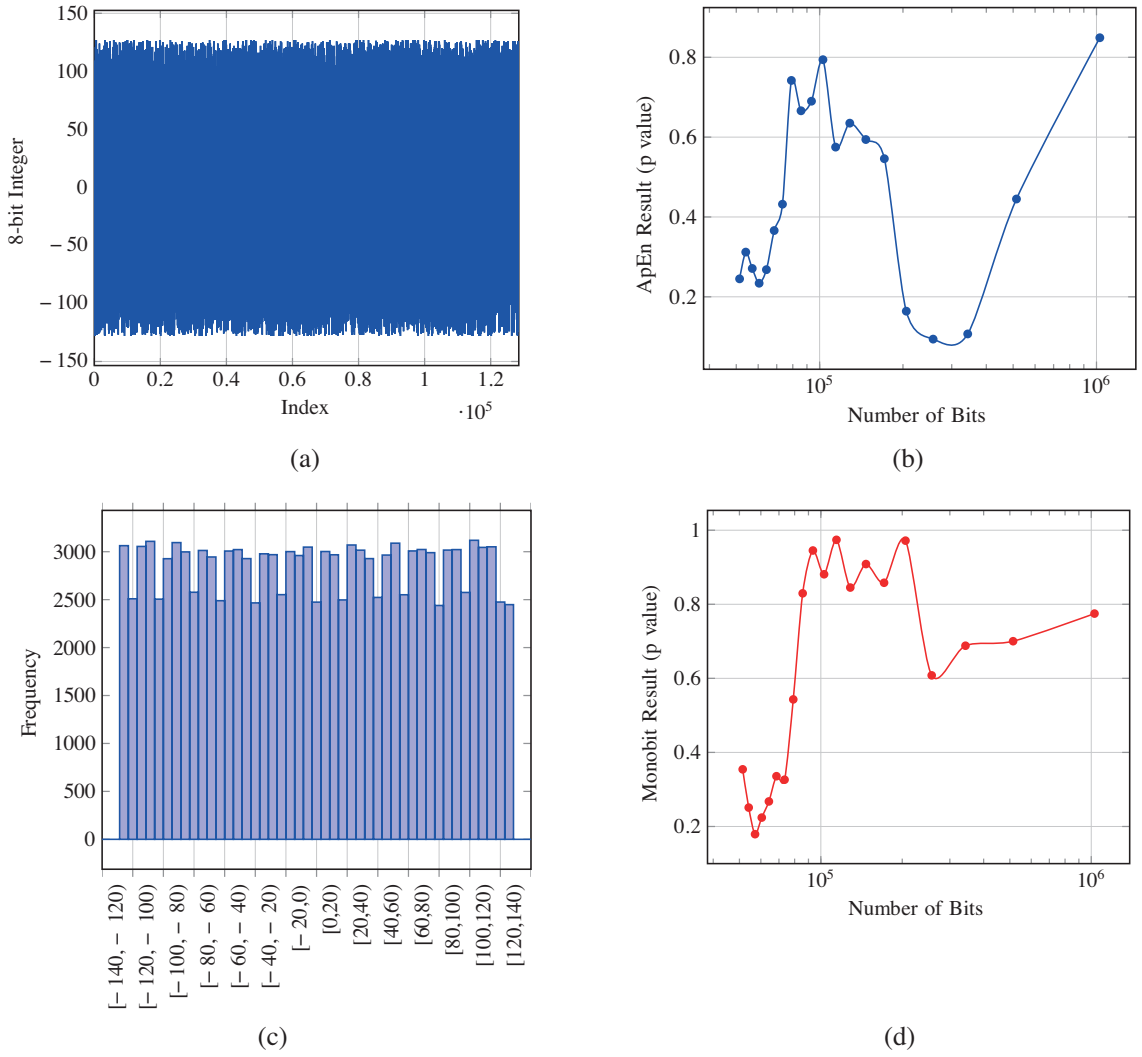


Figure 4. (a) Generated 128,510 8-bit signed integers (1,028,080 bits) ranging between (-128,127) using integrated CMOS photodiode and the proposed algorithm, (c) histogram of the same integers generated using the proposed method, (b) approximate entropy test result (m = 2, r = 0.01), (d) monobit test result for the same bitsream.

and n is number of bits in the bitstream. Figure 4d shows the obtained results from various sizes from 5000 to 1,028,080 bits. The results show that the proposed method passes the monobit test even for small-sized samples.

To test the strength of the generated bitstreams NIST-SP800-22 test suite [46–48] and AIS 31 tests [49] are applied. For NIST tests 1,020,080 bits and for AIS31 tests 7,200,000 bits are generated. Table shows the result obtained by these tests. For gathering data, a Catalex MicroSD Card adapter is connected to the Arduino UNO. 2GB LinkTech microSD card is connected to the adapter. The SD Card is chosen due to its bandwidth value that is significantly higher than the UART serial communication. In this manner, the impact of the data transfer during tests is minimized. All datasets are saved to the microSD card and tests are carried out with these datasets on a PC. For NIST tests Python implementation is carried out. For AIS31 tests, the official Java

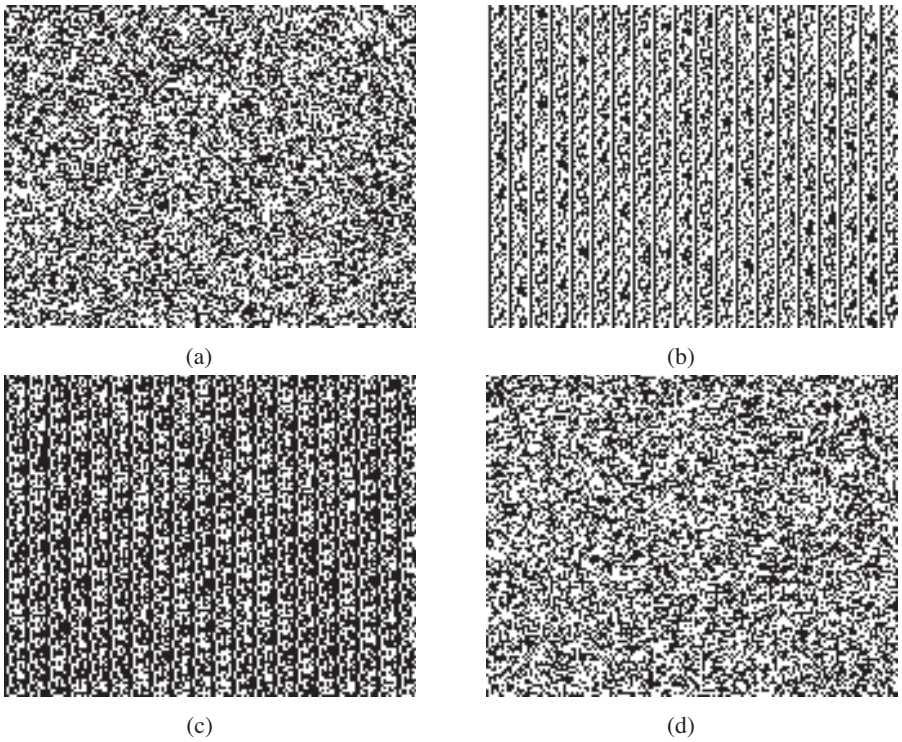


Figure 5. 2D plots constructed from (a) the algorithm with the integratd CMOS Photodiode in 110 lux, (b) ADC connected to the integrated CMOS photodiode in 110 lux, (c) floating ADC pin without algorithm, (d) floating ADC pin with the algorithm.

test suite provided in the BSI.de website is executed with parameters; data format 1Byte=8RNDBit and RNDBITBREITE=8 bit. All the configurations listed above are tested. The popular method of getting random seeds from floating pins does not pass the tests and fails on each occasion. Even with 20 cm cables attached to the floating ADC input pin and GND pin, a successful result was not attainable. In another test, values obtained directly from the ADC when integrated CMOS photodiode connected are gathered. This configuration also fails in the tests. However, when the proposed algorithm with the floating pin is utilized together, an improvement is observed where the DFT test in the NIST suite is passed. In a typical room lighting with 110 lux, when the optical sources listed above are used, the tests fail. When the integrated CMOS photodiode is used in the same environment, a strong result that is comparable with the random() function in stdlib is observed. The difference between the optical sources is determined to be mainly due to the internal parasitic capacitance differences. The capacitances of the mini solar panel and InGaAs photodiode are measured as 48 nF and InGaAs as 82 pF respectively in the same 110 lux environment. The relatively small capacitance of the CMOS photodiode is unable the filter our high-frequency noise. This theory was tested by adding a 100 nF capacitance between the terminals of the integrated CMOS photodiode, which expectedly results in failed NIST tests. The tests are repeated in four different illumination conditions; (1) applying over 100,000 lux with a 1 W white power LED placed 1 mm above the CMOS photodiode, (2) placing the system in daylight with approximately 1000 lux illumination, and (3) dark environment (approx. 0 lux), and (4) free running in various illuminations between 0 lux-1100 lux. With the extreme condition of the 1 mm proximity placement of the power LED, in the worst-case test results, the proposed generator is still able to pass Frequency Within Block, Longest Run Ones In A

Block, Discrete Fourier Transform, Non Overlapping Template Matching tests in the NIST suite. The system is tested also within a dark environment. Due to internal noise generated on the diode and peripheral connections, the proposed generator passed the NIST tests successfully. In order to measure the success of the proposed method, the maximum allowed rejection rate presented in [50] is calculated in three different light intensities of 0 lux, 100 lux, and 1000 lux. The maximum allowed rejection rate for a set of sequences, ρ, can be computed as;

$$\rho = s\left(\alpha + 3\sqrt{\frac{\alpha(1-\alpha)}{s}}\right) \tag{5}$$

where α is the significance level, s is the number of sequences used in the computation.

The total number of sequences is set to 126 with 1,028,080 bits generated in each sequence. The sequences are generated in 0 lux, 110 lux, and 1000 lux illumination levels (42 sequences for each illumination level). The number of rejections is found to be 3 for 0 lux (one sequence failed in DFT test with p = 0.0026, one sequence failed in random excursion test with p = 0.00029 and one sequence failed in random excursion variant test with p = 0.0063), 3 for 110 lux (for all rejections, only random excursion tests are failed with a minimum p = 0.00399), and 3 for 1000 lux (one sequence failed in cumulative sums test with p = 0.005, one sequence failed in random excursion variant test with p = 0.0059, one sequence failed in frequency within block test with p = 0.00248). The calculated significance levels are 1.5% for the listed illuminations. The free-running system in combined illuminations is calculated to have a 7.2% significance level. It is important to note that no postprocessing is applied in the tests and for each rejection, only one test failed. Furthermore, the obtained results are significantly higher than the common random number generation method of using floating ADC pin in microcontrollers, which results in a 100% rejection rate and 100% significance level. Higher optical illumination levels that saturate the CMOS photodiode results in rejection of all generated sequences with at least more than one test failure. It is important to note that Rabbit, Alphabit, and FIPS-140-2 batteries in TestU01 suite are also passed successfully with 1,028,080 random bits generated with the proposed system.

5. Conclusion

A method for generating strong random numbers for low-cost microcontrollers using an integrated CMOS photodiode is proposed. The proposed method utilizes an integrated CMOS photodiode in the photovoltaic mode as the entropy source. To generate the random numbers, analog values from the photodiode are captured with ADC serially, then the LSB of these numbers are extracted. In the final step, a bit is constructed by applying logical XOR on to the captured LSBs. The method is tested on an Arduino UNO board in a typical room environment. The proposed random number generator is implemented with an integrated CMOS photodiode manufactured in 180 nm standard CMOS technology. NIST-SP800-22r1a suite and AIS31 test suite are applied to the numbers generated. The proposed method passes the tests at high scores.

Table . NIST-SP800-22r1a and AIS 31 test score comparison for different configurations in the tests.

NIST-SP800-22r1a test scores with generated 1,028,080 bits				**AIS31 test scores with generated 7,200,000 bits**	
Test	**Floating ADC Pin (p value)**	**Random() function in stdlib (p value)**	**Proposed algorithm with integrated CMOS photodiode (no random seed)**	**Test**	**Proposed algorithm with integrated CMOS photodiode**
Monobit	FAIL	PASS (0.774)	PASS (0.821)	P1:T0 (Disjunktheitstest)	PASS
Frequency within block	FAIL	PASS (0.141)	PASS (0.908)	P2:T1 (Monobittest) 257 Runs	PASS
Runs	FAIL	PASS (0.356)	PASS (0.629)	P2:T2 (Pokertest) 257 Runs	PASS
Longest run ones in a block	FAIL	PASS (0.278)	PASS (0.595)	P2:T3 (Runtest) 257 Runs	PASS
Binary matrix rank	FAIL	PASS (0.082)	PASS (0.744)	P2:T4 (Long Runtest) 257 Runs	PASS
DFT	FAIL	PASS (0.085)	PASS (0.950)	P2:T5 (Autokorrelationstest) 257 Runs	PASS
Nonoverlapping template matching	PASS (0.349)	PASS (0.999)	PASS (0.999)	P2:T6a (Uniform distribution test)	PASS
Overlapping template matching	FAIL	PASS (0.432)	PASS (0.082)	P2:T6b (Uniform distribution test)	PASS
Maurers universal	FAIL	PASS (0.849)	PASS (0.700)	P2:T7a (Homogeneity test)	PASS
Serial	FAIL	PASS (0.849)	PASS (0.679)	P2:T7b (Homogeneity test)	PASS
Approximate entropy	FAIL	PASS (0.849)	PASS (0.954)	P2:T8 (Coron's test)	PASS
Cumulative sums	FAIL	PASS (0.531)	PASS (0.947)		
Random excursion	FAIL	PASS (0.053)	PASS (0.075)		
Random excursion variant	PASS (0.064)	PASS (0.065)	PASS (0.095)		
Linear complexity	PASS (0.981)	PASS (0.476)	PASS (0.386)		

Acknowledgment

This study was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) (EEAG 114E549).

References

[1] Von Neumann J. Various Techniques Used in Connection with Random Digits. National Bureau of Standards Applied Mathematics Series 1951; 12 (13): 36-38.

[2] Park SK, Miller KW. Random number generators: good ones are hard to find. Communications of the ACM 1988; 31 (10): 1192–1201. doi:10.1145/63039.63042

[3] Langdon WB. A fast high quality pseudo random number generator for graphics processing units. In: 2008 IEEE Congress on Evolutionary Computation; Hong Kong, China; 2008. pp. 459-465.

[4] Tseng PH, Lee MH, Lin YH, Lung HL, Wang KC et al. ReRAM-Based Pseudo-True Random Number Generator With High Throughput and Unpredictability Characteristics. IEEE Transactions on Electron Devices 2021; 68 (4): 1593-1597. doi: 10.1109/TED.2021.3057028

[5] Chen Z, Zhao B, Lin H, Chen L. Etoram: A More Efficient ORAM for Secure Computation. IEEE Open Journal of the Computer Society 2020; 1 (1): 285-294. doi: 10.1109/OJCS.2020.3032020

[6] Hu J, Zhang Z, Pan Q. A 15-Gb/s 0.0037-mm² 0.019-pJ/Bit Full-Rate Programmable Multi-Pattern Pseudo-Random Binary Sequence Generator. IEEE Transactions on Circuits and Systems II: Express Briefs 2020; 67 (9): 1499-1503. doi: 10.1109/TCSII.2020.3008567

[7] El-Latif AAA, Abd-El-Atty B, Venegas-Andraca SE, Elwahsh H, Piran J. Providing End-to-End Security Using Quantum Walks in IoT Networks. IEEE Access 2020; 8: 92687-92696.doi: 10.1109/ACCESS.2020.2992820

[8] Mathew SK, Srinivasan S, Anders MA, Kaul H, Hsu SK et al. 2.4 Gbps, 7 mW All-Digital PVT-Variation Tolerant True Random Number Generator for 45 nm CMOS High-Performance Microprocessors. IEEE Journal of Solid-State Circuits 2012; 47 (11): 2807-2821. doi: 10.1109/JSSC.2012.2217631

[9] Kim E, Lee M, Kim J. 8.2 8Mb/s 28Mb/mJ robust true-random-number generator in 65 nm CMOS based on differential ring oscillator with feedback resistors. In: 2017 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA; 2017. pp. 144-145.

[10] Yao Y, Chen X, Kang W, Zhang Y, Zhao W. Thermal Brownian Motion of Skyrmion for True Random Number Generation. IEEE Transactions on Electron Devices 2020; 67 (6): 2553-2558. doi: 10.1109/TED.2020.2989420

[11] Chandrasekaran ST, Karnam VEG, Sanyal A. 0.36-mW, 52-Mbps True Random Number Generator Based on a Stochastic Delta–Sigma Modulator. IEEE Solid-State Circuits Letters 2020; 3: 190-193. doi: 10.1109/LSSC.2020.3010901

[12] Wang X, Liang H, Wang Y, Yao L, Guo Y et al. High-Throughput Portable True Random Number Generator Based on Jitter-Latch Structure. IEEE Transactions on Circuits and Systems I: Regular Papers 2021; 68 (2): 741-750. doi: 10.1109/TCSI.2020.3037173

[13] Zhao Q, Zheng W, Zhao X, Cao Y, Zhang F et al. A 108 F2/Bit Fully Reconfigurable RRAM PUF Based on Truly Random Dynamic Entropy of Jitter Noise. IEEE Transactions on Circuits and Systems I: Regular Papers 2020; 67 (11): 3866-3879. doi: 10.1109/TCSI.2020.3008407

[14] Avaroğlu E, Tuncer T. A novel S-box-based postprocessing method for true random number generation. Turkish Journal of Electrical Engineering and Computer Science 2020; 28 (1): 288-301. doi: 10.3906/elk-1906-194

[15] Liu Y, Chen C, Yang DD, Li Q, Li X. Fast True Random Number Generator Based on Chaotic Oscillation in Self-Feedback Weakly Coupled Superlattices. IEEE Access 2020; 8: 182693-182703. doi: 10.1109/ACCESS.2020.3028735

[16] Matsumoto M, Nishimura T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transaction on Modelling and Computer Simulation 1998; 8 (1): 3–30. doi: 10.1145/272991.272995

[17] Kuka CS, Hu Y, Xu Q, Alkahtani M. An Innovative Near-Field Communication Security Based on the Chaos Generated by Memristive Circuits Adopted as Symmetrical Key. IEEE Access 2020; 8: 167975-167984. doi: 10.1109/ACCESS.2020.3023049

[18] Luo Y, Wang W, Best S, Wang Y, Xu X. A High-Performance and Secure TRNG Based on Chaotic Cellular Automata Topology. IEEE Transactions on Circuits and Systems I: Regular Papers 2020; 67 (12): 4970-4983. doi: 10.1109/TCSI.2020.3019030

[19] Ozkaynak F. A Novel Random Number Generator Based on Fractional Order Chaotic Chua System. Elektronika Ir Elektrotechnika 2020; 26 (1): 52-57. doi: 10.5755/j01.eie.26.1.25310

[20] Koyuncu İ, Şeker H, Alçın M, Tuna M. A Novel Dormand-Prince Based Hybrid Chaotic True Random Number Generator on FPGA. Balkan Journal of Electrical and Computer Engineering 2021; 9 (1):40-47. doi: 10.17694/bajece.722911

[21] Amirany A, Jafari K, Moaiyeri MH. True Random Number Generator for Reliable Hardware Security Modules Based on a Neuromorphic Variation-Tolerant Spintronic Structure. IEEE Transactions on Nanotechnology 2020; 19: 784-791. doi: 10.1109/TNANO.2020.3034818

[22] Milovančev D, Vokić N, Pacher C, Khan I, Marquardt C et al. Towards Integrating True Random Number Generation in Coherent Optical Transceivers. IEEE Journal of Selected Topics in Quantum Electronics 2020; 26 (5): 1-8. doi: 10.1109/JSTQE.2020.3004206

[23] Mikailov M, Sudarsan SD, Luo F. Pseudo Random Number Generation for Parallelized Jobs on Clusters. In: 2012 12th IEEE/ACM International Symposium on Cluster, Ottawa, ON, Canada; 2012. pp. 680-681.

[24] Huang L, Zhou H, Xie C. Quantum Random Number Generation on Alibaba Cloud Servers. 2020 IEEE Photonics Conference (IPC), Vancouver, BC, Canada; 2020, pp. 1-2.

[25] Zungeru AM. A Secured Smart Home Switching System based on Wireless Communications and Self-Energy Harvesting. IEEE Access 2019; 7: 25063-25085. doi: 10.1109/ACCESS.2019.2900305

[26] Friha O, Ferrag MA, Shu L, Maglaras L, Wang X. Internet of Things for the Future of Smart Agriculture: A Comprehensive Survey of Emerging Technologies. IEEE/CAA Journal of Automatica Sinica 2021; 8 (4): 718-752. doi: 10.1109/JAS.2021.1003925

[27] Herrewege AV, Verbauwhede I. Software Only, Extremely Compact, Keccak-based Secure PRNG on ARM Cortex-M. In: 51st Annual Design Automation Conference (DAC '14).New York, NY, USA; 2014, pp. 1–6.

[28] Huth C, Becker D, Merchan JG, Duplys P, Güneysu T. Securing Systems With Indispensable Entropy: LWE-Based Lossless Computational Fuzzy Extractor for the Internet of Things. IEEE Access 2017; 5: 11909-11926. doi: 10.1109/ACCESS.2017.2713835

[29] Wei W, Xie G, Dang A, Guo H. High-Speed and Bias-Free Optical Random Number Generator. IEEE Photonics Technology Letters 2012; 24 (6): 437-439. doi: 10.1109/LPT.2011.2180521

[30] Bisadi Z, Fontana G, Moser E, Pucker G, Pavesi L.Robust Quantum Random Number Generation With Silicon Nanocrystals Light Source. Journal of Lightwave Technology 2017; 35 (9): 1588-1594. doi: 10.1109/JLT.2017.2656866

[31] Wang A, Wang L, Wang Y. Post-processing-free 400 Gb/s true random number generation using optical heterodyne chaos. In: 2016 25th Wireless and Optical Communication Conference; Chengdu, China; 2016, pp. 1-4.

[32] Park BK, Park H, Kim Y, Kang J, Yeom Y et al. Practical True Random Number Generator Using CMOS Image Sensor Dark Noise. IEEE Access 2019; 7: 91407-91413. doi: 10.1109/ACCESS.2019.2926825

[33] AlMutairi D, Bonny T.Image Encryption Based on Chua Chaotic Oscillator. In: 3rd International Conference on Signal Processing and Information Security (ICSPIS); DUBAI, United Arab Emirates; 2020, pp. 1-4.

[34] Bakiri M, Guyeux C, Couchot JF, Oudjida AK. Survey on hardware implementation of random number generators on FPGA: Theory and experimental analyses. Computer Science Review 2018; 27: 135-153. doi:10.1016/j.cosrev.2018.01.002

[35] Bandyopadhyay D, Sen J. Internet of Things: Applications and challenges in technology and standardization, Wireless Personal Communications 2011; 58 (1): 49–69. doi: 10.1007/s11277-011-0288-5

[36] Daugman J. How iris recognition works. IEEE Transactions on Circuits and Systems for Video Technology 2004; 14 (1): 21-30. doi: 10.1109/TCSVT.2003.818350.

[37] Bennett CH, Brassard G. Quantum cryptography: Public key distribution and coin tossing. Processing IEEE International Conferance Computer Systems Signal Processing, Dec. 1984, pp. 175–179.

[38] Brouk I, Nemirovsky A, Nemirovsky Y. Analysis of noise in CMOS image sensor. In: 2008 IEEE International Conference on Microwaves, Tel-Aviv, Israel; 2008. pp. 1-8.

[39] Murari K, Etienne-Cummings R, Thakor N, Cauwenberghs G. Which Photodiode to Use: A Comparison of CMOS-Compatible Structures. IEEE Sensors Journal 2009;9 (7):752-760. doi:10.1109/JSEN.2009.2021805

[40] Chao CY, Chen Y, Chou K, Sze J, Hsueh F et al. Extraction and Estimation of Pinned Photodiode Capacitance in CMOS Image Sensors. IEEE Journal of the Electron Devices Society 2014; 2 (4): 59-64. doi: 10.1109/JEDS.2014.2318060

[41] Vigna S. Further scramblings of Marsaglia's xorshift generators. Journal of Computational and Applied Mathematics 2017; 315: 175-181. doi: 10.1016/j.cam.2016.11.006

[42] Migabo EM, Djouani KD, Kurien AM. The Narrowband Internet of Things (NB-IoT) Resources Management Performance State of Art, Challenges, and Opportunities.IEEE Access 2020; 8: 97658-97675. doi: 10.1109/ACCESS.2020.2995938

[43] Gany F, Bari S, Prasad L. Perception and reality of particulate matter exposure in New York City taxi drivers. Journal of Exposure Science Environmental Epidemiology 2017; 27: 221–226. doi: 10.1038/jes.2016.23

[44] Kök İ, Şimşek MU, Özdemir S. A deep learning model for air quality prediction in smart cities. In: IEEE International Conference on Big Data (Big Data), Boston, USA; 2007. pp. 1983-1990.

[45] Pincus SM. Approximate entropy as a measure of system complexity. Proceedings of the National Academy of Sciences of the United States of America 1991; 88: 2297–301. doi:10.1073/pnas.88.6.2297

[46] Bassham L, Rukhin A, Soto J, Nechvatal J, Smid M et al. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. USA: NIST, 2010.

[47] Pareschi F, Rovatti R, Setti G. On Statistical Tests for Randomness Included in the NIST SP800-22 Test Suite and Based on the Binomial Distribution. IEEE Transactions on Information Forensics and Security 2012; 7(2): 491-505. doi: 10.1109/TIFS.2012.2185227

[48] Goll M, Gueron S. Randomness Tests in Hostile Environments. IEEE Transactions on Dependable and Secure Computing 2018; 15(2): pp. 289-294. doi: 10.1109/TDSC.2016.2537799

[49] Killman W, Schindler W. A proposal for: Functionality classes for random number generators. Bundesamt für Sicherheit in der Informationstechnik (BSI), 2011

[50] Soto, Juan. (1999). Randomness Testing of the Advanced Encryption Standard Candidate Algorithms.