Enhancing Pseudorandom Number Generation using Environmental Sensor-based Entropy Sources*

Svitlana Poperehnyak[1,*,†], Ihor Syvachenko[2,†] and Yurii Shevchuk[3,†]

[1] National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute," 37 Beresteiskyi ave., 03056 Kyiv, Ukraine
[2] Institute of Software Systems of NAS of Ukraine, 40 Academician Glushkov ave., 03187 Kyiv, Ukraine
[3] DataArt, 3530 Carol Ln, Northbrook, 60062 Illinois, USA

Abstract

The article considers one of the methods of building a pseudorandom number generator (PSN) based on physical sensors that use natural fluctuations in the environment to obtain a source of entropy. The focus is on the analysis of different types of sensors, such as temperature, acoustic, light, gyroscopes, and magnetometers, which can provide a sufficient level of randomness to reliably generate pseudorandom numbers. Methods of digital processing of sensor signals are described, including noise filtering, quantization, and data binarization to obtain a sequence of random bits. In addition, the quality of the obtained pseudorandom numbers is assessed using statistical and cryptographic tests. Hardware requirements and possible scenarios for the use of such generators in cryptography, the Internet of Things (IoT), the gaming industry, and other cyber-physical systems are highlighted. The article aims to develop a reliable method for generating high-quality random numbers, which is based on the unpredictability of physical processes. The article discusses an approach to generating a pseudorandom sequence using sensors. It highlights the factors contributing to the randomness of devices based on environmental sensors, which can be utilized in the basic configuration of a slow monostable timer. The method described enables the sensor to generate a random factor without requiring any user input. A study was conducted using a robot, and suggestions for improving the operation of a linear feedback shift register (LFSR) are provided. The concept of an LFSR can also be applied in programming languages to produce sequences of pseudorandom numbers.

Keywords

random, sensors, pseudorandom sequence, shift register, source of entropy, digital signal processing, cryptography, Internet of Things, randomness, hardware generators, information protection

1. Introduction

In today's world, pseudorandom numbers play a key role in many fields, including cryptography, computer modeling, statistical methods, and the gaming industry. However, traditional algorithmic pseudorandom number generators have certain limitations, including repeatability and predictability, which can be a serious problem for applications that require high levels of security and entropy [1, 2].

Because of this, there is an urgent need to create more reliable sources of entropy. One promising approach is to use sensors to generate pseudo-random numbers that can exploit the natural random fluctuations of physical processes [3, 4]. Sensors that measure temperature, noise, motion, lighting, and other environmental parameters provide a high level of entropy, which makes them attractive for building hardware pseudorandom number generators.

* CPITS 2025: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2025, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
spopereshnyak@gmail.com (S. Popereshnyak); igor.syvachenko@gmail.com (I. Syvachenko); shev4ukyuri@gmail.com (Y. Shevchuk)
0000-0002-0531-9809 (S. Popereshnyak); 0009-0005-3248-3371 (I. Syvachenko); 0009-0008-3331-3886 (Y. Shevchuk)

The article, devoted to one of the methods of constructing a generator of pseudo-random numbers based on sensors, is relevant for several reasons:

1. Growing need for security. The requirements for cryptographic security in networks and information systems are growing every year.
2. Problems of predictability of traditional algorithms. Algorithmic generators have limited entropy, which makes them vulnerable to attacks. The physical processes used by sensors have a much greater potential for unpredictability.
3. Development of hardware solutions. As hardware advances, particularly embedded systems and the IoT, there is a growing need for compact and reliable pseudorandom number generators that can be integrated into resource-constrained devices.
4. Innovations in scientific and technical progress. The use of sensors to generate pseudorandom numbers opens up new opportunities for scientific research and the development of new technological solutions in various fields, such as telecommunications, automation, computer science, and others.

Therefore, this article contributes to the development of methods for generating reliable pseudo-random numbers based on hardware, offering an innovative approach to solving the problem of predictability and repeatability of numbers generated by traditional algorithms [5, 6].

Setting the problem. The generation of pseudo-random numbers is an important component of many modern systems, including cryptography, computer modeling, the gaming industry, and scientific research. Traditional algorithmic pseudo-random number generators have certain limitations related to their predictability and low entropy level, which can be critical in cases where high cryptographic security is required [7–9]. The question arises of finding alternative approaches that would increase the unpredictability and randomness of numbers. One such promising method is the use of sensors to generate pseudo-random numbers based on physical processes. However, the problem of developing a mathematical model and technical solutions for the effective implementation of this approach remains open.

The purpose of the article is to develop one of the ways to build a pseudorandom number generator based on sensors that use random physical phenomena to ensure a high level of entropy and unpredictability. The article deals with methods of measuring physical processes, analysis of their randomness, mathematical processing of data to obtain bits of a random sequence, and evaluation of the quality of the received pseudorandom numbers.

The task of the article:

1. Analysis of entropy sources available through sensors. An overview of the possible types of sensors (temperature, acoustic, light, etc.) that can be used to generate random signals.
2. Development of a mathematical model for the generation of pseudorandom numbers based on sensor measurements. Study of the processes of data normalization, quantization, and signal processing to convert physical measurements into a sequence of bits.
3. Implementation of digital data processing to ensure randomness. Definition of noise filtering methods, reduction of influence of deterministic components, and construction of process of obtaining bits.
4. Evaluation of the quality of the received pseudorandom numbers. Conducting randomness tests, such as cryptographic and statistical tests, to verify the reliability and unpredictability of the generator.
5. Development of recommendations for the practical implementation of a sensor-based generator. Discussion of hardware requirements and potential scenarios for the use of such generators in various industries.

So, the article will outline the concept of building a generator of pseudo-random numbers based on physical phenomena measured by sensors, using methods of mathematical processing and evaluation of the randomness of the received numbers.

2. Optimization of information protection means integrating sensory sources of entropy

Optimization of the composition of information protection tools in wartime conditions is one of the key elements of the research, which is focused on improving the quality of pseudo-random number generation using entropy sources based on environmental sensors.

In wartime conditions, when traditional means of ensuring information security may be ineffective due to resource constraints and reduced infrastructure reliability, there is a need to develop flexible, adaptive protection systems capable of operating in conditions of unstable power supply, limited access to Internet resources, and other logistical constraints. In this case, the optimization of information protection tools involves a strategic approach to the selection of methods and technologies that allow for ensuring the maximum level of security at minimal cost.

A key aspect in such optimization is the effective use of cryptography technologies, in particular the generation of pseudo-random numbers, which underlie many encryption algorithms. Traditional methods of generating pseudorandom numbers using standard hardware and software can be vulnerable to attacks, especially in wartime conditions, when infrastructure can be destroyed or sabotaged.

The use of entropy sources based on environmental sensors, such as temperature sensors, humidity sensors, or wind speed sensors, can significantly improve the quality of random number generation, as such sources provide high unpredictability and complexity. Given that most modern cryptographic algorithms require stable random number generation for encryption and authentication, the integration of such technologies requires careful optimization in terms of cost and efficiency.

Therefore, optimization of the composition of information protection tools in wartime conditions can be achieved through the integration of sensor technologies for random number generation, which allows reducing the cost of traditional hardware while maintaining high protection efficiency. Taking into account the specifics of wartime conditions, such as limited resources and high risk of attacks, is important to achieve a balance between the cost and effectiveness of the developed solutions, which guarantees the resistance of information systems to malicious influences.

3. Analysis of entropy sources available through sensors

Entropy in the context of pseudorandom number generation refers to the amount of information or randomness that can be extracted from a physical process. Sensors can measure various physical quantities, which by their nature contain a significant level of randomness. These values can be used as sources of entropy for generating random signals [10–12]. Below is an overview of the most common sensor types that can be used for this task.

Table 1
Overview of the most common sensor types

Type	Description	Source of entropy	Advantages	Disadvantages
Temperature sensors	Temperature sensors measure the temperature of the environment or certain components. Although temperature changes are often gradual, at the micro level temperature can fluctuate due to external factors that are random.	**Thermal fluctuations**: Temperature changes caused by both internal processes (for example, device operation) and external conditions (atmospheric effects) can have a random component. **Thermal noise** (Johnson noise): It occurs due to random movements of charged particles in conductors. This type of noise can be used to generate random bits.	Ease of use. Durability and stability in various environments.	Low frequency of fluctuations in macroscopic conditions. Additional processing is required to obtain stable random bits.
Acoustic sensors (microphones)	Acoustic sensors are used to measure sound vibrations in air or liquids. Sound waves contain a significant level of noise that can be used as a source of entropy.	**Acoustic noise**: Sound fluctuations are always present in the environment, which are difficult to predict. For example, background noise may include wind noise, people's movements, the operation of devices. **Signal Coherence**: Small random fluctuations can be detected even in a relatively quiet environment.	Constant access to unpredictable fluctuations. High frequency of signal changes.	Possible stability issues in very quiet environments. Dependence on the level of ambient noise.
Magnetic sensors (magnetometers)	Magnetometers measure changes in the magnetic field around the device. Even in the absence of visible changes in the magnetic field, random fluctuations caused by microfluctuations in the environment can occur.	**Changes in the magnetic field**: Random changes in the magnetic field can be caused by the movement of magnetic objects, electrical currents, or geophysical factors. **Quantum fluctuations**: Under certain conditions, magnetometers can pick up quantum fluctuations in magnetic fields.	No need for direct physical contact or changing the position of the sensor. Constant presence of fluctuations in the magnetic field.	Sensitivity to electromagnetic interference. Possible stability of the magnetic field under controlled conditions.

Light sensors (photodiodes, phototransistors)	Light sensors are used to measure the level of illumination. Random fluctuations in light intensity, even in a stable environment, can be a source of entropy.	**Quantum Fluctuations in Light Intensity**: The light perceived by the sensor has random quantum fluctuations due to the nature of photons, especially in low-intensity conditions (darkness or very low light). **External factors:** Changes in illumination due to natural factors, such as cloud movement, changes in the angle of the sun's rays, or other random processes that change the level of illumination.	High frequency of fluctuations in variable environments. Availability of photodiodes and other light sensors.	In a controlled environment with constant lighting, entropy can be low. Need to have a variable light source to get enough random changes.
Magnetic sensors (magnetometers)	Magnetometers measure changes in the magnetic field around the device. Even in the absence of visible changes in the magnetic field, random fluctuations caused by microfluctuations in the environment can occur.	**Changes in the magnetic field**: Random changes in the magnetic field can be caused by the movement of magnetic objects, electrical currents, or geophysical factors. **Quantum fluctuations**: Under certain conditions, magnetometers can pick up quantum fluctuations in magnetic fields.	No need for direct physical contact or changing the position of the sensor. The constant presence of fluctuations in the magnetic field.	Sensitivity to electromagnetic interference. Possible stability of the magnetic field under controlled conditions.
Gyroscopes and accelerometers	These sensors are used to measure the orientation and movement of the device. They are very sensitive to the slightest changes in movement, even if the device is in relative rest.	**Microfluctuations of movement**: Gyroscopes and accelerometers can register small movements or vibrations caused by random processes, for example, micro-vibrations of the environment or vibrations of internal components. **Unpredictable vibrations**: These sensors can detect micro-movements that are invisible to the eye, caused by, for example, changes in position or random vibrations.	High sensitivity to fluctuations. Quick response to changes.	In stationary systems where there is no motion, entropy can be limited. May require a high polling rate to efficiently use entropy.

Electrochemical sensors	These sensors measure chemical changes in the environment, such as humidity levels or the concentration of certain chemicals.	**Random chemical reactions**: In many environments, even under stable conditions, random changes in chemical composition or concentration can occur. **Fluctuations in humidity or gases**: Random changes in air composition or humidity can affect the performance of electrochemical sensors.	High sensitivity to changes in the chemical environment. Use in special environments (for example, in chemical laboratories or controlled atmosphere environments)	Slow reaction in stable conditions. Dependence on external factors.

Various types of sensors can be used as sources of entropy to generate random numbers, but their effectiveness depends on the nature of the measured physical processes and the environment in which they operate. Depending on the target system, one or more sensors can be used to improve the randomness of the generator:

- Temperature and light sensors are suitable for systems with slow environmental changes.
- Acoustic sensors and gyroscopes are suitable for environments with high dynamics or movement.
- Magnetic sensors and electrochemical sensors can be used in specific environments or to measure special phenomena.

The variety of sensors available allows for flexible customization of pseudorandom number generators to specific conditions or system requirements.

4. Random and environmental sensors

One approach to generating random numbers involves stopping a high-frequency timer at an arbitrary moment. Some projects require user interaction to stop the timer. However, let's consider a method in which a sensor can generate a random factor without any user involvement. In our experiment, we utilize a thermistor. To incorporate it into the circuit, simply replace the resistor in the original timer setup with the thermistor.

Random factors influencing the operation of the timers include not only changes in the thermistor's resistance but also additional variables that may affect performance:

- During operation, timers can generate slight heat, which may affect their performance.
- The repeat button might behave differently with each press.
- The circuit's power supply could experience minor fluctuations in current and/or voltage.
- The connections between the breadboard slots, jumpers, and component terminals have some resistance, which may change when jumpers or components are tapped.
- Other external factors, which we may not be aware of, could also influence the circuit's operation.

Automation of arbitrary selection scheme

The testing process can be accelerated by eliminating user input. The first step in this direction is to change the operation mode of the first timer from monostable to self-oscillating, with a period of one second. This adjustment allows for observing the results without needing to repeatedly press the repeat button.

In the second step, the counter reset button can be removed, and the circuit modified to perform the reset automatically. In some microcircuits, the counter is reset by a positive edge of the signal supplied to its reset input. In our setup, the output level transitions from low to high at the start of each cycle. This signal can be used to reset the timer through a coupling capacitor, ensuring that the high level is only briefly present on the reset input.

With these changes, the system should now operate independently without any user intervention.

Timer frequency setup

If the results from the automatic generation circuit for arbitrary numbers lack sufficient variability, the speed (i.e., frequency) of the second timer should be increased. To raise the frequency of the second timer to 500 Hz, it is recommended to replace the 1 μF timing capacitor with a 0.2 μF capacitor. For a frequency of 5 000 Hz, a capacitor with a capacitance of 0.01 μF should be used.

The faster the second timer operates, the higher the likelihood that the states recorded after stopping will differ due to small variations in the timing.

5. Implementation of digital data processing to ensure randomness

To implement digital data processing to generate pseudo-random numbers based on sensor indicators, it is necessary to ensure (Fig. 1):

1. Noise filtering.
2. Reducing the influence of deterministic components.
3. Construction of the quantization process for obtaining bits.

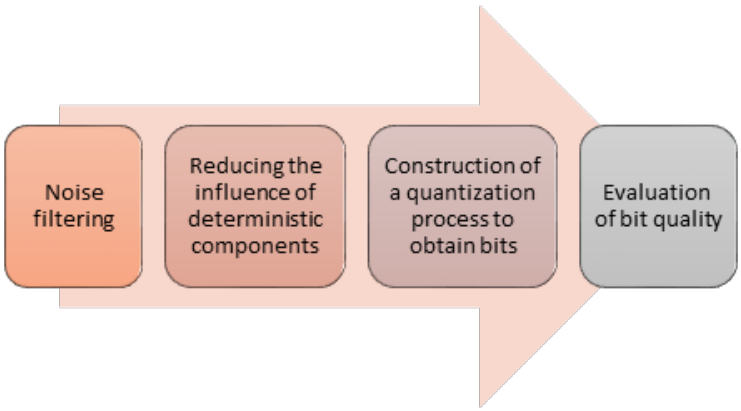

Figure 1: Stages of data processing

Let's consider the main stages of data processing.

5.1. Noise filtering

Real-world sensors measure signals consisting of both useful signals and random noise. Filtering is used to extract the random component of the signal. One of the most common approaches to noise filtering is the use of high-pass filters (such as low-pass filters) that filter out long-term fluctuations (the deterministic component).

Noise filtering algorithm:

- **Median filter**: used to reduce peak noise in signals. It works by calculating the median of a certain number of values in a sliding window of size *n*.

Suppose we have a signal $x(t)$ read from a sensor:

$$x_{filtered}(t) = median\left(x(t-n), \ldots, x(t+n)\right). \quad (1)$$

This method allows you to reduce the impact of abnormal values or outliers while preserving the underlying fluctuations.

- **Moving average:** Another way to smooth a signal is to average it over a moving window of size w:

$$x_{smooth}(t) = \frac{1}{w} \sum_{i=t-w/2}^{t+w/2} x(i). \quad (2)$$

Such filtering allows you to smooth the signal, reducing the impact of short-term fluctuations.

5.2. Reducing the influence of deterministic components

To isolate the truly random part of the signal, it is necessary to reduce or eliminate the deterministic components. This can be done in several ways:

- **Average subtraction**: If the deterministic component has a stable trend, it can be eliminated by subtracting the average value of the signal:

$$x_{detrended}(t) = x(t) - \bar{x}, \quad (3)$$

where $\bar{x}$ is the average value of the signal over a certain period

- **Fourier analysis**: Fourier transform can be applied to detect regular components of a signal that repeat with a certain frequency. After determining the main harmonics (deterministic frequencies), they can be removed from the spectrum, leaving only high-frequency fluctuations (random noise).

Fourier transform for signal $x(t)$:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-2\pi i f t} dt. \quad (4)$$

After the conversion, the signal can be processed by removing the low-frequency harmonics corresponding to the regular oscillations and performing the inverse conversion to obtain a pure random signal.

5.3. Bit acquisition process

After processing the signal, it is necessary to convert it into a sequence of bits that will be used to generate random numbers.

Quantization algorithm:

- **Signal normalization**: First, the signal is normalized to the range from 0 to 1. Suppose that the signal values are in the range from x_{min} to x_{max}:

$$x_{norm}(t) = \frac{x(t) - x_{min}}{x_{max} - x_{min}} \quad (5)$$

- **Signal binarization**: After normalization, the signal can be quantized to obtain bits. For this, you can use a threshold value, for example 0.5:

$$b(t)=\begin{cases}1, if\ x_{norm}(t)\geq 0.5\\ 0, if\ x_{norm}(t)<0.5\end{cases}. \quad (6)$$

The resulting bits can be sequenced and used as random numbers. To construct multi-bit numbers, several bits can be combined into groups of k bits:

$$R=\sum_{i=0}^{k-1} b(t+1)\cdot 2^i. \quad (7)$$

This will allow you to get a pseudorandom number in the range from 0 to 2^k-1.

5.4. Evaluation of the quality of bits

After generating the bits, it is important to check them for randomness using the following methods:

- Fries test (to analyze the uniformity of distribution).
- Series test (to check the absence of regular patterns in the sequence).
- Test for the length of blocks of zeros and ones (to check whether there are no long sequences of the same bits).

Implementation of digital data processing to generate random numbers from sensor signals includes several stages:

- Noise filtering to remove deterministic components.
- Reducing the influence of regular components by subtracting the average or using Fourier analysis.
- Normalization and binarization of the signal to obtain a sequence of random bits.
- Checking the quality of bits using randomness tests.

These methods make it possible to obtain pseudo-random numbers using the physical properties of sensor signals, which significantly increases their unpredictability compared to traditional algorithmic generators.

5.5. General mathematical model

Let $S(t)$ be the signal from the sensor at the time t, then the PRNG model can be described by the equation:

$$R(t)=H\left(Q\left(\frac{S(t)-min(S)}{max(S)-min(S)}\right)\right), \quad (8)$$

where H is a hashing or post-processing function; $Q(x)$ is a quantization function that converts values into binary bits.

Advantages:

- Number generation is based on physical processes, so it is difficult to predict the obtained results.
- The possibility of using different types of sensors to increase entropy.

Disadvantages:

1. Measurements may be sensitive to external conditions or equipment calibration.
2. Additional processing is required to obtain high-quality pseudo-random numbers.

Thus, building a sensor-based pseudorandom number generator requires combining physical measurements with mathematical signal processing to obtain unpredictable values.

6. Empirical problems

Empirical research involves deriving results from observation or experience. It is generally assumed that sensors like thermistors, humidity sensors, accelerometers, or pressure sensors should perform consistently across different users, producing a wide range of random values. However, this assumption cannot be fully trusted. In reality, researchers have long been interested in whether it is possible to design a system that can generate an unpredictable sequence of numbers completely independently, without being influenced by any external factors. It is crucial to demonstrate and mathematically prove that a given sequence of numbers remains consistent with each activation of the system.

Another key characteristic that indicates the quality of the generated sequence is the inability to predict the next number until the sequence starts to repeat. This would resemble an ideal pseudo-random number generator, as long as the sequence does not always start from the same point. Despite these challenges, let's attempt to implement such a device. Whether or not you need it depends on how you intend to use it.

6.1. Linear feedback shift register

Imagine we have a black box containing a mechanism that generates a stream of numbers without any external influence. It is crucial to determine whether these numbers are truly random.

To achieve this, the stream must meet two requirements:

- The sequence of numbers should be relatively unpredictable. The term "relatively" is used because any autonomous random number generator will eventually repeat its cycle if it runs for a sufficiently long period. The goal of such a generator is to produce a sequence that is long or complex enough to exceed human memory capacity or attention span. Ideally, the generator would have a large enough physical size to avoid being affected by quantum effects.
- The range of numerical values should be uniformly distributed, meaning each value should have an equal probability of appearing in the sequence, with none being omitted.

There is a design that can nearly satisfy both of these requirements: a linear feedback shift register (LFSR). The output sequence of an LFSR can be (almost) any length, and its values are (almost) perfectly balanced. Let's construct an LFSR such that these "almost" limitations are minimal enough to be negligible.

To summarize what we've learned about the linear feedback shift register:

- If the LFSR's memory cells start with all low states (e.g., 0000), they will continue to hold these states, simply cycling through them during further operation.
- If the LFSR's memory cells start with any value other than 0000, the register will cycle through fourteen other distinct combinations before the sequence repeats. The output sequence will include all values from 0001 to 1111, though not in a specific order. Every value will appear exactly once (except 0000), and no value will repeat until the entire sequence is completed.

However, the issue is that this sequence is short enough for the human brain to quickly recognize its repetition pattern.

6.2. Finding a solution

6.2.1. The problem with zeros

To address the issue of the LFSR's apparent non-responsiveness when all its memory cells are set to a low state upon activation, various recommendations typically suggest preloading the register with different values.

This can be done by modifying the circuit to include a module that provides clock pulses at a high level for a short duration at the register's input. However, there is a simpler solution: replacing the XOR logic element in the circuit with an XNOR element. Although XNOR elements are not commonly used in microcontrollers, they can effectively solve the problem.

6.2.2. Ensuring non-repeatability

Before assembling a test circuit using a four-element chip with two-input XNOR elements, it is important to revisit the issue of sequence repetition. Specifically, it is crucial to ensure that more than fifteen value combinations are generated before the sequence starts to repeat.

By utilizing all eight memory locations of the shift register, the output range can be expanded from 00000000 to 11111110, allowing the sequence to reach 255 combinations before repeating.

6.2.3. Features of the microcircuit XNOR

It is crucial to exercise particular caution when connecting the XNOR chip to avoid incorrect connections. The internal connections of this microcircuit are entirely distinct from those of other logic microcircuits. If this microcircuit is mistakenly connected as an OR or XOR circuit, it may sustain irreparable damage.

6.3. Conducting research

To obtain results from the study that are consistent with those in the article, the initial state of the shift register needs to be identical. Specifically, at the start of the cycle, all cells in the examined shift register must be set to a low state.

At this point, the LFSR will begin generating a random sequence, where a 0 indicates that the LED is off and a 1 indicates that it is on. Each time the button is pressed, the subsequent sequence in the table corresponds to the state of the LEDs in the circuit (Table 2).

Table 2
Sequence of 255 combinations of eight-bit linear feedback shift register, plus initial state repeat

Sequence of 255 combinations							
00000000	01110001	11001000	01001111	01111001	01011011	10000100	11100010
00000001	11100011	10010001	10011110	11110010	10110111	00001000	11000101
00000011	11000111	00100011	00111100	11100100	01101110	00010000	10001010
00000111	10001110	01000110	01111000	11001001	11011101	00100000	00010101
00001111	00011101	10001101	11110000	10010011	10111010	01000000	00101010
00011110	00111011	00011011	11100000	00100111	01110101	10000001	01010101
00111101	01110110	00110111	11000001	01001110	11101011	00000010	10101010
01111010	11101101	01101111	10000010	10011100	11010110	00000101	01010100

11110100	11011010	11011111	00000100	00111000	10101101	00001011	10101000
11101000	10110100	10111110	00001001	01110000	01011010	00010110	01010000
11010000	01101000	01111101	00010010	11100001	10110101	00101100	10100000
10100001	11010001	11111010	00100100	11000011	01101010	01011001	01000001
01000011	10100011	11110101	10001000	10000110	11010101	10110011	10000011
10000111	01000111	11101010	10010000	00001100	10101011	01100110	00000110
00001110	10001111	11010100	00100001	00011000	01010110	11001100	00001101
00011100	00011111	10101001	01000010	00110001	10101100	10011001	00011010
00111001	00111111	01010010	10000101	01100011	01011000	00110010	00110101
01110010	01111110	10100100	00001010	11000110	10110001	01100101	01101011
11100101	11111100	01001001	00010100	10001100	01100010	11001010	11010111
11001011	11111001	10010010	00101000	00011001	11000100	10010101	10101111
10010111	11110011	00100101	01010001	00110011	10001000	00101011	01011110
00101111	11100110	01001010	10100010	01100111	00010001	01010111	10111101
01011111	11001101	10010100	01000101	11001110	00100010	10101110	01111011
10111111	10011011	00101001	10001011	10011101	01000100	01011100	11110110
01111111	00110110	01010011	00010111	00111010	10001001	10111001	11101100
11111110	01101101	10100110	00101110	01110100	00010011	01110011	11011000
11111101	11011011	01001101	01011101	11101001	00100110	11100111	10110000
11111011	10110110	10011010	10111011	11010010	01001100	11001111	01100000
11110111	01101100	00110100	01110111	10100101	10011000	00111110	11000000
11101110	11011001	01101001	11101111	01001011	00110000	01111100	10000000
11011100	10110010	11010011	11011110	10010110	01100001	11111000	00000000
10111000	01100100	10100111	10111100	00101101	11000010	11110001	

Subsequently, a computer program was developed to emulate a linear feedback shift register and record its output.

In the following step, the binary numbers were converted into a decimal format for easier comprehension by the human brain. This resulted in the same sequence being represented in decimal format.

0, 1, 3, 7, 15, 30, 61, 122, 244, 232, 208, 161, 67, 135, 14, 28, 57, 114, 229, 203, 151, 47, 95, 191, 127, 254, 253, 251, 247, 238, 220, 184, 113, 227, 199, 142, 29, 59, 118, 237, 218, 180, 104, 209, 163, 71, 143, 31, 63, 126, 252, 249, 243, 230, 205, 155, 54, 109, 219, 182, 108, 217, 178, 100, 200, 145, 35, 70, 141, 27, 55, 111, 223, 190, 125, 250, 245, 234, 212, 169, 82, 164, 73, 146, 37, 74, 148, 41, 83, 166, 77, 154, 52, 105, 211, 167, 79, 158, 60, 120, 240, 224, 193, 130, 4, 9, 18, 36, 72, 144, 33, 66, 133, 10, 20, 40, 81, 162, 69, 139, 23, 46, 93, 187, 119, 239, 222, 188, 121, 242, 228, 201, 147, 39, 78, 156, 56, 112, 225, 195, 134, 12, 24, 49, 99, 198, 140, 25, 51, 103, 206, 157, 58, 116, 233, 210, 165, 75, 150, 45, 91, 183, 110, 221, 186, 117, 235, 214, 173, 90, 181, 106, 213, 171, 86, 172, 88, 177, 98, 196, 136, 17, 34, 68, 137, 19, 38, 76, 152, 48, 97, 194, 132, 8, 16, 32, 64, 129, 2, 5, 11, 22, 44, 89, 179, 102, 204, 153, 50, 101, 202, 149, 43, 87, 174, 92, 185, 115, 231, 207, 62, 124, 248, 241, 226, 197, 138, 21, 42, 85, 170, 84, 168, 80, 160, 65, 131, 6, 13, 26, 53, 107, 215, 175, 94, 189, 123, 246, 236, 216, 176, 96, 192, 128, 0.

At first glance, this sequence appears to be quite pseudo-random. A computer program was utilized to verify that each value occurs only once, without any gaps or repetitions.

6.3.1. Ones and zeros

After confirming that the logic circuit is functioning correctly, the next step is to determine how to modify this sequence so that only 1 or 0 is produced in each cycle.

One method to achieve this output is by applying the XOR operation to the outputs of the shift register. This approach utilizes a three-level "tree" of Exclusive OR gates, where each gate averages the outputs of the previous level to yield a final 0 or 1.

All eight memory locations remain in use during the feedback process. The eighth, sixth, fifth, and fourth cells continue to be XNORed as before, allowing us to maintain a non-repeating sequence of 255 values. It is recommended to take a subsample from this sequence, which will also undergo 255 combinations before starting to repeat. Consequently, the computer program was adjusted to extract only the rightmost bit from each of the 255 combinations generated by the linear feedback process. As a result, the following sequence of 1 and 0 was obtained:

0 1 1 1 1 0 1 0 0 0 0 1 1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 0 1 1 0 1 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 1
1 0 1 1 0 0 1 0 0 0 1 1 0 1 1 1 1 1 0 1 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0
1 0 0 0 1 0 1 1 1 0 1 1 1 1 0 0 1 0 0 1 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 1 0 1 0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 1 0 1
0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 1 0
1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 0 0 0 0 0 0

If we conduct this experiment on a physical circuit and only observe the rightmost LED, the same sequence will be produced.

6.3.2. The weighting factors problem

Analyzing the frequencies of the repeating subsequences of 1s and 0s yields the following results, which are presented in Table 3.

Table 3
Analysis of the frequencies of repeating subsequences 1 and 0

Value 0		Value 1	
0	33 times	1	32 times
00	16 times	11	16 times
000	8 times	111	8 times
0000	4 times	1111	4 times
00000	2 times	11111	2 times
000000	1 time	111111	1 time
0000000	1 time	1111111	1 time
Total	128 values 0	Total	127 values 1

In the presented sequence list (Table 2), there is one anomaly that appears to be incorrect—it contains 33 single zeros but only 32 single ones. The issue arises as follows: theoretically, the range of all values in the sequence should extend from 00000000 to 11111111. However, an XNOR gate linear feedback shift register cannot process the value 11111111, so it skips it. Since this value ends with a 1, it does not appear in our table of values that end in 1.

6.3.3. Skipping the 254th value

One solution to this problem would be to add additional shift registers. For instance, by connecting a chain of four registers that together have 32 memory cells, the entire non-repeating sequence of such an LFSR would consist of over four trillion values. However, this sequence would also include all combinations of 32 ones and zeros, except for the last value, which consists of 32 ones. This single value would now be obscured among more than four trillion other values, which should be acceptable for nearly any application.

Unfortunately, implementing this scheme requires additional work and components. Therefore, let's consider a simpler alternative. We can simply instruct the shift register to skip one value that has a zero at the end to compensate for the value 11111111, which ends with a one. This adjustment will balance the number of ones and zeros in the final sequence of one-digit values.

To avoid the need to guess which value to skip, we will bypass the value 11111110 (decimal 254), which precedes the unused value 11111111. While blocking one of the input values may not be the most elegant solution, many LFSRs in the encryption industry, where large-scale LFSRs are utilized to generate pseudo-random numbers, employ more shift registers. However, this approach can be quite complex to implement.

6.3.4. Seed

There is one more problem to address. The core issue is that if there are no voltage surges when the circuit is powered on, the initial state of the shift register will consistently be 00000000. We cannot achieve the goal of making the sequence random if it always starts in the same state. Therefore, we need to initialize it with some unknown sequence value.

To accomplish this, we will utilize a standard technique in random number generation known as "seeding". This means that each time the generator is activated, it is initialized to a different value. In computer programs, this value is often derived from the current system clock value, as time is constantly changing. For our random-generating circuit, the ideal solution would be to allow it to run for an arbitrary number of idle cycles before applying its output.

To implement this, a system can be designed where, upon powering on, a combination of a resistor and a capacitor initiates a slow timer (operating at a low frequency) in monostable mode, generating a single pulse. The duration of this pulse can be configured using any type of sensor. During this pulse, the slow timer enables the fast self-oscillating timer to function, which feeds the clock signal to the LFSR. At the end of the slow timer pulse, the LFSR circuit halts at an unknown state of memory cells, thereby providing an ideal level of pseudo-randomness.

It's also worth noting that to seed the internal random number generator, it is a common practice to read the value of an unconnected pin of the microcircuit through the microcontroller's internal A/D converter.

7. Development of recommendations for the practical implementation of a generator of pseudo-random numbers based on sensors

7.1. Hardware requirements for implementing a sensor-based generator

For the successful implementation of a pseudo-random number generator (PSN) based on sensors, it is necessary to take into account several hardware requirements that will ensure stable operation and sufficient entropy of random number generation.

Selection of sensors. The main types of sensors that can be used to generate random numbers include:

- Temperature sensors: measure temperature fluctuations at the micro level, especially in sensitive environments (eg in cooling systems).
- Acoustic sensors: use microphones or other devices to pick up noise in the environment.

- Light sensors: measure changes in illumination or the infrared spectrum, which can vary even under controlled conditions.
- Gyroscopes and accelerometers: These can measure small oscillations or movements of the device that are difficult to predict.

Data collection performance and frequency. Sensor-based PRNG requires a sufficiently high sampling rate to provide a stable stream of random numbers. A typical sensor polling frequency can vary from a few hundred hertz to a few kilohertz, depending on the accuracy and characteristics of the signal.

- For sensors with a low frequency of data collection (for example, temperature sensors), it is possible to use filtering and accumulation of data to obtain sufficient entropy.
- For high-frequency sensors (acoustics or gyroscopes), a more direct data stream can be used with minimal processing.

Processing power. Since the signal from the sensors often needs processing (noise filtering, normalization, binarization), computing power is required. The generator can be implemented as:

- Embedded System on Microcontroller: For resource-constrained devices (e.g. IoT) where power consumption must be kept to a minimum.
- Hardware in servers or security modules: here a more powerful processor can be used to process large amounts of sensor data in real-time.

Energy consumption. Sensor-based PRNG can be critical in power-constrained systems such as mobile or IoT devices. Recommendations:

- Use sensors with low energy consumption.
- Adjust the polling frequency of sensors according to the needs of a specific application (reducing the frequency of data collection in cases where high entropy is not critical).

7.2. Methods of signal processing in practical implementation

To build a generator with a high level of unpredictability, it is necessary to implement effective signal processing methods:

- Signal filtering: application of median filters or low-pass filters to remove regular components (environmental fluctuations).
- Quantization and binarization: normalization of data from sensors and their transformation into binary sequences through a threshold value.
- Application of hashing or cryptographic algorithms to further improve the quality of random numbers and their security.

Let's consider some potential application scenarios for sensor-based generators.

Cryptography and security. The most important application scenario for such generators is cryptography. The use of sensor-based PRNG makes it possible to significantly increase the reliability and stability of security systems due to the generation of more unpredictable keys:

- Generation of cryptographic keys for data encryption.
- Protection against pseudo-randomness attacks: The physical nature of sensors allows protection against predictable algorithmic schemes.

Internet of Things (IoT). Sensor-based PRNG can be integrated into IoT devices, where resistance to hacking and ensuring the uniqueness of communication sessions is important:

- Device authentication through cryptographic algorithms that use random numbers to generate unique tokens.
- Encryption of transmitted data to protect privacy in IoT networks.

Gaming industry. In-game applications, randomness plays a key role in ensuring the integrity of the game and creating an unpredictable experience for users. Sensor-based PRNG can be used to:

- Generation of random events or numbers in-game processes (for example, in gambling, simulations, or generation of playing cards).
- Protection against player manipulation through the use of complex physical processes to generate unpredictable numbers.

Modeling and forecasting systems. Sensor-based generators can be used in scientific research where high-quality random numbers are required:

- Simulation of physical processes: in environments where the accuracy of random parameters is important, for example, in simulations of quantum systems or climatic conditions.
- Financial models: for random market changes or volatility forecasting.

Cyber-physical systems. Sensor-based PRNG can be integrated into cyber-physical systems (CPS), which combine physical components and computational processes. Examples:

- Security control in energy supply or production automation systems where random numbers are required to protect against hacking or sabotage.

Requirements for testing and certification. Before the practical implementation of sensor-based PRNG, it is necessary to conduct thorough testing for randomness and resistance to attacks [13, 14]:

- Use of NIST test suites to verify cryptographic strength.
- Carrying out statistical tests to assess the uniformity and entropy of random numbers.
- Certification to cryptographic security standards to ensure compliance with international standards such as FIPS 140-2.

Sensor-based pseudorandom number generators open new possibilities for applications where high unpredictability and reliability are required. For the successful implementation of such generators, it is necessary to ensure the correct selection of sensors, optimal signal processing, low power consumption, as well as thorough testing and certification.

Conclusions

The article considered one of the promising ways to build a pseudorandom number generator (PRNG) based on physical sensors. Sensors of various types, such as temperature, acoustic, light, gyroscopes, and magnetometers, are reliable sources of entropy due to their natural fluctuations and unpredictable characteristics. The physical phenomena measured by these sensors provide a high level of unpredictability, which is necessary for the construction of reliable pseudorandom numbers.

A method of digital processing of signals from sensors has been developed, which includes noise filtering, normalization, quantization, and binarization to obtain random bits. The evaluation of the quality of the obtained numbers using cryptographic and statistical tests showed that generators based on sensors can successfully provide a high level of randomness and resistance to predictability.

The article presents an approach for generating a pseudo-random sequence based on sensors. It discusses the factors influencing the randomness of devices that rely on environmental sensors, which can be utilized in the basic design of a slow monostable timer. The method proposed allows the sensor to generate a randomness factor without requiring any user input. A study was conducted on a robot, leading to suggestions for enhancing the circuit for the operation of a linear feedback shift register.

The concept of a linear feedback shift register is used in programming languages to create sequences of pseudo-random numbers. These languages also encompass the high-level languages employed for programming microcontrollers. Depending on the specific microcontroller used, various operators can enable the program to generate seemingly random numbers on demand.

It is important to note that the results from studies on random number generation functions in the C language version of the Arduino microcontroller system yield relatively evenly weighted sequences. However, there was a dependence on the range specified for generating random numbers, resulting in some values being encountered significantly more frequently than others.

Practical guidelines include selecting the appropriate sensor type based on specific application conditions and hardware requirements. Generators of this type can be effectively applied in industries where high security is required, including cryptography, the Internet of Things (IoT), the gaming industry, and cyber-physical systems.

Therefore, the generation of pseudo-random numbers based on physical sensors is a promising direction for creating reliable and secure generators that can be integrated into a variety of systems to provide high entropy and protect against possible attacks.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] F. Rastoceanu, R. Rughiniș, D.-C. Tranca, Lightweight cryptographic secure random number generator for IoT devices, in: 24th International Conference on Control Systems and Computer Science (CSCS), 2023, 180–185. doi:10.1109/CSCS59211.2023.00036

[2] A. Parisot, L. M. S. Bento, R. C. S. Machado, Testing and selecting lightweight pseudo-random number generators for IoT devices, in: IEEE International Workshop on Metrology for Industry 4.0 & IoT, 2021, 715–720. doi:10.1109/MetroInd4.0IoT51437.2021.9488454

[3] S.-M. Cho, E. Hong, S.-H. Seo, Random number generator using sensors for drone, in: IEEE Access, vol. 8, 2020, 30343–30354. doi:10.1109/ACCESS.2020.2972958

[4] P. Jindal, B. Singh, RC4 encryption—A literature survey, Procedia Computer Science 46 (2015) 697–705.

[5] B. Zhurakovskyi, et al., Secured remote update protocol in IoT data exchange system, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 3421 (2023) 67–76.

[6] V. Dudykevych, et al., Platform for the security of cyber-physical systems and the IoT in the intellectualization of society, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS, vol. 3654 (2024) 449–457.

[7] A. Ruhillo, Using smartphone sensors to generate cryptographic keys, Int. J. Innov. Technol. Explor. Eng. 9 (2020) 1025–1029. doi:10.35940/ijitee.C8994.029420
[8] G. Cotrina, A. Peinado, A. Ortiz, Gaussian pseudorandom number generator based on cyclic rotations of linear feedback shift registers, Sensors 20(7) (2020) 2103. doi:10.3390/s20072103
[9] U. Zia, M. McCartney, B. Scotney, A novel pseudo-random number generator for IoT based on a coupled map lattice system using the generalised symmetric map, SN Appl. Sci. 4(48) (2022). doi:10.1007/s42452-021-04919-4
[10] R. Florin, et al., Sensor-based entropy source analysis and validation for use in IoT environments, Electron. 10(10) (2021). doi:10.3390/electronics10101173
[11] S. Hong, L. Chang, Sensor-based random number generator seeding, in: Access IEEE, vol. 3, 2015, 562–568.
[12] N. Lv, T. Chen, Y. Ma, Analysis on entropy sources based on smartphone sensors (2020) 21–31. doi:10.1145/3442520.3442528
[13] S. Popereshnyak, Technique of the testing of pseudorandom sequences, Int. J. Comput. 19(3) (2020) 387–398. doi:10.47839/ijc.19.3.1888
[14] V. Masol, S. Popereshnyak, Joint distribution of some statistics of random bit sequences, Cybernetics Syst. Anal. 57(1) (2021) 139–145. doi:10.1007/s10559-021-00337-x