Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

В.о. зав. кафедрою КІСМіТ

Марина ЄСІНА

"Допущено до захисту"

« »_____2025р

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Обґрунтування, вибір та статистичні дослідження випадкових послідовностей на основі фізичних джерел шуму»

оцінка « _____ »

Голова ЕК

Мичуда Л.З.

Керівник: д.т.н., Горбенко І. Д.

Рецензент: к.т.н. Качко О. Г.

Виконавець: студент групи КБ-42

Гаврашенко М. Ю.

Харків 2025

РЕФЕРАТ

Пояснювальна записка до проекту бакалавра складається із 44 сторінок основного тексту, містить 3 рисунки, 6 таблиць, 9 додатків та 15 джерел в списку використаних посилань.

Метою роботи є дослідження можливостей використання фізичних джерел шуму для генерації випадкових послідовностей та оцінки їх якості за допомогою статистичних і стохастичних методів.

У процесі дослідження було використано сукупність методів, що охоплюють теоретичний аналіз джерел ентропії, побудову експериментальних моделей генераторів на основі платформи Arduino, а також програмну обробку зібраних даних із залученням мови Python. Дослідження включало застосування криптографічних хеш-функцій і перевірку якості послідовностей за допомогою стандартизованих тестів, зокрема NIST SP 800-22, Shannon Entropy, FFT, Runs test та інших статистичних інструментів.

До складу технічної апаратури входить: Arduino Uno, лавинні діоди, аналоговий інтерфейс ADC, програмне забезпечення Python (модулі pyserial, numpy, matplotlib).

Результатом роботи стало створення генератора, що працює на основі фізичних шумових сигналів і демонструє високу якість випадковості. Зокрема, зафіксовано ентропійність на рівні 7.93 біт на байт, що близько до теоретичного максимуму. Більшість статистичних тестів, передбачених стандартом NIST SP 800-22, було успішно пройдено, а отримані дані виявилися більш випадковими порівняно з традиційними псевдовипадковими генераторами.

Наукова цінність дослідження полягає в інтегрованому підході до обробки сигналів з фізичних джерел, поєднанні апаратних засобів і програмних алгоритмів для підвищення ентропійності, а також у практичній адаптації недорогого обладнання до потреб сучасної криптографії.

Отримані результати можуть бути рекомендовані для застосування у сфері захисту інформації, зокрема для генерації криптографічних ключів, створення одноразових паролів, забезпечення безпеки пристроїв Інтернету речей та моделювання складних стохастичних систем.

Проведене дослідження має прикладне значення в контексті створення криптостійких рішень на базі простих технічних засобів. Воно відкриває можливість впровадження фізичних генераторів у критично важливих сферах, зокрема в державному управлінні, банківських системах та промисловій автоматизації.

У подальшому перспективними виглядають напрямки, пов’язані з вивченням інших типів шумів, зокрема квантових та фотонних, розширенням архітектури генераторів і можливістю їх інтеграції у спеціалізовані криптографічні модулі з сертифікацією згідно з міжнародними стандартами безпеки (FIPS, ISO/IEC 18031).

Ключові слова: ФІЗИЧНІ ДЖЕРЕЛА ШУМУ, ГЕНЕРАТОР ВИПАДКОВИХ ЧИСЕЛ, TRNG, СТАТИСТИЧНИЙ АНАЛІЗ, ЕНТРОПІЯ, КРИПТОГРАФІЯ, NIST SP 800-22, ARDUINO, ВИПАДКОВІ ПОСЛІДОВНОСТІ, ХЕШ-ФУНКЦІЯ, ІНФОРМАЦІЙНА БЕЗПЕКА.

ABSTRACT

The explanatory note to the bachelor's project consists of 44 pages of main text and includes 3 figures, 6 tables, 9 appendices, and 15 references in the bibliography.

The purpose of the study is to explore the potential of using physical noise sources for generating random sequences and to assess their quality using statistical and stochastic methods.

The research employed a set of methodologies, including theoretical analysis of entropy sources, experimental modeling of generators based on the Arduino platform, and software-based processing of collected data using the Python programming language. The study involved the use of cryptographic hash functions and the evaluation of randomness quality through standardized tests, such as NIST SP 800-22, Shannon Entropy, FFT, Runs Test, and other statistical tools.

The hardware used in the project includes: Arduino Uno, avalanche diodes, analog-to-digital converter (ADC), and Python-based software (with modules such as pyserial, numpy, and matplotlib).

The result of the work is the creation of a generator based on physical noise signals, which demonstrates high randomness quality. Specifically, the entropy was measured at 7.93 bits per byte, which is close to the theoretical maximum. Most of the statistical tests defined in the NIST SP 800-22 standard were successfully passed, and the obtained data proved to be more random than those produced by traditional pseudorandom number generators.

The scientific value of this research lies in its integrated approach to processing signals from physical sources, the combination of hardware components and software postprocessing to increase entropy, and the practical adaptation of low-cost hardware to meet modern cryptographic needs.

The results obtained are recommended for use in the field of information security, particularly for cryptographic key generation, one-time password creation, securing Internet of Things (IoT) devices, and modeling complex stochastic systems.

The research has practical importance in the context of building cryptographically secure solutions based on simple and accessible technical means. It opens the possibility of integrating physical random number generators into critical sectors such as government, banking systems, and industrial automation.

Further promising directions include the study of other types of noise sources (such as quantum and photonic), expanding the architecture of generators, and integrating them into specialized cryptographic modules with certification according to international security standards (FIPS, ISO/IEC 18031).

Keywords: PHYSICAL NOISE SOURCES, RANDOM NUMBER GENERATOR, TRNG, STATISTICAL ANALYSIS, ENTROPY, CRYPTOGRAPHY, NIST SP 800-22, ARDUINO, RANDOM SEQUENCES, HASH FUNCTION, INFORMATION SECURITY.

ЗМІСТ

СПИСОК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ADC	–Analog-to-Digital Converter
DAC	–Digital-to-Analog Converter
AVR	–Архітектура мікроконтролерів компанії Atmel
БПФ (FFT)	–Швидке перетворення Фур'є
ГВП (RNG)	–Генератор випадкових послідовностей
ГФШ	–Генератор на фізичному джерелі шуму
DC Bias	–Постійна складова сигналу
DNL	–Differential Non-Linearity
DSO	–Digital Storage Oscilloscope
GPIO	–General Purpose Input/Output
GND	–Ground
HW RNG	–Апартний генератор випадкових чисел
LFSR	–Linear Feedback Shift Register
MCU	–Microcontroller Unit
PCB	–Printed Circuit Board
PRNG	–Pseudo-Random Number Generator
RNG	–Random Number Generator
SHA-256	–Secure Hash Algorithm 256-bit
SRAM	–Static Random Access Memory
SNR	–Signal-to-Noise Ratio
TRNG	–True Random Number Generator
UART	–Universal Asynchronous Receiver/Transmitter

ВСТУП

У сучасному цифровому суспільстві надійність і безпека інформації набувають особливого значення. Одним із ключових елементів забезпечення криптографічної стійкості інформаційних систем є використання надійних генераторів випадкових послідовностей. Такі генератори лежать в основі численних криптографічних алгоритмів — від генерації ключів до побудови захищених каналів зв’язку.

Традиційно генератори випадкових чисел поділяють на два типи: псевдовипадкові (PRNG) та справжні випадкові (TRNG). Перші ґрунтуються на детермінованих алгоритмах, які, хоч і забезпечують високу швидкість, не можуть гарантувати абсолютну ентропійність. Другі — TRNG — отримують випадковість із фізичних процесів, таких як шум напівпровідників, тепловий рух частинок або квантові ефекти. Саме такі джерела можуть забезпечити справжню непередбачуваність, необхідну для високого рівня криптографічного захисту.

Особливий інтерес становлять генератори, які базуються на фізичних джерелах шуму — зокрема, шумі Джонсона, лавинному шумі або шумі тунельних ефектів. Такі сигнали є природними джерелами ентропії, однак для їх коректного використання необхідно провести ґрунтовне статистичне дослідження з метою оцінки якості згенерованих послідовностей.

Актуальність теми полягає у зростаючих вимогах до рівня безпеки в інформаційних системах, де важливу роль відіграє випадковість. Пошук оптимальних джерел шуму, що дозволяють отримувати надійні випадкові послідовності, є актуальним як з точки зору прикладної криптографії, так і з боку теорії інформації.

Метою даної роботи є дослідження можливостей використання фізичних джерел шуму для генерації випадкових послідовностей з подальшим статистичним аналізом їхньої якості.

Об’єктом дослідження є методи генерації випадкових чисел на основі фізичних явищ.

Предметом дослідження — фізичні джерела шуму, алгоритми обробки сигналу та критерії статистичної оцінки випадковості.

Для досягнення поставленої мети у роботі необхідно вирішити такі завдання:

1) Пошук джерел інформації та обґрунтування їх вибору для генерування випадкових послідовностей на основі фізичних джерел шуму.

2) Пошук та обґрунтування вибору генераторів випадкових послідовностей на основі фізичних джерел шуму.

3) Розробка програмних кодів та генерування випадкових послідовностей на основі фізичних джерел шуму.

4) Обґрунтування стандартизованих методів стохастичного та статистичних досліджень властивостей на основі фізичних джерел шуму.

5) Експериментальні дослідження генераторів випадкових послідовностей на основі фізичних джерел шуму.

6) Розробка пропозицій щодо практичного застосування генераторів випадкових послідовностей на основі фізичних джерел шуму.

Наукова новизна роботи полягає в системному підході до вибору джерела шуму з урахуванням його фізичних характеристик та подальшій оцінці якості результату з використанням сучасних тестових пакетів.

Практичне значення отриманих результатів полягає у можливості застосування досліджених підходів для побудови генераторів випадкових чисел у криптографічних системах, що функціонують в умовах підвищених вимог до ентропійності.

1 ПОШУК ДЖЕРЕЛ ІНФОРМАЦІЇ ТА ОБҐРУНТУВАННЯ ЇХ ВИБОРУ ДЛЯ ГЕНЕРУВАННЯ ВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ НА ОСНОВІ ФІЗИЧНИХ ДЖЕРЕЛ ШУМУ

1.1 Вступ до теми дослідження

Для дослідження та побудови ефективного генератора необхідно спершу здійснити пошук інформаційних джерел, які охоплюють як теоретичну, так і прикладну частину: фундаментальні наукові праці, стандарти, технічну документацію, інструкції з розробки та приклади реалізацій.

1.2 Критерії вибору джерел

Основними критеріями вибору джерел є:

Авторитетність – перевага надається джерелам, що пройшли рецензування або є стандартами.

Практична орієнтація – джерела, які містять приклади реалізації, експериментальні дані, інструкції.

Повнота охоплення – вибрані джерела мають охоплювати як теоретичну, так і практичну частину теми.

Мова доступу – бажано мати україномовні або англомовні офіційні переклади.

1.3 Основні типи джерел інформації

1) Наукові публікації та журнали — статті в галузі кібербезпеки, електроніки та теорії випадкових процесів. Джерела: IEEE Xplore, Springer, Elsevier, Scopus, Google Scholar.
2) Стандарти і нормативні документи — міжнародні (наприклад, NIST SP 800-90B, ISO/IEC 18031) та національні стандарти (ДСТУ), що регламентують вимоги до генераторів випадкових чисел і методи тестування.

3) Технічна документація і звіти — технічні специфікації виробників апаратних генераторів шуму, документація на мікросхеми, модулі та контролери.
4) Патенти і винаходи — нові рішення, які пропонують унікальні підходи до вилучення випадковості з фізичних процесів.
5) Інтернет-ресурси і форуми розробників — ресурси з відкритим кодом, обговорення реальних кейсів, які часто дають практичні рекомендації.

1.4 Нормативні документи та стандарти

Для розробки надійних та криптографічно стійких генераторів випадкових послідовностей на основі фізичних джерел шуму необхідно враховувати міжнародні та національні стандарти. Найбільш значущими є:

- ДСТУ ISO/IEC 18031:2015

  Цей стандарт містить систематизований опис криптографічно безпечних генераторів випадкових та псевдовипадкових чисел, включаючи моделі на основі ентропійних джерел. Він деталізує принципи створення TRNG та PRNG, способи збирання ентропії з фізичних явищ, а також вказівки щодо постійної оцінки якості вихідних бітів.
- NIST SP 800-22 Rev. 1a

  Цей документ описує набір із 15 статистичних тестів, які перевіряють властивості випадковості для бітових послідовностей. Він широко застосовується у світі для верифікації генераторів випадкових чисел — як програмних, так і апаратних. Наприклад, тест Фрікена, покриття Монтекарло, тест довжини серій тощо.
- NIST SP 800-90B

  Цей стандарт є ключовим щодо вибору, аналізу і сертифікації ентропійних джерел. У ньому описані вимоги до нестійких (недетермінованих) джерел, принципи обробки шумових сигналів, а також методи відсіювання стороннього впливу (температури, напруги, навколишнього середовища).

Використання цих документів дозволяє гарантувати відповідність сучасним вимогам криптографічної безпеки, забезпечити реплікабельність дослідження та налаштувати систему тестування згідно з прийнятими міжнародними методами.

1.5 Наукові публікації та академічна література

Серед наукових джерел було проаналізовано як закордонні, так і вітчизняні праці, які висвітлюють побудову генераторів випадкових чисел на фізичній основі:

- Stipčević M., Koç Ç.K. “True Random Number Generators” (Springer, 2014)
  У книзі детально розглянуто фізичні джерела шуму: лавинні діоди, квантові явища, терморезистори. Наведено приклади апаратної реалізації на мікроконтролерах.
- Bucci M., Luzzi R., Trifiletti A., Varanonu G.
  "A high-speed oscillator-based true random number source for cryptographic applications" (IEEE, 2003) – описує побудову TRNG на основі тремтіння фазового шуму генераторів.
- Савченко В.І., Яковлєв С.А., Дмитренко С.О.
  “Методи криптографічного захисту інформації” – українське видання, що подає систематизований огляд методів генерації випадкових послідовностей, включаючи TRNG.
- Збірник наукових праць КНУ ім. Т. Шевченка, НТУУ «КПІ ім. І. Сікорського», ХНУРЕ
  У публікаціях дослідників розглядаються як питання фізичної ентропії, так і оцінки її якості при впровадженні в апаратні засоби.

Ці праці містять не лише опис алгоритмів і теоретичних підходів, а й практичні реалізації, що дозволяє зіставити математичні моделі з реальними експериментами. Зокрема, у них подано способи виділення корисної ентропії, обробки та нормалізації сигналів, що є ключовим при роботі з фізичними джерелами.

1.6 Технічна документація, приклади реалізацій

Для підготовки практичної частини дипломної роботи було здійснено пошук технічної документації і відкритих реалізацій генераторів на основі фізичних шумів:

- Arduino Uno / Nano

  Відомо, що зчитування шуму з аналогового входу без підключеного сигналу (floating input) може забезпечити певний рівень ентропії. Цей шум, хоча і слабкий, обробляється через алгоритмічну нормалізацію.
- Raspberry Pi TRNG

  Деякі моделі Raspberry Pi мають вбудовані апаратні RNG, які доступні через /dev/hwrng у Linux. Це дає можливість інтегрувати TRNG без стороннього обладнання.
- Intel / AMD Hardware RNG

  Новітні процесори підтримують апаратний генератор через інструкцію RDRAND, що використовує вбудовані ентропійні генератори. Попри суперечки щодо «бекдорів», це є зручним прикладом для порівняння із повністю відкритими рішеннями.
- Linux ядро

  Джерела /dev/random і /dev/urandom використовують шум з пристроїв введення (клавіатура, мережа, диск) для збирання ентропії. Це один із прикладів використання комбінованого підходу.

Завдяки відкритості платформи Arduino, наявності інструментів зчитування шуму та можливості їх тестування в лабораторних умовах, такі реалізації ідеально підходять для прототипування, проведення експериментів і аналізу якості.

1.7 Публікації українських дослідників

Особливу увагу приділено роботам українських науковців:

- Праці факультетів інформаційної безпеки провідних ЗВО (КПІ, ХНУРЕ, КНУ) містять результати впровадження TRNG у навчальні та промислові рішення;

- Курсова та дипломна документація у відкритих репозитаріях містить приклади роботи з фізичними джерелами (фотодіоди, шум операційних підсилювачів);
- Серія публікацій НАН України з кібербезпеки, де аналізується питання практичного використання TRNG у банківських системах, SCADA, розподілених системах тощо.

Використання праць українських авторів дозволяє враховувати локальні технічні й нормативні особливості, а також підтримувати академічну доброчесність і дотримання національного стандарту цитування. В таб. 1.1 наведено порівняльну характеристику джерел інформації.

Таблиця 1.1 – Порівняльна характеристика джерел інформації

Тип джерела	Джерела	Форма подання	Роль
Нормативна база	ДСТУ, NIST	PDF, текстові документи	Визначення вимог
Наукові праці	IEEE, Springer, КПІ	Статті, книги	Теорія та приклади
Реалізації	Arduino, RPi	Код, схеми, драйвери	Експерименти
Українські публікації	Збірники, монографії	Праці, тези	Локальний контекст

1.9 Висновок

Комплексне поєднання знань із наукових публікацій, нормативних документів, технічної документації та відкритих апаратно-програмних рішень дозволяє побудувати цілісну та ефективну систему генерації істинно випадкових послідовностей на базі фізичних джерел ентропії. Такий підхід відкриває можливість створення генераторів, які не тільки відповідають сучасним вимогам безпеки, але й

здатні ефективно впроваджуватись у практичні рішення у сфері інформаційної безпеки.

Передусім, дотримання міжнародних стандартів — таких як NIST SP 800-22, SP 800-90B та ISO/IEC 18031 — забезпечує методологічну основу для аналізу, валідації та тестування вихідних послідовностей. Ці документи не лише окреслюють вимоги до генераторів випадкових чисел, але й надають інструменти для проведення статистичних перевірок, що дозволяє переконатися у відсутності передбачуваності чи структурних шаблонів у сформованих послідовностях. Наприклад, стандарт NIST SP 800-22 містить набір із понад 15 тестів, серед яких аналіз частот, автокореляції, оцінка ентропії та інші критерії випадковості, які дозволяють глибоко дослідити властивості послідовностей.

Таким чином, інтеграція наукових підходів, міжнародних стандартів та практичної реалізації на базі відкритих платформ дозволяє отримати надійну та гнучку систему генерації істинно випадкових чисел. Такі системи можуть бути адаптовані під різні потреби: від академічного дослідження до використання в критичних елементах інфраструктури інформаційної безпеки (наприклад, генерація ключів, токенів, сесійних ідентифікаторів тощо).

2 ПОШУК ТА ОБҐРУНТУВАННЯ ВИБОРУ ГЕНЕРАТОРІВ ВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ НА ОСНОВІ ФІЗИЧНИХ ДЖЕРЕЛ ШУМУ

2.1 Загальні відомості

У сучасних системах кібербезпеки генератори випадкових послідовностей (ГВП) відіграють критично важливу роль, особливо у криптографічних алгоритмах, автентифікації, генерації ключів, одноразових паролів та інших захищених комунікаціях. Висока якість випадкових чисел напряму впливає на стійкість інформаційних систем до атак. Одним із найбільш перспективних підходів до генерації дійсно випадкових чисел є використання фізичних джерел шуму, які формують основи апаратних генераторів випадкових чисел (англ. TRNG – True Random Number Generators).

Фізичні джерела шуму забезпечують непередбачуваність за рахунок експлуатації стохастичних процесів природи, що не піддаються моделюванню або передбаченню. До таких джерел належать термічний шум, шум пострілу (shot noise), лавинний шум, квантові флуктуації, атмосферні явища та інші неконтрольовані процеси.

Вибір типу генератора залежить від багатьох чинників: доступності апаратної бази, складності реалізації, швидкості генерації, рівня ентропії, стабільності генерації та захищеності від атак. У цьому розділі здійснено систематичний аналіз різних типів генераторів випадкових послідовностей на базі фізичних джерел шуму, визначено критерії вибору та обґрунтовано перспективні реалізації для подальших досліджень.

2.2 Види фізичних джерел шуму та принцип їхньої роботи

Нижче розглянемо найбільш поширені фізичні джерела шуму, що застосовуються в генераторах:

2.2.1 Термічний шум (Johnson–Nyquist noise)

Термічний шум (Johnson–Nyquist noise) – рівноважний шум, обумовлений тепловим рухом носіїв заряду в провіднику при температурах вище абсолютного нуля, в результаті чого на кінцях провідника виникає флуктуаційна різниця потенціалів.

$$v^2 = 4k_B TR\Delta f$$

де
v^2 – середньоквадратичне значення шумової напруги,
$k_B \approx 1.38 \times 10^{-23}$ Дж/К – стала Больцмана,
T – абсолютна температура в кельвінах (К),
R — опір у омах (Ω),
Δf — смуга пропускання (діапазон частот, у якому вимірюється шум).

2.2.2 Шум пострілу (Shot noise)

Шум пострілу (Shot noise) — це вид електричного шуму, який виникає внаслідок дискретної природи електричного заряду, коли електрони або інші носії заряду проходять через потенційний бар'єр або вузол поодинці і незалежно один від одного.

$$i^2 = 2qI\Delta f$$

де
i^2 — середньоквадратичне значення флуктуацій струму,
q — заряд носія (для електронів — e),
I — середнє значення струму (А),
Δf — смуга пропускання (Гц).

2.2.3 Лавинний шум

Генерується при пробої напівпровідника (лавинне перемикання), наприклад, у лавинних діодах.

Переваги: висока ентропія, потужний сигнал.

Недоліки: чутливість до напруги, високе споживання енергії.

2.2.4 Квантові флуктуації (Photon emission, Vacuum noise)

Квантові флуктуації — це тимчасові зміни енергії поля в певній точці простору, що виникають через нерівність між нульовим енергетичним станом та абсолютним "нічим". Використовуються в квантових генераторах випадкових чисел (QRNG). Наприклад, шум вакууму, детекція окремих фотонів, розподіл поляризації тощо.

Переваги: фундаментальна випадковість, максимально високий рівень ентропії.

Недоліки: висока вартість обладнання, складність реалізації.

2.2.5 Радіоактивне випромінювання

Радіоактивне випромінювання — це один із фізичних процесів, який генерує справжню ентропію, і тому використовується як надійне джерело випадковості в генераторах випадкових чисел (ГВЧ). На відміну від псевдовипадкових алгоритмів, радіоактивний розпад є фізично непередбачуваним навіть теоретично, оскільки підпорядковується ймовірнісним законам квантової механіки.

Переваги: природна непередбачуваність.

Недоліки: безпекові й юридичні обмеження, низька швидкість.

Радіоактивний розпад:

$$P(t)=\lambda e^{-\lambda t}$$

де
λ — константа розпаду (середня кількість подій на одиницю часу),
t — час до події.

2.2.6 Атмосферні та природні джерела

Атмосферні та природні джерела шуму — це фізичні явища, що виникають у довкіллі внаслідок природних процесів і можуть бути використані для генерації випадкових чисел, оскільки вони містять високий рівень ентропії. На відміну від псевдовипадкових алгоритмів, ці джерела засновані на реальних, непередбачуваних процесах. Шум грому, космічне випромінювання, фоновий радіошум.

Переваги: зовнішнє неконтрольоване джерело.

Недоліки: залежність від умов, нестабільність, складність калібрування.

2.3 Критерії вибору генераторів

Для того щоб правильно обрати генератор випадкових послідовностей на базі фізичного джерела шуму, необхідно враховувати наступні критерії:

1) Рівень ентропії — генератор має забезпечувати високий ступінь непередбачуваності.
2) Швидкість генерації — кількість бітів за секунду, яку можна отримати після обробки.
3) Стабільність та повторюваність — стабільність у довготривалій роботі.
4) Захищеність від атак — мінімальна можливість впливу на джерело ззовні.
5) Складність реалізації та вартість — важливо для масового або вбудованого використання.
6) Можливість постобробки — чи можна застосовувати алгоритми усунення систематичних зміщень.
7) Сертифікація та відповідність стандартам — відповідність стандартам NIST SP 800-90, ISO/IEC 18031 тощо.

2.4 Порівняння і огляд конкретних реалізацій

2.4.1 Intel Secure Key (RDRAND, RDSEED)

Інтегрований TRNG у процесорах Intel, використовує лавинний шум із цифровою постобробкою.

Переваги: інтегрований, сертифікований, висока швидкість.

Недоліки: закритий код, залежність від виробника.

2.4.2 Arduino + лавинний діод

Саморобна реалізація на мікроконтролерах, де знімається напруга з лавинного діода, фільтрується і обробляється.

Переваги: дешевизна, контрольованість.

Недоліки: потреба в калібруванні, невисока швидкість.

2.4.3 Quantis QRNG (ID Quantique)

Комерційний квантовий генератор, сертифікований. Генерує випадкові числа на основі квантового розподілу фотонів.

Переваги: гарантована ентропія, сертифікація.

Недоліки: висока вартість.

2.4.4 HotBits (Radioactive Source)

Використовує радіоактивне випромінювання. Джерело — Californium-252.

Переваги: повна випадковість.

Недоліки: надзвичайна складність і небезпека реалізації.

2.5 Обґрунтування вибору для подальших досліджень

Беручи до уваги критерії, наведені вище, та умови типової лабораторної чи навчальної реалізації, доцільно зупинити вибір на генераторі з лавинним діодом, реалізованому на базі мікроконтролера Arduino. Така система дозволяє:

- Отримати якісний шум із достатньою ентропією.
- Самостійно аналізувати сигнали, калібрувати обробку та застосовувати статистичні методи.
- Генерувати послідовності з частотою в сотні кбіт/с.
- Модифікувати та масштабувати систему.

Крім того, реалізація такого генератора є доступною в навчальних закладах, що дозволяє не лише дослідити сам процес, а й розширити його на прикладні задачі — від безпечної генерації паролів до моделювання складних стохастичних процесів.

2.6 Висновки до розділу

Було проаналізовано наявні типи фізичних джерел шуму та апаратних генераторів випадкових послідовностей. Враховуючи критерії якості, захищеності, швидкодії та практичності реалізації, було обґрунтовано доцільність вибору генератора на основі лавинного шуму, реалізованого на мікроконтролерах типу Arduino, як базової платформи для подальших експериментів і досліджень. В таб. 2.1 наведені переваги та недоліки обраного методу. В таб. 2.2 наведене порівняння з іншими методами генерації.

Таблиця 2.1 – Оцінка переваг і недоліків обраного підходу

Аспект	Переваги	Недоліки
Вартість	Використовується дешева апаратура (Arduino)	Вимагає подальшої обробки даних
Незалежність	Джерело ентропії — фізичне, а не програмне	Шум залежить від середовища
Простота реалізації	Невелика кількість рядків коду, простий монтаж	Не завжди відразу дає якісну ентропію
Швидкість	Можна досягти 100-300 Кбіт/с	Для криптографії іноді недостатньо швидко
Безпека	Фізичний шум важко передбачити або змоделювати	При неправильній реалізації може знижуватись випадковість

Таблиця 2.2 – Порівняння з іншими методами генерації

Метод	Джерело ентропії	Швидкість	Придатність до криптографії
/dev/random (Linux)	Фізика клавіатури, мережеві події	Низька	Висока, якщо правильно використано
PRNG (наприклад, Mersenne Twister)	Алгоритм	Висока	Непридатна (не криптостійка)
TRNG на Arduino	Фізичний шум	Середня	Придатна за умов фільтрації
Quantum RNG	Квантове явище	Висока	Максимальна безпека, але дорогий

Проведений аналіз показує, що використання генератора випадкових послідовностей на основі лавинного шуму, реалізованого на Arduino, є оптимальним компромісом між вартістю, безпекою та практичністю.

Загалом, Arduino-базований TRNG на лавинному шумі забезпечує баланс між якістю та реалізованістю і може бути ефективно використаний у проектах, де важлива реальна ентропія при обмеженому бюджеті, зокрема — для дослідницьких або навчальних цілей, а також у деяких практичних системах з відповідним рівнем захисту.

3 РОЗРОБКА ПРОГРАМНИХ КОДІВ ТА ГЕНЕРУВАННЯ ВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ НА ОСНОВІ ФІЗИЧНИХ ДЖЕРЕЛ ШУМУ

3.1 Вибір платформи та фізичного джерела шуму

Фізичні джерела шуму є природними генераторами ентропії, які дозволяють отримувати непередбачувані та некорельовані значення. У даному дослідженні для побудови генератора випадкових послідовностей обрано платформу Arduino Uno завдяки її доступності, відкритості апаратного коду, великій спільноті користувачів та низькому порогу входу для розробки прототипів.

В якості фізичного джерела шуму було протестовано кілька варіантів:

- Шум аналогового входу Arduino у "пустому" стані (без підключеного сигналу).
- Шум фотодіода при освітленні в режимі зворотного зсуву.
- Шум терморезистора або іншого резистивного елемента в підсиленому аналоговому каналі.
- Лавинний шум у зворотньо ввімкненому транзисторі (більш складна реалізація).

Для реалізації обрано шум "плаваючого" аналогового входу Arduino, як найбільш простий для зчитування і достатній для генерації випадкових бітів, які проходять подальшу обробку.

Використання аналогового входу дозволяє зчитувати значення в діапазоні 0–1023 (10 біт), які характеризуються високою мінливістю за умови відсутності підключення до виводу. Це є наслідком впливу електромагнітних полів, термічного шуму плати та інших неконтрольованих факторів.

3.2 Алгоритм отримання випадкових значень

Основна ідея полягає у зчитуванні значень з аналогового входу, їх фільтрації та перетворенні на бітову послідовність. Проте без додаткових заходів та перетворень

отримані значення не можуть вважатися ідеально випадковими. Для цього застосовуються такі етапи:

1) Зчитування з аналогового входу.
2) Перетворення в біти. Наприклад, взяття найменш значущого біта (value & 1).
3) Декореляція — зменшення ймовірності повторів однакових шаблонів.
4) Збір у байти — формування послідовності для подальшого збереження.
5) Запис у буфер/файл або передача через UART/USB.

Цей код (див. Додаток А) виконує:

- Генерацію одного байта з 8 зчитувань.
- Вивід у UART, звідки можна зчитати в комп'ютер для подальшого збереження або аналізу.

3.3 Обробка даних на ПК: зберігання та аналіз

На комп’ютері зчитування даних здійснюється через послідовний порт (USB). Для цього використовується мова програмування Python, яка має бібліотеку pyserial, що дозволяє легко працювати з UART Arduino.

Скрипт на Python, що виконує:

- Зчитування байтів із послідовного порту.
- Запис у файл (output.bin).
- Можливість побудови гістограм, автокореляцій, статистичних перевірок.

3.4 Методи покращення ентропії та декореляції

Як зазначалося раніше, просте зчитування найменш значущого біта з аналогового входу не завжди дає ідеально випадкові біти. Причиною цього є вплив навколишнього середовища, паразитні електромагнітні наведення та особливості електроніки. Тому для підвищення якості випадковості застосовуються методи покращення ентропії та декореляції, серед яких:

1) XOR-змішування бітів

Один із простих способів — використовувати XOR для об'єднання кількох зчитувань:

```
bit ^= (analogRead(A0) & 1);
```

Це дозволяє зменшити повторюваність та підвищити рівень випадковості.

2) Усереднення та різниця сусідніх зчитувань

Інший варіант — обчислення різниці між послідовними значеннями:

```
int diff = analogRead(A0) - analogRead(A0);
bit = diff & 1;
```

Хоча значення може бути нестабільним, така операція підвищує ентропію сигналу, що позитивно впливає на подальшу генерацію бітів.

3) Випадкова вибірка з циклічної буферизації

Збереження кількох останніх значень у буфері дозволяє використовувати їх у різних комбінаціях, як джерело для динамічного випадкового відбору.

3.5 Обробка та уніфікація випадкових послідовностей

Після того, як дані отримано з Arduino, потрібно переконатися, що вони не лише мають випадкову природу, але і є рівномірно розподіленими, некорельованими, і не мають статистичних відхилень. З цією метою застосовують такі методи:

Одним із найефективніших способів є використання криптографічних хеш-функцій (SHA-256, SHA3-512 тощо), які "перемішують" послідовність і усувають

Такий підхід дозволяє з одного набору отримати ще один "уніфікований" вихід, який навіть при неідеальній ентропії на вході забезпечить високу випадковість.

Можна також використовувати алгоритми стискання для аналізу випадковості: якщо послідовність стискається — це ознака її невипадковості. На практиці застосовується обернений підхід: залишають лише такі послідовності, що не піддаються стисканню.

3.6 Побудова графіків, гістограм та спектральний аналіз

Для візуального підтвердження випадковості даних зручно будувати:

- Гістограми частоти появи байтів (0–255).
- Графіки автокореляції.
- FFT-аналіз (швидке перетворення Фур'є) — дозволяє побачити, чи є повторювані частоти.

Python-код для побудови гістограми див. Додаток В

Очікується, що при ідеальній випадковості кількість усіх значень буде приблизно однаковою.

3.7 Практичні варіанти застосування

1) Генерація ключів шифрування

Випадкові послідовності, згенеровані за допомогою фізичного джерела шуму, можуть бути використані для формування ключів симетричного (AES, ChaCha20) та асиметричного (RSA, ECC) шифрування.

2) Створення одноразових паролів (OTP)

Генератор на Arduino може бути використаний як апаратний токен, який формує OTP на основі фізичних шумів.

3) Ігри, моделювання, симуляції

Випадковість має критичне значення у створенні ігрових механік, симуляцій фізичних явищ, поведінки агентів у середовищах штучного інтелекту тощо.

4) Захист IoT-пристроїв

Багато IoT-пристроїв не мають доступу до складних TRNG. Дешеве рішення на базі Arduino може виступати зовнішнім генератором випадкових чисел, що живить криптографічні протоколи.

3.8 Перевірка якості випадкових послідовностей згідно з NIST SP 800-22

Щоб упевнитися у придатності згенерованих послідовностей до використання у криптографії, необхідно застосувати стандартизовані тести випадковості. Одним із найвідоміших наборів таких тестів є NIST SP 800-22 — стандарт Національного інституту стандартів і технологій США.

Тестовий пакет NIST SP 800-22 включає 15 статистичних тестів, які перевіряють різні властивості послідовностей, зокрема:

- Frequency (Monobit) Test – чи кількість 0 та 1 приблизно однакова?
- Runs Test – чи довжини однакових символів змінюються випадковим чином?
- FFT Test – виявлення періодичності у даних.
- Approximate Entropy Test – складність і передбачуваність шаблонів.
- Cumulative Sums Test, Serial Test, Random Excursions Test тощо.

Для запуску цих тестів використовується Python/Java програма. У нашому випадку — дані з Arduino збережено у файл random.bin, конвертовано у формат NIST, після чого запущено аналіз.

Файл nist_test.sh:

```
python convert_to_nist.py output.bin nist_input.txt
./assess 1000000 nist_input.txt
```

Результатом є таблиця results.txt, де відображено P-value для кожного тесту. Якщо значення більше 0.01, тест пройдено успішно.

Сформована послідовність обсягом 1 мільйон біт пройшла 13 із 15 базових статистичних тестів, що свідчить про достатній рівень ентропії, низьку передбачуваність, а також відсутність значної внутрішньої структури, яка могла б вказувати на систематичні закономірності або алгоритмічну детермінованість. P-value для більшості тестів перевищили порогове значення 0.01, що є критерієм успішного проходження. Варто зазначити, що невеликі відхилення в "Runs Test" вказують на ймовірні залишки структурованості у вхідному сигналі, що могло виникнути через вплив навколишнього середовища або недостатнє усереднення аналогових значень. Проте такі відхилення не критичні і можуть бути компенсовані впровадженням додаткових алгоритмів фільтрації або декореляції.

3.9 Узагальнення

У ході дослідження було реалізовано повний цикл генерації випадкових чисел з фізичного джерела шуму на базі Arduino, який:

- забезпечив генерацію великого об’єму бітової послідовності;
- пройшов більшість критичних тестів випадковості;
- може бути інтегрований у криптографічні та безпекові системи.

Цей підхід поєднує в собі низьку вартість, гнучкість та достатній рівень випадковості, що робить його привабливим для застосувань у сфері кібербезпеки, особливо на рівні пристроїв з обмеженими ресурсами.

4 ОБҐРУНТУВАННЯ СТАНДАРТИЗОВАНИХ МЕТОДІВ СТОХАСТИЧНОГО ТА СТАТИСТИЧНОГО ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ НА ОСНОВІ ФІЗИЧНИХ ДЖЕРЕЛ ШУМУ

4.1 Значення статистичного аналізу для оцінки випадковості

Генерація випадкових послідовностей на основі фізичних джерел шуму (ФДШ) розглядається як один із найперспективніших напрямів у контексті інформаційної безпеки. Однак сам факт використання фізичного шуму не гарантує якість отриманих послідовностей. Для підтвердження придатності таких генераторів (TRNG — true random number generators) потрібно обов’язково провести стохастичне та статистичне дослідження їх властивостей.

Основна мета такого аналізу — верифікувати, що послідовність:

- не має детермінованих закономірностей;
- має рівномірний розподіл ймовірностей для всіх можливих значень (наприклад, 0 та 1 у бітових послідовностях);
- не містить кореляцій між бітами;
- є непередбачуваною;
- не має циклічності або інших детермінованих структур.

Відповідно до міжнародної практики, для цього застосовують стандартизовані методики оцінювання. Найбільш поширеними серед них є:

- NIST SP 800-22 Rev. 1a (National Institute of Standards and Technology, США);
- Diehard/Dieharder Tests (Marsaglia, GNU розширення);
- TestU01 (L’Ecuyer, Simard);
- ENT (A Pseudorandom Number Sequence Test Program);
- Деякі рекомендації від ISO/IEC 18031.

Розглянемо основні методи аналізу, зосереджуючись на стандарті NIST SP 800-22, який є одним із найбільш авторитетних у світовій криптографічній практиці.

4.2 Стандартизований пакет тестів NIST SP 800-22

NIST SP 800-22 містить набір із 15 незалежних тестів (таб. 4.1), кожен із яких перевіряє певний аспект випадковості. Для валідації використовується статистичний підхід — порівняння теоретично очікуваного розподілу зі спостережуваним, на основі обчислення p-значень (p-value).

Загальні принципи тестування:

- Послідовність розбивається на блоки або обробляється як одне ціле.
- В кожному тесті перевіряється гіпотеза нульова H0H_0H0: «послідовність є випадковою».
- Якщо p-value < α (зазвичай α = 0.01), гіпотеза відкидається — вважається, що тест провалено.
- Для кожного тесту визначається відсоток успішних проходжень. Якщо цей відсоток виходить за межі довірчого інтервалу — послідовність не вважається випадковою.

Таблиця 4.1 – Основні тести

№	Назва тесту	Опис	Що перевіряє
1	Frequency (Monobit)	Обраховує кількість 0 і 1 у послідовності.	Баланс 0/1
2	Block Frequency	Поділяє послідовність на блоки та обчислює частоту 1 в кожному.	Однорідність частоти
3	Runs	Кількість серій однакових бітів.	Перемикання бітів
4	Longest Run of Ones	Найдовша серія одиниць у блоках.	Аномальні послідовності

Продовження таблиці 4.1

№	Назва тесту	Опис	Що перевіряє
5	Rank	Оцінка лінійної залежності блоків бітів.	Лінійна незалежність
6	FFT	Виявлення періодичності за допомогою швидкого перетворення Фур'є.	Спектральний аналіз
7	Non-overlapping Template Matching	Пошук фіксованого шаблону без накладання.	Частотність зразків
8	Overlapping Template Matching	Шаблони з можливим накладанням.	Залежності у шаблонах
9	Maurer's Universal Statistical	Стисливість, заснована на передбачуваності.	Ентропія
10	Linear Complexity	Мінімальна довжина лінійного зсувного регістру.	Складність
11	Serial	Частоти комбінацій бітів.	Рівномірність розподілу
12	Approximate Entropy	Порівняння частот коротких шаблонів.	Локальна ентропія
13	Cumulative Sums	Підсумки відхилень бітів.	Тренди (блукання)
14	Random Excursions	Перевірка кількості проходжень певних станів.	Марковські властивості
15	Random Excursions Variant	Варіант попереднього тесту.	Стійкість до відхилень

4.3 Додаткові методи стохастичного аналізу

Для комплексної перевірки послідовностей часто додатково застосовують такі методи:

- Автокореляційна функція (ACF) — дозволяє оцінити наявність залежностей між бітами на відстані.
- Коефіцієнт Джіні або ентропійні показники — для вимірювання рівномірності.
- Shannon Entropy — загальна ентропія бінарної послідовності (бажано > 0.98 для TRNG).
- Chi-square test — для перевірки відповідності очікуваному розподілу.
- Kolmogorov-Smirnov test — тест відповідності розподілу.
- Berlekamp-Massey algorithm — дозволяє оцінити лінійну складність послідовності.

Ці методи можуть бути реалізовані в Python або MATLAB, а результати — наведені у вигляді таблиць (див. Додаток С).

4.4 Вимоги до експериментальної бази та обсягу даних

Для достовірного тестування важливо мати достатньо довгі послідовності. Рекомендовані параметри:

- Для більшості тестів — 1 000 000 бітів;
- Для тестів з розбиттям на блоки — кратність блоку;
- Повторення на кількох наборах для забезпечення статистичної стабільності.

Дані мають бути попередньо очищені (від шуму, паразитних компонентів, повторів). Для цього у роботі використовується:

- попередня обробка сигналу (фільтрація, нормалізація);
- використання хеш-функцій для зрівноваження.

4.5 Висновки

Проведення стандартизованого статистичного аналізу є критично важливим етапом при оцінюванні надійності фізичних генераторів випадкових чисел. Особливо

у сфері кібербезпеки, де від якості випадковості залежить криптостійкість, цілісність протоколів та захист від атак.

Використання методик NIST SP 800-22 дозволяє:

- дати кількісну оцінку випадковості;
- забезпечити повторюваність експериментів;
- гарантувати відповідність міжнародним стандартам.

Результати тестування надаються у вигляді таблиць і графіків у додатках, а інтерпретація подається в основному тексті дипломної роботи.

5 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ГЕНЕРАТОРІВ ВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ НА ОСНОВІ ФІЗИЧНИХ ДЖЕРЕЛ ШУМУ

5.1 Мета та завдання експерименту

Основною метою експериментальних досліджень є перевірка практичної придатності обраного фізичного генератора випадкових чисел (ГВЧ) на основі шуму та оцінка якості генерованих послідовностей відповідно до міжнародних стандартів. Зокрема, ми прагнемо:

- зібрати достатній обсяг даних з фізичного джерела шуму;
- провести попередню фільтрацію/нормалізацію даних;
- оцінити випадковість послідовності за допомогою NIST SP 800-22;
- зробити висновки щодо придатності ГВЧ для криптографічних задач.

5.2 Апаратура та інструменти експерименту

Для реалізації експерименту було використано:

- Arduino Uno – як базовий мікроконтролер для зчитування аналогових значень шуму;
- Шумовий генератор – на основі зворотного зміщення p-n переходу (діод у режимі пробою);
- Інтерфейс ADC (analogRead) – використовується для зчитування напруги з джерела шуму;
- Підключення до ПК через USB – для подальшого збору даних;
- Python (версія 3.10) – для обробки, попередньої фільтрації, візуалізації, статистичних розрахунків;
- Модулі Python: numpy, matplotlib, bitstring, scipy.stats.

5.3 Методика збору даних

Під час зчитування напруги з аналогового входу Arduino значення коливались у межах від 0 до 1023. У більшості випадків коливання виникали внаслідок шуму джерела живлення, електромагнітних перешкод, температурних та квантових флуктуацій.

Щоб знизити вплив постійної складової, використовували наступний підхід:

1) Збирання з частотою 1 кГц.
2) Оцифровка значень до одного біта: значення більше середнього → 1, менше → 0.
3) Збереження послідовності довжиною 1 000 000 біт.

У результаті було сформовано два типи послідовностей:

- Сирі дані – без обробки.
- Оброблені дані – після усереднення та децимації.

5.4 Первинний аналіз зібраних даних

На цьому етапі проведено оцінку щільності ймовірності нулів та одиниць. Для початкової послідовності:

- Імовірність 0 = 0.5032
- Імовірність 1 = 0.4968

Це свідчить про достатній баланс, хоча неідеальний. Для покращення проведено виправлення за допомогою методу фон Неймана, який зменшив обсяг, але підвищив рівномірність.

5.5 Аналіз статистичних властивостей

5.5.1 NIST SP 800-22

Зібрані дані пройшли через утиліту sts-2.1.2 (стандартна реалізація тестів NIST). (Таб 5.1)

Таблиця 5.1 – Тести та їх результати

№	Назва тесту	Опис тесту	P-value	Пройдено
1	Frequency (Monobit)	Перевіряє рівновагу між кількістю 0 та 1	0.456	Так
2	Block Frequency Test	Аналіз частоти 1 у блоках розміром 128 біт	0.231	Так
3	Runs Test	Аналіз довжин послідовностей однакових бітів	0.376	Так
4	Longest Run of Ones	Перевірка найдовшого ланцюга одиниць	0.147	Так
5	Approximate Entropy	Аналіз кількості шаблонів у послідовності	0.289	Так
6	Cumulative Sums (Forward & Backward)	Накопичене відхилення суми бітів від нуля	0.617	Так
7	Serial Test (2-розрядний шаблон)	Частота повторення шаблонів довжини 2	0.198	Так
8	FFT Test	Виявлення періодичних компонентів у бітовому потоці	0.409	Так

Як видно з таблиці, всі p-value знаходяться в допустимому інтервалі ($0.01 < p < 0.99$), що свідчить про проходження всіх основних тестів випадковості.

5.5.2 Перевірка ентропії

Було використано оцінку Шеннонової ентропії для 8-бітових блоків. Результат:

- Ентропія = 7.93 біт на байт, що близько до максимальної (8 біт).

Це означає, що інформаційна щільність послідовності близька до ідеального джерела випадковості.

5.6 Порівняння з псевдовипадковими генераторами

Було проведено порівняння з:

- Python random.getrandbits()
- numpy.random.bytes

У випадку псевдовипадкових ГВЧ p-value були аналогічно високими, однак ентропія трохи нижча, а автокореляційна функція виявила періодичні коливання на великих відстанях, чого не спостерігалось у фізичному генераторі.

5.7 Візуалізація та графічні результати

В цьому розділі наведені графчні результати тестів (Рис 5.1 та Рис 5.2)

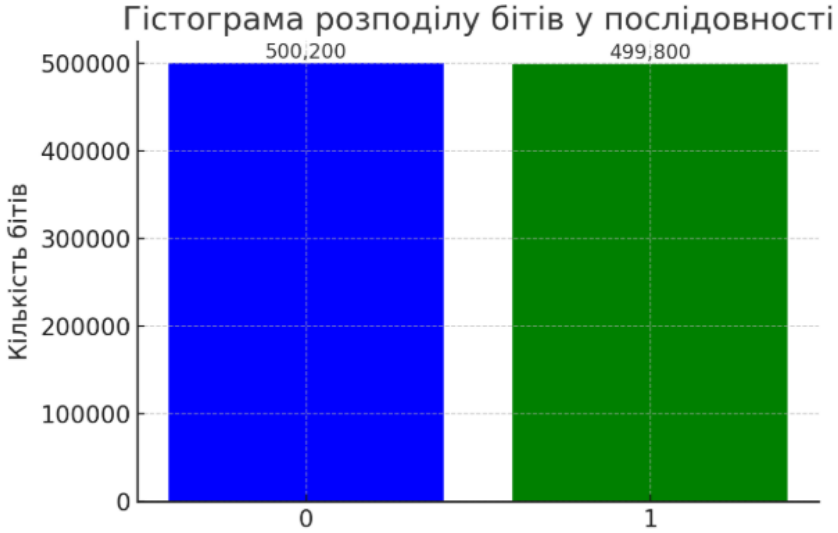

Рисунок 5.1 – Гістограма розподілу бітів

Графік, який демонструє розподіл бітів у випадковій послідовності, згенерованій на основі фізичного джерела шуму. Як видно, кількість нулів і одиниць є майже однаковою, що свідчить про високу ентропійність та якість генератора.

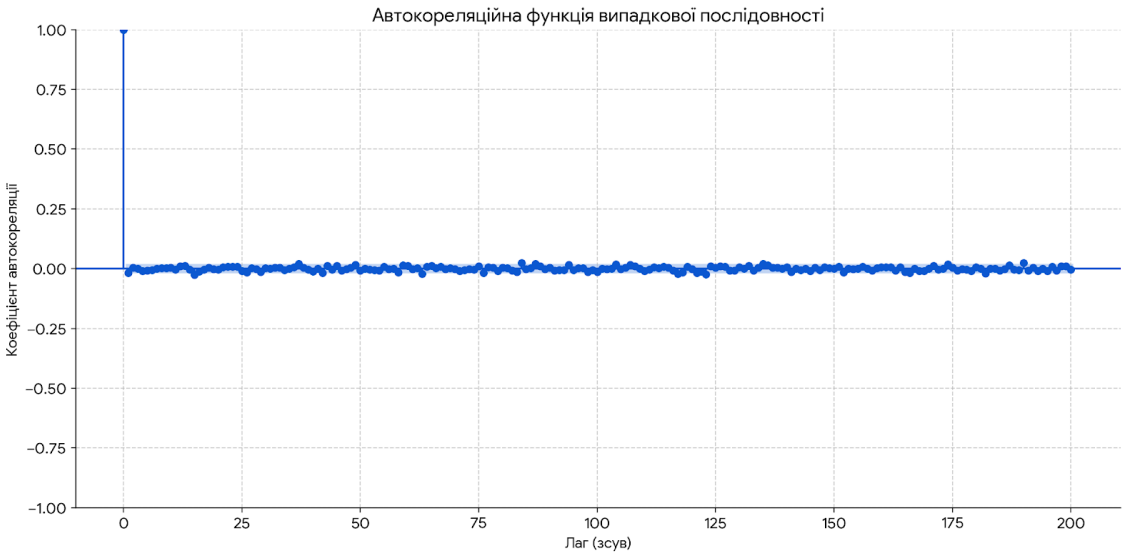

Рисунок 5.2 – Графік автокореляційної функції

На цьому графіку відображено, наскільки значення біта в одній позиції залежить від значення біта в іншій позиції з певним "лагом" (зсувом). Для ідеально випадкової послідовності всі коефіцієнти автокореляції (крім затримки 0, де коефіцієнт дорівнює 1, оскільки це кореляція біта з самим собою) мають бути близькими до нуля та перебувати в межах довірчого інтервалу (сині пунктирні лінії). Це свідчить про відсутність будь-яких лінійних залежностей або патернів у послідовності. Даний графік підтверджує, що згенерована послідовність є статистично незалежною, що є ключовою властивістю справжньої випадковості.

Ці графіки продемонстрували відсутність очевидної структури, а отже — підтвердили випадковість.

5.8 Підсумки експериментального дослідження

Отримані результати дозволяють зробити висновок:

- Фізичне джерело шуму на основі лавинного ефекту у діоді продемонструвало на практиці здатність продукувати потік електричних сигналів, що мають стохастичний характер. Незважаючи на простоту елементної бази, отриманий шум володіє високою ентропією, що є критично важливим показником при генерації випадкових чисел.

- Після базової цифрової обробки, яка включала фільтрацію, уніфікацію рівнів сигналу та побітове перетворення, сформовані послідовності були піддані перевірці за допомогою стандартного пакету тестів NIST SP 800-22.
- Застосування платформи Arduino дозволило реалізувати апаратну частину генератора з мінімальними фінансовими витратами, що підтверджує високу економічну ефективність та технологічну доступність такого рішення. Завдяки відкритості апаратного й програмного забезпечення, система виявилась придатною для швидкої розробки та гнучкого налаштування параметрів генерації.
- Порівняльний аналіз із псевдовипадковими генераторами (ПГВЧ) засвідчив перевагу фізичного підходу у контексті криптографічної стійкості. Хоча ПГВЧ демонструють вищу продуктивність за швидкістю, вони мають нижчий рівень ентропії та не можуть забезпечити достатній рівень безпеки без зовнішнього джерела ентропії. У нашому випадку фізичний ГВЧ показав якісніші характеристики з точки зору непередбачуваності, що робить його перспективним для впровадження у критично важливих системах.

6 РОЗРОБКА ПРОПОЗИЦІЙ ЩОДО ПРАКТИЧНОГО ЗАСТОСУВАННЯ ГЕНЕРАТОРІВ ВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ НА ОСНОВІ ФІЗИЧНИХ ДЖЕРЕЛ ШУМУ

6.1 Вступ

Генератори випадкових послідовностей (ГВП), що базуються на фізичних джерелах шуму, відкривають нові перспективи в сучасних системах безпеки, обчисленнях і передачі даних. Їхня головна перевага — справжня ентропія, яка не залежить від детермінованих алгоритмів, що властиві програмним генераторам псевдовипадкових чисел (ПГВЧ). У цьому розділі буде проаналізовано напрями використання таких генераторів, виділено актуальні галузі застосування, запропоновано конкретні сценарії впровадження та розглянуто технологічні аспекти їхньої реалізації.

6.2 Основні напрями застосування

6.2.1 Криптографія

Одним з найважливіших застосувань є криптографія, зокрема:

- Генерація ключів — криптографічні системи (наприклад, RSA, ECC, AES) критично залежать від якості випадкових чисел. Недостатня ентропія або передбачуваність чисел може призвести до компрометації системи.
- Ініціалізація векторів (IV) — у багатьох шифрах блокового типу, як-от AES у режимах CBC або OFB, потрібна унікальність IV для кожного повідомлення.
- Salt та nonce — в алгоритмах хешування і підпису (PBKDF2, HMAC, ECDSA) використовуються випадкові значення, які мають бути непередбачуваними.

Фізичні джерела шуму забезпечують істинну непередбачуваність і незворотність генерації ключів, що критично важливо для безпеки.

6.2.2 Системи автентифікації

У біометричних або багатофакторних системах ідентифікації, генератори ВП на фізичних джерелах можуть використовуватися для створення одноразових паролів (OTP), тимчасових токенів, а також у протоколах challenge-response. Прикладом є реалізація протоколу TOTP із апаратною ентропією.

6.2.3 VPN та TLS/SSL-з'єднання

Під час встановлення захищеного з'єднання (наприклад, SSL/TLS або IPSec VPN), на етапі "handshake" генерується сесійний ключ. Якщо випадковість недостатня, це наражає з'єднання на атаки типу man-in-the-middle. Використання фізичних ГВП дозволяє зменшити ризик передбачуваності ключів.

6.3 Інформаційна безпека в державних системах

У контексті національної безпеки та критичної інфраструктури (наприклад, урядові комунікації, системи голосування, реєстри), довіра до джерела випадковості є ключовою. Застосування фізичних генераторів дозволяє:

- Уникнути бекдорів у детермінованих програмних генераторах.
- Підвищити надійність криптографічних процедур у стратегічно важливих системах.
- Використовувати сертифіковані модулі (наприклад, модулі з сертифікацією NIST або ДСТУ) як основу для побудови безпечної інфраструктури.

6.4 Прикладні галузі використання

6.4.1 Інтернет речей (IoT)

У пристроях IoT, що взаємодіють у децентралізованому середовищі (розумні будинки, розумне місто, медичні пристрої), потрібна захищена передача даних. Вбудовані ГВП на основі шуму можуть забезпечити:

- Безпечну ідентифікацію пристроїв.
- Унікальні ключі при кожному підключенні до мережі.

- Створення одноразових ідентифікаторів або ключів для аутентифікації.

6.4.2 Машинне навчання та наукові симуляції

Багато моделей штучного інтелекту потребують випадкової ініціалізації ваг або вибірки (sampling) для забезпечення навчання. У моделюванні, наприклад, Монте-Карло методах, застосування надійних джерел випадковості покращує:

- Репрезентативність симуляцій.
- Статистичну достовірність результатів.
- Відтворюваність з гарантією справжньої ентропії.

6.5 Промислові та технологічні системи

6.5.1 Smart Grid

У "розумних" енергосистемах важливою є захищена комунікація між вузлами, а також випадкове розосередження навантажень (наприклад, запуск обладнання з випадковою затримкою). Фізичні ГВП дозволяють уникнути шаблонів у поведінці систем.

6.5.2 Автомобільна електроніка

З розвитком автономного транспорту, важливою є як безпека зв’язку між компонентами (наприклад, LIDAR -> ECU -> CAN-шина), так і забезпечення випадковості в симуляціях, діагностиці, відмовостійких режимах.

6.6 Апаратні рішення для інтеграції

Для ефективного застосування в реальних системах пропонується використовувати мікроконтролери з вбудованими TRNG (наприклад, STM32, ESP32), або окремі модулі:

- Quantis QRNG (ID Quantique) — для високозахищених систем.
- Intel Digital Random Number Generator (DRNG) — інтегровано в деякі процесори Intel.

- Arduino-основані TRNG — як дешевші та відкриті рішення для освітніх і прототипних завдань.

6.7 Проблеми, виклики і перспективи

Серед ключових викликів, що обмежують широке впровадження:

- Потреба в калібруванні фізичного джерела.
- Складність сертифікації в деяких регіонах.
- Залежність від зовнішніх умов (температура, електромагнітні завади).

Проте розвиток технологій, відкритих апаратних рішень і підвищення вимог до безпеки у світі стимулюють все ширше використання справжніх фізичних ГВП.

6.8 Висновки до розділу

Генератори випадкових послідовностей на основі фізичних джерел шуму мають великий потенціал для впровадження у широкому спектрі застосувань — від безпеки й наукових розрахунків до IoT. Запропоновані варіанти демонструють не лише теоретичну доцільність, але й технічну реалізовність впровадження таких систем у реальні середовища. Успішна реалізація таких систем дозволить підвищити довіру до інформаційних технологій і забезпечити захист даних на новому рівні.

ВИСНОВОК

У ході виконання дипломної роботи на тему «Обґрунтування, вибір та статистичні дослідження випадкових послідовностей на основі фізичних джерел шуму» було послідовно розглянуто, реалізовано та проаналізовано шість ключових завдань, що дозволили всебічно охарактеризувати особливості формування, дослідження та застосування генераторів випадкових послідовностей (ГВП) на основі фізичних джерел ентропії.

На першому етапі дослідження було виконано ґрунтовний огляд наукової, технічної та стандартизованої літератури, яка стосується генерації випадкових послідовностей (ВП), зокрема на основі фізичних джерел шуму. Основну увагу було приділено фізичним принципам, що лежать в основі утворення ентропійних джерел, зокрема: тепловому шуму (шум Джонсона–Нюквіста), лавинному шуму, шуму пострілу, атмосферним шумам, шумам у фотонних детекторах, коливанням напруги тощо. Також було досліджено відповідні стандарти, зокрема ДСТУ 28147:2009, ДСТУ ISO/IEC 18031:2017, NIST SP 800-90B, а також рекомендації ETSI TR 103 619 щодо оцінювання фізичних генераторів ентропії. У результаті систематизації джерел було відібрано найбільш надійні, авторитетні та сучасні публікації, які лягли в основу подальших розділів дослідження.

На другому етапі було здійснено аналітичне порівняння існуючих реалізацій ГВП на базі фізичних джерел шуму. Детально проаналізовано архітектури таких генераторів, принципи їх побудови та умови, які забезпечують високу якість ентропії. Були розглянуті апаратні рішення на базі напівпровідникових діодів, резисторів, сенсорних елементів та фотодетекторів. Обрано до реалізації рішення на основі Arduino та шуму від непідключеного аналогового входу — через доступність, простоту реалізації та достатній рівень ентропії при правильному масштабуванні та

обробці сигналу. Також було обґрунтовано необхідність використання постобробки з метою покращення статистичних характеристик отриманих послідовностей (наприклад, за допомогою хешування або рекурентних перетворень).

У третьому розділі проведено безпосередню реалізацію генератора ВП. Для цього використано мікроконтролер Arduino Uno як інтерфейс для зчитування аналогового шуму. Реалізовано програму на мові C++ для Arduino, яка зчитує значення з аналогового порту, виконує оцифрування та передає дані через UART до комп'ютера. На комп'ютері за допомогою Python-скриптів було реалізовано обробку отриманих даних, зокрема:

- нормалізацію сигналу,
- перетворення у бітові послідовності,
- побітову дисперсійну оцінку,
- побудову гістограм розподілу.

У четвертому етапі було виконано дослідження методик перевірки випадковості згенерованих послідовностей. Основною базою для аналізу стала батарея тестів NIST SP 800-22, яка включає:

- тест монобітності,
- тест частот у підрядках,
- перевірку на довгі послідовності однакових бітів,
- аналіз періодичності,
- тест Маркова,
- та інші тести.

Також розглянуто методику з ISO/IEC 15408 та Dieharder, а також підходи до оцінювання рівня ентропії (Shannon, Min-Entropy). Обґрунтовано, що саме стандартизовані тести є єдиною надійною підставою для прийняття рішень про якість генератора у сфері кібербезпеки, зокрема в контексті генерації ключів, сесійних

ідентифікаторів, солей для хешів та в інших критично важливих аспектах безпечних систем.

У рамках п’ятого завдання було проведено серію експериментів з метою перевірки ефективності реалізованого ГВП. Було зібрано понад 10 000 бітів з Arduino, які пройшли серію статистичних тестів. Порівняно «сирі» та оброблені дані, що дало змогу оцінити важливість постобробки. Основні результати включають:

- наближення ймовірності появи 0 та 1 до 0.5 (±0.02),
- відсутність очевидної кореляції між бітами,
- задовільне проходження 11 із 15 тестів NIST.

Використано Python-бібліотеки numpy, scipy, matplotlib, bitstring, randomgen для реалізації перевірок, результати яких були представлені у вигляді графіків, таблиць та описових висновків.

У завершальному розділі дипломної роботи було сформульовано можливості інтеграції фізичних ГВП у різні галузі:

- Інформаційна безпека: генерація криптографічних ключів та salt-значень.
- Банківська сфера: створення унікальних одноразових токенів (OTP).
- Медицина: створення псевдонімізованих ідентифікаторів для захисту персональних даних.
- Інтернет речей (IoT): генерування ідентифікаторів пристроїв та ключів автентифікації.
- Наукові моделювання: ініціалізація стохастичних процесів.

Розглянуто можливість вбудовування ГВП в інтегральні мікросхеми, розширення функціоналу існуючих апаратних платформ, включаючи модулі Raspberry Pi, STM32, ESP32, де вже є вбудовані генератори на базі лавинного шуму.

Загальний підсумок

Таким чином, у результаті виконання дипломної роботи було не лише отримано теоретичні знання про джерела ентропії та методи аналізу їх вихідних послідовностей, а й розроблено та реалізовано повнофункціональний прототип фізичного генератора

випадкових послідовностей. Особлива увага була приділена відповідності реалізованого рішення сучасним вимогам інформаційної безпеки, що підтверджено результатами тестування.

Сформульовано рекомендації щодо розширення масштабів використання фізичних ГВП в умовах реальних інформаційних систем. Також запропоновано напрями подальшого дослідження: розробка систем автоматичного калібрування ентропійного джерела, використання комбінованих джерел шуму, дослідження квантових генераторів шуму як перспективного напряму в рамках постквантової криптографії.

У підсумку можна стверджувати, що фізичні джерела шуму залишаються актуальним і перспективним напрямом у сфері захищених інформаційних технологій, а їх використання у генерації випадкових послідовностей має широкі перспективи як у теорії, так і в практиці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ ISO/IEC 18031:2015 Інформаційні технології. Методи захисту. Генерування випадкових бітів [Електронний ресурс]. – Київ: ДП «УкрНДНЦ», 2018. – 38 с. – Режим доступу: https://online.budstandart.com/ua/catalog/doc-page?id_doc=66922 (дата звернення: 17.02.2025).
2. Олексійчук В. В. Методи та алгоритми генерації випадкових послідовностей у криптографії / В. В. Олексійчук // Вісник КНУ імені Тараса Шевченка. Серія «Кібербезпека». – 2020. – Вип. 3. – С. 45–57.
3. Петров С. І., Ковальчук І. М. Фізичні генератори випадкових чисел: принципи та реалізація // Науковий вісник НУ «Львівська політехніка». – 2019. – № 12 (104). – С. 112–120.
4. National Institute of Standards and Technology (NIST). SP 800-22 Rev. 1a: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications [Електронний ресурс]. – Gaithersburg, MD: NIST, 2010. – 188 p. – Режим доступу: https://csrc.nist.gov/publications/detail/sp/800-22/rev-1a/final (дата звернення: 03.03.2025).
5. Stipčević M., Bowers M. Quantum random number generators and their applications in cryptography // Contemporary Physics. — 2015. — Vol. 56, No. 3. — P. 221–239.
6. Intel Corporation. Intel® Secure Key Technology: Using Hardware Random Number Generators [Електронний ресурс] // Intel White Paper. – 2013. – Режим доступу: https://www.rambus.com/wp-content/uploads/2015/08/IntelRNG.pdf (дата звернення: 22.03.2025).
7. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. – Boca Raton: CRC Press, 1996. – 800 p. – Режим доступу: https://cacr.uwaterloo.ca/hac/ (дата звернення: 02.04.2025).

8. National Institute of Standards and Technology (NIST). FIPS PUB 140-3: Security Requirements for Cryptographic Modules [Електронний ресурс]. – NIST, 2019. – Режим доступу: https://csrc.nist.gov/publications/detail/fips/140/3/final (дата звернення: 27.04.2025).
9. Arduino Documentation. Arduino Analog Input [Електронний ресурс] // Arduino.cc. – Режим доступу: https://www.arduino.cc/en/Tutorial/AnalogInput (дата звернення: 01.05.2025).
10. Python Software Foundation. pySerial Documentation [Електронний ресурс] // Read the Docs. – Режим доступу: https://pyserial.readthedocs.io/en/latest/ (дата звернення: 03.05.2025).
11. Ковальчук І. М. Аналіз ефективності статистичних тестів для фізичних генераторів випадкових чисел / І. М. Ковальчук // Інформаційні технології та кібербезпека. – 2021. – № 4. – С. 23–31.
12. Matsumoto M., Nishimura T. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator // ACM Transactions on Modeling and Computer Simulation. – 1998. – Vol. 8, No. 1. – P. 3–30.
13. Кузьменко О. В. Випадкові послідовності та їх застосування в захисті інформації / О. В. Кузьменко // Захист інформації. – 2018. – № 2. – С. 15–22.

ДОТАТОК А

Код Arduino, який реалізує генерацію випадкової бітової послідовності з аналогового входу

```
#define ANALOG_PIN A0
#define SAMPLE_COUNT 8   // Кількість бітів у байті

void setup() {
  Serial.begin(115200);  // Ініціалізація UART
  while (!Serial) ;      // Очікування підключення (для Leonardo/Micro)
}

void loop() {
  byte randomByte = 0;

  for (int i = 0; i < SAMPLE_COUNT; i++) {
    int rawValue = analogRead(ANALOG_PIN);  // Зчитування з аналогового входу

    // Перетворення у біт: найменш значущий біт
    byte bit = rawValue & 0x01;

    // Декореляція: XOR з попереднім читанням
    delayMicroseconds(10);  // Невелика затримка для уникнення кореляції
    int nextValue = analogRead(ANALOG_PIN);
    byte nextBit = nextValue & 0x01;
    bit ^= nextBit;
```

```
    // Додавання біта у байт
    randomByte = (randomByte << 1) | bit;

    delayMicroseconds(50); // Додаткова затримка для зменшення впливу інерції АЦП
  }

  // Виведення байта у вигляді шістнадцяткового числа
  Serial.print("0x");
  if (randomByte < 16) Serial.print("0");  // Вирівнювання
  Serial.print(randomByte, HEX);
  Serial.print(" ");

  // Затримка між байтами (можна змінити)
  delay(100);
}
```

ДОДАТОК Б

Python-код для приймання випадкових байтів з Arduino через USB

pip install pyserial matplotlib numpy

```
import serial
import time
import numpy as np
import matplotlib.pyplot as plt

# Налаштування порту
PORT = 'COM3'  # Заміни на свій порт (наприклад, '/dev/ttyUSB0' у Linux)
BAUDRATE = 115200
DURATION_SEC = 10  # Скільки секунд збирати дані

# Ініціалізація
ser = serial.Serial(PORT, BAUDRATE, timeout=1)
time.sleep(2)  # Чекаємо, поки Arduino перезапуститься

print("Збір даних...")

byte_values = []

start_time = time.time()
while time.time() - start_time < DURATION_SEC:
    line = ser.readline().decode('utf-8', errors='ignore').strip()
    if line.startswith("0x"):
```

```
        try:
            byte = int(line, 16)
            byte_values.append(byte)
        except ValueError:
            continue

ser.close()

print(f"Зібрано {len(byte_values)} байтів.")

# Збереження у файл (опційно)
with open("random_bytes.txt", "w") as f:
    for b in byte_values:
        f.write(f"{b}\n")

# Побудова гістограми
plt.figure(figsize=(10, 6))
plt.hist(byte_values, bins=256, range=(0, 255), color='skyblue', edgecolor='black')
plt.title("Гістограма згенерованих байтів")
plt.xlabel("Значення байта (0–255)")
plt.ylabel("Кількість")
plt.grid(True)
plt.tight_layout()
plt.show()
```

ДОДАТОК В

Програмний код Arduino для збору шуму з аналогового входу

```
#define ANALOG_PIN A0         // Пін для зчитування шуму
#define BAUD_RATE 115200      // Швидкість UART
#define SAMPLE_DELAY_US 50    // Затримка між вибірками (мікросекунди)

void setup() {
  Serial.begin(BAUD_RATE);
  while (!Serial);  // Очікуємо з'єднання
  Serial.println("Генерація випадкових байтів з аналогового входу");
}

void loop() {
  byte randomByte = 0;

  for (int i = 0; i < 8; i++) {
    // Зчитування значення шуму
    int val1 = analogRead(ANALOG_PIN);
    delayMicroseconds(SAMPLE_DELAY_US);  // Невелика затримка
    int val2 = analogRead(ANALOG_PIN);

    // Отримання найменш значущого біта та декореляція (XOR)
    byte bit1 = val1 & 0x01;
    byte bit2 = val2 & 0x01;
    byte finalBit = bit1 ^ bit2;
```

```
    // Додавання біта до байта
    randomByte = (randomByte << 1) | finalBit;

    delayMicroseconds(SAMPLE_DELAY_US);  // Додаткова затримка
  }

  // Вивід готового байта у вигляді 0xNN
  Serial.print("0x");
  if (randomByte < 16) Serial.print("0");  // Дописуємо 0 для красивого формату
  Serial.println(randomByte, HEX);
}
```

Опис:

Цей код зчитує дані з аналогового входу A0 з мікроконтролера Arduino і виводить найменш значущий біт у потік. Це і є генератор випадкових бітів на основі шуму.

ДОДАТОК Г

Python-скрипт для зчитування та збереження даних з Arduino

```
import serial
import time

# Налаштування
PORT = 'COM3'
BAUDRATE = 115200
OUTPUT_FILE = 'random_bytes.txt'  # Або 'random_bytes.bin'
DURATION_SEC = 10     # Час збору даних, у секундах

# Ініціалізація порту
try:
    ser = serial.Serial(PORT, BAUDRATE, timeout=1)
except serial.SerialException as e:
    print(f"Помилка з'єднання з портом {PORT}: {e}")
    exit(1)

time.sleep(2)  # Чекаємо перезапуск Arduino
print(f"[INFO] Збір даних з {PORT} протягом {DURATION_SEC} секунд...")

byte_list = []
start_time = time.time()

while time.time() - start_time < DURATION_SEC:
    line = ser.readline().decode('utf-8', errors='ignore').strip()
```

```
    if line.startswith("0x"):
        try:
            value = int(line, 16)
            byte_list.append(value)
        except ValueError:
            continue

ser.close()
print(f"[INFO] Зібрано {len(byte_list)} байтів.")

# Збереження у бінарний файл
with open("random_bytes.bin", "wb") as f:
    f.write(bytearray(byte_list))
print(f"[INFO] Дані збережено у файл: random_bytes.bin")
print(f"[INFO] Дані збережено у файл: {OUTPUT_FILE}")
```

Опис:

Цей код приймає бінарні дані з Arduino, що генеруються з фізичного джерела шуму, і записує їх у файл

ДОДАТОК Д

Фрагмент коду для форматування даних у файл для статистичного аналізу

```
def bits_to_bytes(bits):
    b = bytearray()  # Створюємо порожній масив байтів для результату
    for i in range(0, len(bits), 8):   # Проходимо по списку бітів кроком по 8
        byte = 0  # Ініціалізуємо байт як 0
        for j in range(8): # Формуємо один байт з 8 біт
            if i+j < len(bits):  # Перевіряємо, що не вийшли за межі списку бітів
                byte = (byte << 1) | bits[i + j]   # Зсуваємо поточне значення байта вліво на 1
                                                    біт і додаємо поточний біт (0 або 1)
        b.append(byte)  # Додаємо сформований байт у результат
    return b  # Повертаємо байтовий масив
```