Software–Hardware Random Numbers Generating Complex Based on Mobile Device

Viktor Shynkarenko[1,†], Denys Ostapets[1,†] and Artur Opriatnyi[1,*,†]

[1] *Ukrainian State University of Science and Technologies, Lazaryana Street 2, 49010 Dnipro, Ukraine*

Abstract
The effectiveness of many modern technologies in such fields as statistics analysis, computer games, gambling, testing, computer graphics, simulation, cryptography, information security algorithms and any others depends on the quality of random number generating means. The article represents the development principles of the software – hardware pseudo and true random numbers generating complex based on using a mobile device. An analysis of entropy sources provided by mobile devices is submitted. The accelerometer, gyroscope, and magnetometer sensors of mobile device are chosen as entropy sources. The complex architecture and the TCP - connection based communication protocol between the complex parts have been developed. The total number of considered modes for generating pseudo and true random number sequences is six. The software of server, client and mobile parts of the complex is implemented. Research of pseudo and true random number sequences quality obtained in different modes using a statistical and visual tests suite is submitted. An analysis of the results and a comparison with the previously known ones is performed. Recommendations for using different generating modes are given.

Keywords
pseudorandom numbers, PRN, true random numbers, TRN, true random number generator, TRNG, pseudo random number generator, PRNG, mobile device, entropy source, accelerometer, gyroscope, magnetometer

1. Introduction

Modern technologies widely use random number sequences in various fields – from simulation modeling to cryptography. The effectiveness of many algorithms and systems used in these areas depends on the quality of random number generation.

There are two main approaches for obtaining random numbers [1]:

- Generation of pseudorandom numbers (PRN) using special algorithms or tables.
- Use of special hardware devices that generate true random numbers (TRN).

The PRN generation algorithm is a procedure that uses mathematical formulas or pre-calculated values to create numerical sequences that may appear random at first glance. PRN generators can produce many numbers in a short time and are deterministic, meaning that a certain sequence of numbers can be reproduced later if the initial number in the sequence is known.

Hardware devices for generating TRN use some form of noise signal (or entropy source). An entropy source is a physical source of information whose output either appears random or becomes random after applying a certain filtering/distillation process [2].

Workshop "Intelligent information technologies" UkrProg-IIT`2025 co-located with the 15th International Scientific and Practical Programming Conference UkrPROG'2025, May 13-14, 2025, Kyiv, Ukraine
[*] Corresponding author.
[†] These authors contributed equally.
shinkarenko_vi@ua.fm (V. Shynkarenko); odaua@i.ua (D. Ostapets); artur.opriatnyi@icloud.com (A. Opriatnyi)
0000-0001-8738-7225 (V. Shynkarenko); 0000-0003-1778-7770 (D. Ostapets); 0000-0001-7145-9677 (A. Opriatnyi)

For many existing PRN generation algorithms, a significant challenge is ensuring their high quality and unpredictability. The relatively high cost and limited speed of hardware devices for TRN generation are the main obstacles to their widespread use. Thus, the presented work is relevant.

The purpose of the work is to research the quality of random number sequences obtained by means of software–hardware true random number generator (TRNG) using different entropy sources provided by mobile devices. Tasks to be solved in the work: analysis and selection of a entropy source for a software–hardware generating complex of TRN provided by mobile devices; architecture design of the software–hardware complex based on mobile devices for generating random numbers; development of communication protocol between complex parts; software–hardware complex implementation; analysis of the quality of random number sequences in TRN and PRN modes using known sets of tests.

By mobile devices, we will refer to smartphones, tablets, and other similar devices running mobile operating systems.

2. Entropy sources available in mobile devices

The paper proposes using mobile device sensors as entropy sources. Sensors are devices used in mobile devices to detect various aspects of the surrounding environment [3]. Nowadays, there are quite a few sensors available in smartphones, which are built-in and help the smartphone function. Most of these sensors are designed to assess user experience. A random number generator that uses the sensors of a modern mobile device can generate high-quality random bit sequences [3]. The main sensors in a smartphone that can be used for random number generation are shown on Figure 1.

Motion sensors are used to monitor the movement of the device. These movements can include tilt, shake, rotation, or oscillation [4]. In [5], data obtained from motion sensors and other various types of sensors present on most boards used for IoT nodes are analyzed.

An accelerometer is a device that measures the reaction force induced by acceleration or gravity [4]. In mobile devices, accelerometers are mostly used for controlling the orientation of the image on the screen (portrait or landscape mode). The work [6] explores the possibility of implementing a random number generator using data obtained from an accelerometer.

A smartphone has two sensors that allow determining the physical orientation of the device – a magnetometer sensor combined with an accelerometer sensor [7]. These sensors are useful for determining the device's physical position. For example, the geomagnetic sensor can be used in combination with the accelerometer to determine the device's orientation relative to the magnetic north pole [8].

A light sensor is a device that regulates the brightness level of the screen [9]. It is available in both inexpensive and expensive smartphones. Depending on the intensity of light, this sensor controls the screen's brightness. In [10], the authors demonstrated that the type of ambient light measurements and sensor devices are sources of randomness.

The gyroscope sensor is responsible for measuring the rotational speed around the device's axis [9]. In [11], a random number generator based on a gyroscope is proposed for drones.

The barometer sensor is used to measure the surrounding environmental pressure [9]. For example, health apps on smartphones also use these sensors. Additionally, in the previously mentioned work [5], a random number generator based on barometer sensors was also considered.

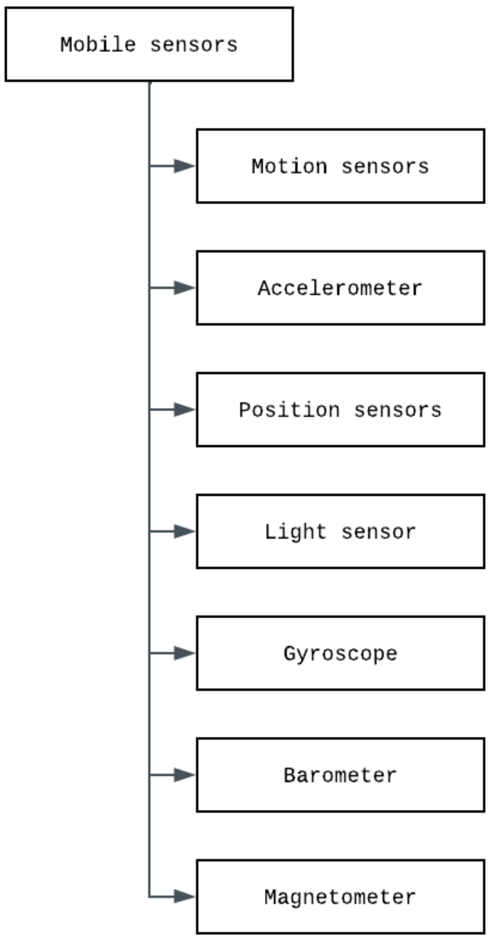


Figure 1: Main sensors available in mobile devices

A magnetometer is a sensor that measures the magnetic field [7]. There are two types of magnetometers: stationary, which are used for measurements at fixed points, and mobile, which are used in applications where movement detection is required.

When selecting an entropy source for developing a random number generator based on a mobile device, authors believe the following requirements should be considered:

- Sensor sensitivity.
- Availability of the sensor in most mobile devices.
- Data digitization speed.
- The number of bits obtained per measurement.

The accelerometer, gyroscope, and magnetometer are present in almost all modern smartphones, they are quite sensitive and allow for the acquisition of the highest number of bits because they capture changes in three coordinates, whereas the light sensor and barometer capture only a single value. Therefore, these sensors were chosen as the entropy source for the random number generator.

3. Complex architecture

This work proposes the implementation of a software-hardware system that includes a random number generator based on a mobile device and a server that provides clients with a convenient software interface for generating random numbers. The system should generate sequences of random (TRN) and pseudorandom (PRN) numbers. The PRN generator (PRNG) receives its seed from the TRNG [2].

The following generating modes are offered:

- generating of 32-bit unsigned TRN integers using entropy sources available in the mobile device (accelerometer/gyroscope/magnetometer);
- generating of 32-bit unsigned PRN integers, using hardware-generated seed (with accelerometer/gyroscope/magnetometer).

Thus, the total number of modes for generating number sequences is 6.

The TCP protocol was chosen for data exchange between the mobile device and the server, as well as between the server and clients. The structure and relationships between the elements of the system are shown in Figure 2.

A device running the iOS family of operating systems was chosen as the mobile device. For the pseudorandom number generation algorithm, the BBS (Blum-Blum-Shub) algorithm was selected due to its mathematical reliability and high entropy [12].

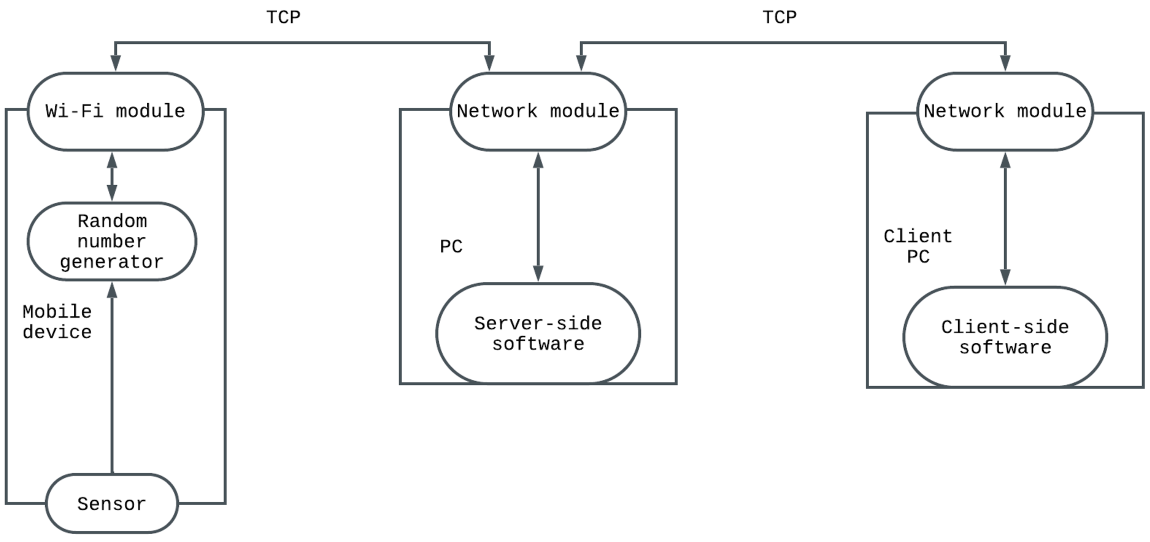


Figure 2: Communication between parts of the complex

A special protocol has been developed to implement data exchange between complex elements. The server must process 4 types of commands from the client: a command to set the generation mode (along with a field that identifies the mode), a command to set the entropy source (along with a field that identifies the entropy source), a command to generate a seed for the pseudorandom number generator, and a command to generate a random number modulo (along with a field specifying the modulus).

In response to these commands, 4 types of replies are expected. Among them, two types of errors exist: an error indicating an incorrect command format and an error indicating the absence of a connection with the mobile device. Additionally, two types of successful command execution responses exist: a response indicating the successful execution of the command and a response that returns the generated random number. The random number generating requests between the client and the server are presented in Table 1. The random number generating responses between the client and the server are presented in Table 2.

Table 1
Structure of random number generating requests between the client and server

Request name and interpretation	Request identifier	Possible parameters	
Mode setting	SM (set mode)	Parameter 1 – generator mode identifier: 0 – true random number generator, 1 – pseudorandom number generator	Parameter 2 – entropy source identifier: 0 – accelerometer, 1 – gyroscope, 2 – magnetometer
Random number generation	GR (generate random)	modulus (1 to 2^{32}-1)	

Table 2
Structure of random number generating responses between the client and server

Response interpretation	Response body
Number generated successfully	OK[generated number]
Generation mode is set	OK
Invalid request format	EF
Mobile device is not connected	EC

For the implementation of data exchange between the server and the mobile device, only one type of request is expected, where the entropy source identifier is specified (0 – accelerometer, 1 – gyroscope, 2 – magnetometer). The mobile device sends the generated number as response.

For the implementation of the server and client parts of the system, the Go programming language was chosen [13]. JetBrains Goland [14] was selected as the development environment for both the server and client parts. For the mobile device software development, the Dart language was chosen, which was created for convenient user interface development [15], along with the

Flutter framework for mobile operating system development [16]. Android Studio [17] was selected as the development environment for the mobile device, together with the Flutter plugin for software development using the Flutter framework. A simple console program has been developed as a client application, which establishes a TCP connection with the server and makes it possible to execute user commands entered from the keyboard.

4. Research of the quality of random numbers generated by the complex

Among the possible methods for testing the randomness of number sequences, authors chose to use the NIST test suite [18] and a visual graphical test [19]. The NIST test suite consists of 15 tests designed to assess the randomness of binary sequences of arbitrary length generated by both hardware and software random and pseudorandom number generators. The NIST test suite includes the following tests [2]:

- The Frequence (Monobit) Test.
- Frequency Test within a Block.
- The Runs Test.
- Tests for the Longest-Run-of-Ones in a Block.
- The Binary Matrix Rank Test.
- The Discrete Fourier Transform (Spectral) Test.
- The Non-overlapping Template Matching Test.
- The Overlapping Template Matching Test.
- Maurer’s “Universal Statistical” Test.
- The Linear Complexity Test.
- The Serial Test.
- The Approximate Entropy Test.
- The Cumulative Sums (Cusums) Test.
- The Random Excursions Test.
- The Random Excursions Variant Test.

The NIST Randomness Test Suite program [18] was used to verify the sequences by the NIST test suite. As a visual graphical test, it is proposed to generate an image where the color of each pixel represents a bit of the random sequence. Black color represents 1, and white represents 0. If the sequence of bits is random, there should be no visible patterns in the image, and the image should look like "noise" – a chaotic arrangement of black and white pixels. If there are repeating patterns, this may indicate that the sequence is not random. For the experiments, a laboratory set based on a MacBook Pro M3 (for running the client and server) and iPhone 14 is used.

Computer specifications:

- processor: Apple M3 Pro (12-core CPU, 18-core GPU).
- RAM: 36GB.
- operating system: macOS Sequoia (version 15.2).

All three sensor coordinates are converted into integers, which represent a bit sequence of coordinates according to the IEEE 754 standard [20]. Based on a series of experiments, it has been determined that the best randomness is achieved by the method of generating random numbers, which uses the XOR operation on the 4 least significant bits for the accelerometer, the XOR operation on the 16 least significant bits for the gyroscope, and the XOR operation on the 8 least significant bits for the magnetometer.

To verify the quality of the numbers generated by the developed system, 2000 numbers modulo 1024 were generated for all 6 possible mode variants. The results of testing the generated sequences of numbers using the NIST test suite showed that the obtained sequences in all 6 modes have characteristics of randomness. The testing was carried out using 10 tests from the set. For Overlapping Template Matching Test, Maurer's "Universal Statistical" Test, Linear Complexity Test, Random Excursions Test and Random Excursions Variant Test generated sequences don't meet the requirements of length.

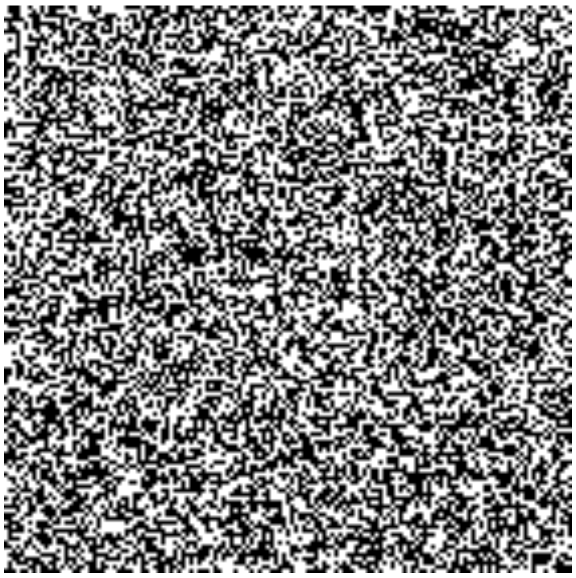
Figures 3–4 show the results of the graphical tests for each of the generation modes.

The approximate time to generate 1000 numbers on the used setup was as follows: for TRN using the accelerometer, 27 minutes 10 seconds; for TRN using the gyroscope, 7 minutes 9 seconds; for TRN using the magnetometer, 13 minutes 32 seconds; for PRN, 66.2 milliseconds.

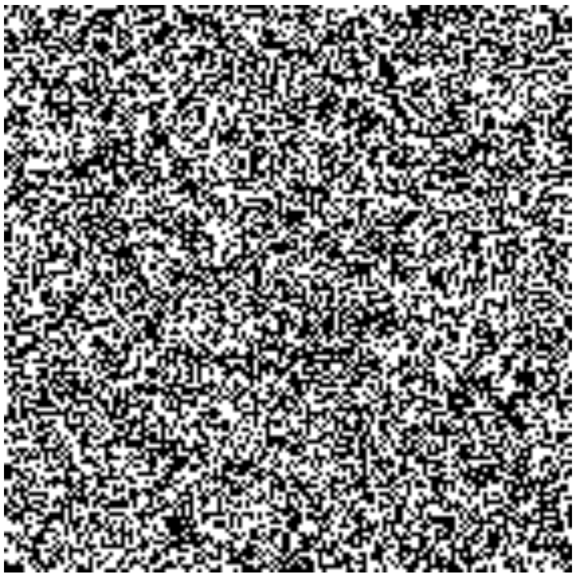
The results of testing the randomness of numbers generated using a mobile device in this work can also be compared with the results obtained in work [1]. In the specified work, the characteristics of the randomness degree of numbers generated by a device that uses the noise of a semiconductor device (Zener diode) were obtained. From the obtained results, it is clear that the generating of random numbers using accelerometer, gyroscope and magnetometer sensors available in mobile devices gives no worse results than the generating of random numbers based on a device that uses the noise of a Zener diode [1].

5. Conclusions

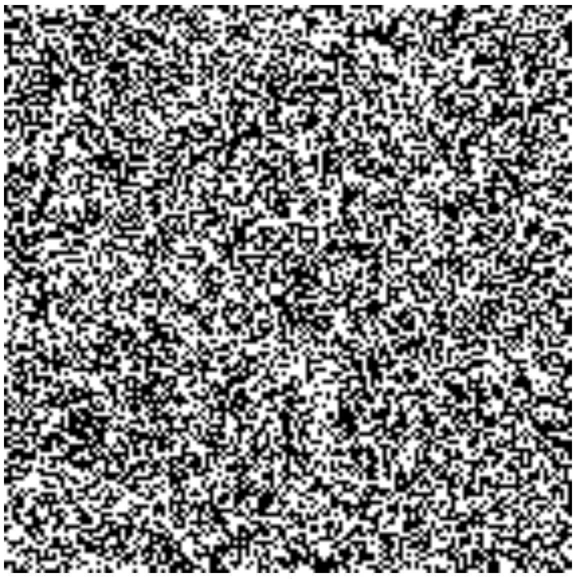
The developed software–hardware complex allows to obtain PRN and TRN sequences using the accelerometer, gyroscope, and magnetometer sensors of a mobile device as a source of entropy. Based on the results of the experiments, it has been determined that the PRN and TRN sequences obtained in all 6 modes meet the randomness requirements. In cases where high performance is required, the PRNG with the BBS algorithm and a true random seed can be used, as it produces results of similar quality to the TRNG based on the mobile device sensors. Also, sequences of numbers obtained using the accelerometer, gyroscope, and magnetometer sensors of a mobile device have no worse randomness characteristics than sequences obtained using the noise of a semiconductor device (Zener diode) as a source of entropy.



a)



b)



c)

Figure 3: Bitmap for TRN (generated by: a – accelerometer, b – gyroscope, c – magnetometer)

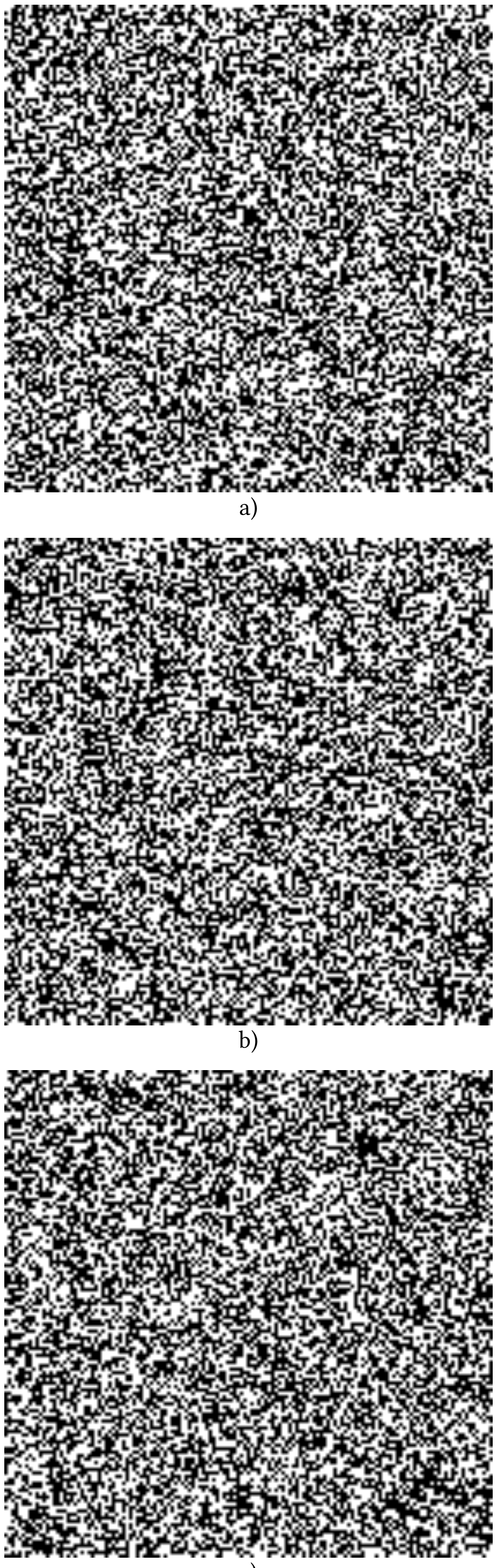


Figure 4: Bitmap for PRN (seed generated by: a – accelerometer, b – gyroscope, c – magnetometer)

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] D. Ostapets, V. Dziuba, P. Ivin, Hardware random numbers generator based on microcontroller, MATEC Web of Conferences 390, 04002 (2024). doi: 10.1051/matecconf/202439004002.

[2] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. URL: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf.

[3] Introduction to Randomness and Random Numbers. URL: https://www.random.org/randomness/.

[4] Motion sensors. URL: https://developer.android.com/develop/sensors-and-location/sensors/sensors_motion.

[5] C. Hennebert, H. Hossayni, C. Lauradoux, Entropy harvesting from physical sensors, Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks (2013) 149–154. doi:10.1145/2462096.2462122.

[6] J. Voris, N. Saxena, T. Halevi, Accelerometers and Randomness: Perfect Together, Fourth ACM Conference on Wireless Network Security (2011) 115–126. doi: 10.1145/1998412.1998433.

[7] Position sensors. URL: https://developer.android.com/develop/sensors-and-location/sensors/sensors_position.

[8] A. Suciu, D. Lebu, K. Marton, Unpredictable Random Number Generator Based on Mobile Sensors, IEEE International Conference on Intelligent Computer Communication and Processing (2011). doi: 10.1109/ICCP.2011.6047913.

[9] Environment sensors. URL: https://developer.android.com/develop/sensors-and-location/sensors/sensors_environment.

[10] M. P. Pawlowski, A. Jara, M. Ogorzalek, A. J. Jara, Harvesting Entropy for Random Number Generation for Internet of Things Constrained Devices Using On-Board Sensors (2015) doi: 10.3390/s151026838.

[11] S.M. Cho, E. Hong, S.H. Seo, Random Number Generator Using Sensors for Drone, *IEEE Access* (2020) doi: 10.1109/ACCESS.2020.2972958.

[12] L. Blum, M. Blum, M. Shub, A Simple Unpredictable Pseudo-Random Number Generator, SIAM Journal on Computing (1986) 364-383. doi: 10.1137/0215025.

[13] Build simple, secure, scalable systems with Go. URL: https://go.dev.

[14] Goland – Go Productive. URL: https://www.jetbrains.com/go/.

[15] Paint your UI to life. URL: https://dart.dev.

[16] Build for any screen. URL: https://flutter.dev.

[17] Android Studio. URL: https://developer.android.com/studio.

[18] NIST Randomness Testsuite. URL: https://github.com/stevenang/randomness_testsuite.

[19] B. Allen, Rseudo-Random vs. True Random. URL: https://boallen.com/random-numbers.html.

[20] IEEE Standard for Binary Floating-Point Arithmetic (IEEE Std 754-1985). URL: https://www.ime.unicamp.br/~biloti/download/ieee_754-1985.pdf.