УДК 004.71, 004.031.6, 004.021 **doi: 10.26906/SUNZ.2023.4.085**

А. О. Зуєв, Д. Г. Караман

Національний технічний університет «Харківський політехнічний інститут», Харків, Україна

ПРОГРАМНА РЕАЛІЗАЦІЯ СПЕЦІАЛІЗОВАНИХ АЛГОРИТМІВ ГЕНЕРАЦІЇ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ НА ПЛАТФОРМАХ ДЛЯ ВБУДОВАНИХ СИСТЕМ

Анотація. У статті проведено аналіз програмних реалізацій різних алгоритмів генерації псевдовипадкових чисел (ПВЧ) та псевдовипадкових бітових послідовностей (ПВБП), придатних для використання в складі вбудованих систем із обмеженими ресурсами: пристроїв Інтернету речей (IoT), бездротових сенсорних мереж (WSN), комплексів радіозв'язку малого радіусу дії та радіочастотної ідентифікації тощо. Виконано короткий огляд і аналіз існуючих платформ для розробки вбудованих систем, серед яких обрано рішення для проведення досліджень. Здійснено аналіз вимог до алгоритмів генерації ПВЧ, які підходять для реалізації у складі елементів систем із обмеженими ресурсами. Обрано найбільш характерні представники різних класів алгоритмів, реалізація яких буде найбільш доцільною. Проведено експериментальні дослідження, результати яких повинні допомогти розробникам у виборі найбільш підходящого алгоритму для генерації псевдовипадкових чисел у складі систем із обмеженими ресурсами.

Ключові слова: псевдовипадкові числа та бітові послідовності, генератори ПВЧ, вбудовані системи з обмеженими ресурсами, пристрої IoT, криптографічні алгоритми, ESP32.

Вступ

Розвиток і поширення вбудованих систем, таких як, наприклад, пристрої Інтернету речей (IoT), забезпечує їх глибоке проникнення в різні сфери діяльності людини: сільське господарство, побутова і комунальна сфера, промисловість, медицина. За оцінками провідних експертів у галузі [1], до початку 20-х років кількість розумних пристроїв, організованих в єдину мережу по всьому світу, має досягти майже 30 мільярдів, а в сферах, де пристрої об'єднуються в мережу та взаємодіють між собою без участі людини, — понад 15 мільярдів одиниць. Такий високий рівень інтеграції вбудованих систем у критично важливі галузі призводить до необхідності забезпечення цих пристроїв надійними засобами захисту інформації.

Існує практика застосування традиційних, добре перевірених, відточених засобів і алгоритмів криптографічного захисту даних, які використовуються в дротових та бездротових мережах загального призначення [2, 3]. Однак ці методи були розроблені для роботи в умовах без обмежень ресурсів на високопродуктивних обчислювальних системах. Більшість з них для виконання завдання вимагає великої кількості обчислень і оперує числами великої розрядності. У той же час, основні платформи для вбудованих систем будуються на мікропроцесорах і мікроконтролерах з невеликою продуктивністю, розрядністю, об'ємом пам'яті, а кінцеві рішення на їх основі мають ряд додаткових обмежень, наприклад, змушені працювати від автономного джерела живлення невеликої потужності і не мають можливості регулярного обслуговування для його підзарядження чи заміни. За таких умов пристрої змушені працювати в економному режимі, використовуючи короткі, адаптовані під архітектуру обчислення, що вимагають невеликого обсягу пам'яті, готові до частих і тривалих переходів базової платформи в різні енергозберігаючі режими роботи.

Такий стан справ уже протягом понад півтора десятків років активно надихає фахівців з криптографії та інформаційної безпеки на розробку нових та адаптацію існуючих рішень щодо захисту даних. Ці рішення повинні бути придатні для використання вбудованими системами із обмеженими ресурсами, а водночас забезпечувати криптографічний захист на рівні, який не поступається традиційним засобам.

Розв'язання основних завдань із захисту даних для вбудованих систем передбачає реалізацію цілого комплексу криптографічних перетворень. Методи криптографічного захисту використовуються для шифрування (забезпечення конфіденційності) даних, що зберігаються та передаються, виконання необхідних процедур з встановлення та підтримки захищених каналів зв'язку, ідентифікації та аутентифікації абонентів і таке інше. Одним з найважливіших елементів цього комплексу є генератори випадкових/псевдовипадкових чисел та бітових послідовностей.

Формулювання проблеми

У відповідності до принципу Керхгоффса, стійкість криптографічної системи захисту інформації повинна визначатися ключем шифрування. Ключ шифрування повинен бути достатньо великим і максимально непередбачуваним для зловмисника, щоб зробити завдання його відбору практично неможливим. Генерація подібних ключів — одне з завдань, яке мають виконувати генератори випадкових чисел та бітових послідовностей (ГВЧ і ГВБП). До завдань для ГВЧ (ГВБП) також входить генерація заповнень, одноразових послідовностей (nonce), так званої "солі", яка використовується для підвищення надійності зберігання паролів. У рідких випадках генератори псевдовипадкових бітових послідовностей можуть використовуватися як джерело гами шифру для здійснення потокового шифрування.

У будь-якому випадку генератор повинен відповідати жорстким вимогам щодо якості генерованих чисел або послідовностей з точки зору статистичних та кореляційних залежностей, а також бути безпечним з точки зору криптографії. Його механізм генерації повинен бути побудований таким чином, щоб з вихідної послідовності будь-якої довжини було не-

можливо відновити параметри налаштувань чи початкові значення стану генератора (вектори ініціалізації), а також неможливо передбачити наступні значення біт в послідовності чи наступні вихідні числа. Це досягається за рахунок використання станів, розмір яких значно перевищує розрядність вихідних значень, а також за рахунок багаторазового циклічного застосування досить складних перетворень. Такі заходи вступають у конфлікт з основними вимогами до методів, призначених для реалізації вбудованих систем: мінімальна обчислювальна складність, висока ефективність та швидкодія на наявних обмежених ресурсах, а також мінімізація енергоспоживання. У кінцевому підсумку розробник повинен уважно і зважено підходити до вибору генератора випадкових чисел, що буде реалізовано в системі, спираючись на дані досліджень статистичної та криптографічної надійності розглянутих кандидатів, а також на досвід їхньої реалізації для вбудованих систем.

Основною метою статті є аналіз ефективності та продуктивності програмних реалізацій обраних алгоритмів генерації псевдовипадкових чисел та бітових послідовностей для найбільш популярних і важливих платформ для побудови вбудованих систем в складі IoT.

Аналіз існуючих та перспективних рішень

В останній час було опубліковано багато оглядових публікацій, які розглядають питання використання різних методів генерації псевдовипадкових чисел (ГПВЧ) та бітових послідовностей (ГПБП) у системах IoT. Більшість з них стосується основних аспектів використання ГПВЧ для різних криптографічних цілей. У роботі [4] розглядаються різні методи отримання випадкових чисел, а також перспективи їх використання для систем IoT. Основний акцент робиться на алгоритмах, придатних для програмної реалізації в складі вбудованого програмного забезпечення чи операційних систем для IoT. Особлива увага приділяється криптографічній безпеці запропонованих рішень. Проведено якісні дослідження та оцінка результатів роботи розглянутих методів генерації випадкових чисел і послідовностей за допомогою різних пакетів спеціалізованих тестів. У статті [5] проведено детальний аналіз задач в системах IoT, для яких необхідне використання генераторів випадкових чисел. Для кожної задачі пропонується найбільш підходяще рішення, в зв'язку з чим автори докладно аналізують і оцінюють різні способи отримання випадкових та псевдовипадкових чисел. Виконаний огляд можна використовувати як рекомендацію та стартову точку при виборі або розробці генератора випадкових чисел під конкретну задачу для систем з обмеженими ресурсами. Також у статті розглянуті методи оцінки роботи генераторів випадкових чисел в складі систем, основні складнощі, які можуть виникнути при програмній чи апаратній реалізації генератора, а також способи їх вирішення. Деякі обзори присвячені аналізу використання ГПВЧ та ГПБП лише з точки зору криптографії та кібербезпеки. Наприклад, у роботі [6] автори провели широкий огляд публікацій, присвячених розробці, вибору та використанню генераторів псевдовипадкових чисел у складі вбудованих систем з певними обмеженнями, такими як пристрої IoT, вузли безпровідних сенсорних мереж (WSN), пристрої ідентифікації радіочастот (RFID) і т. д. У статті [7] проведений огляд основних методів отримання псевдовипадкових чисел та їх застосування в галузі забезпечення кібербезпеки. Проаналізовано характеристики методів та сфери їх застосування, зокрема в області криптографічного захисту та кібербезпеки, а також для використання у складі систем IoT. Розглянуті методи оцінки якості вихідних послідовностей і чисел, надано описи основних програмних інструментів, які можна використовувати для цього. В багатьох дослідженнях пропонуються практичні реалізації генераторів для використання у вбудованих системах. Наприклад, у роботі [8] запропоновано невибагливий генератор псевдовипадкових чисел Arrow, який належить до сімейства генераторів Trifork. Цей генератор базується на основі двох взаємопов'язаних генераторів Фібоначчі з запізнюванням (LFG), а також з внутрішнім взаємним перемішуванням. За заявами авторів, він підходить для широкого спектру завдань в галузі IoT. Проведено оцінку апаратних витрат у випадку, якщо алгоритм реалізується у інтегральному вигляді за технологією CMOS, та оцінку ресурсів, витрачених на програмну реалізацію для деяких пристроїв з екосистеми Arduino. Останнє показує не дуже високий рівень продуктивності, але, тим не менш, може бути прийнятною для деяких типів пристроїв, які працюють на невеликій тактовій частоті, наприклад, пристроїв радіочастотної ідентифікації (RFID). У роботі [9] представлено два варіанти генераторів псевдовипадкових чисел, що ґрунтуються на методах Блум-Блум-Шуба (BBS), Xorshift та конгруентній генерації псевдовипадкових чисел з перестановкою. Один варіант пропонується використовувати для цілей загального призначення, тоді як другий — для пристроїв IoT в умовах суворо обмеженого енергоспоживання. Апаратні реалізації обох методів на базі програмованої логіки FPGA показали прийнятні характеристики за енергоспоживанням та продуктивністю. Однак не у кожній системі в складі IoT може бути доцільно використовувати FPGA у якості основної платформи. Також, розробка систем з використанням програмованих логічних мікросхем вимагає спеціальної кваліфікації від розробника та є досить коштовною. Перевірка програмних реалізацій для найбільш популярних платформ вбудованих систем не проводилась. Хоча в роботі зазначається можливість використання запропонованих методів у галузі кібербезпеки, самі алгоритми не відносяться до класу криптографічно безпечних, що на практиці суттєво обмежує область їх можливого використання. У статті [10] описано практичний спосіб реалізації генератора випадкових чисел на базі доступних та недорогих компонентів. Генератор побудований за комбінованою схемою, де апаратна частина відповідає за формування джерела ентропії на основі оцифрованих образів непередбачуваних сигналів з навколишнього середовища, а програмна частина виконує додаткову обробку даних джерела ентропії. Автори не наводять даних щодо результатів синтезу і моделювання запропонованої схеми, а також не проводять хоча б статистичну оцінку генерованої послідовності. Іноді для побудови генера-

торів пропонується використовувати досить складні та навіть екзотичні методи. Наприклад, у статті [11] розглядається метод побудови генераторів послідовностей де Брейна на базі зсувних регістрів з нелінійними зворотними зв'язками (NLFSR). Запропонований авторами метод дозволяє синтезувати генератори псевдовипадкових чисел з послідовностями максимальної довжини для заданої розрядності зсувного регістра. Такі генератори особливо актуальні в системах захисту інформації, оскільки вони вимагають мінімальних ресурсів для реалізації і при цьому забезпечують більш якісну послідовність вихідних бітів, яка є значно менш передбачуваною ніж у регістрів з лінійними зворотними зв'язками. Описана в роботі схема дозволяє отримати генератор, але не дає ніякої перевірки результатів його роботи та оцінки криптографічної надійності запропонованого рішення. У статті [12] запропоновано метод побудови генератора псевдовипадкових чисел на основі хаотичних відображень. Якість отриманої випадкової послідовності додатково поліпшується за допомогою застосування функції приведення за модулем та перевіряється за допомогою набору математичних та статистичних тестів. Алгоритм шифрування використовує запропонований генератор псевдовипадкових чисел для захищеного передавання кольорових зображень, отриманих кінцевими пристроями в мережі IoT, за протоколом MQTT через бездротові канали зв'язку та Інтернет. Окрім стандартної оцінки якості випадкової послідовності, автори провели ряд криптоаналітичних досліджень запропонованої ними криптосистеми (побудова статистичних гістограм, оцінка ентропії та ключового простору, кореляція сусідніх відліків, диференційний криптоаналіз), проте ці дослідження не підтверджують криптографічну безпеку запропонованого алгоритму отримання псевдовипадкової послідовності.

Вибір та обґрунтування цільової платформи

Після проведення аналізу перспективних платформ для реалізації вузлів для різних типів вбудованих систем із обмеженими ресурсами (таблиця 1), серед усіх кандидатів особливу увагу звернули на себе комплекти розробки на основі мікроконтролера ESP32. ESP32 — це багатофункціональна система на кристалі (SoC), розроблена Espressif Systems — китайською компанією, що базується у Шанхаї. ESP32 позиціонується як автономне рішення для організації бездротових мереж Wi-Fi, яке може організовувати підключення будь-якого стороннього мікроконтролерного пристрою (МК) до Wi-Fi, а також здатне виконувати програми автономно на вбудованому МК [13]. До позитивних особливостей цієї платформи можна віднести низьку ціну, велику потужність, вбудований МК має 3 ядра, 2 з яких працюють на частоті до 240 МГц [14], а також багато різних апаратних модулів для вводу-виводу включаючи Wi-Fi та Bluetooth, що відкриває великі можливості для застосування цієї платформи для виготовлення різних пристроїв IoT [15]. ESP32 підтримується різними популярними середовищами розробки, такими, як Arduino, PlatformIO, або може програмуватися за допомогою фреймворку ESP-IDF від виробника МК.

Таблиця 1 – **Зведені характеристики популярних платформ для вбудованих систем**

Пристрій	ЦП	ОЗП	ПЗП (Flash)	Wi-Fi	Лінії вв./вив.	ОС	Вартість
Arduino	8-20 МГц	1-8 КБ	16-256 КБ	ні	14-54	N/A	$6-25
ESP8266	80 МГц	80 КБ	512 КБ	так	9	FreeRTOS	$3-12
ESP32	160 МГц	512 КБ	4-16 МБ	так	43	FreeRTOS	$5-15
RPi Pico (W)	133 МГц	256 КБ	2-16 МБ	ні (так)	26	N/A	$6-12
Omega2	580 МГц	64 МБ	16 МБ	так	12	Linux	$22
RPi Zero (W)	1 ГГц	512 МБ	16-32 ГБ	ні (так)	40	Linux	$15-20
C.H.I.P.	1 ГГц	512 МБ	4-8 ГБ	так	45	Linux	$50

Платформа ESP32 використовується для отримання та обробки даних з сенсорів [16], для керування сонячними панелями та системами іригації [17], для систем розумних будинків та систем контролю якості повітря [18], [19], а також для контролю систем життєдіяльності людини [20].

Характеристики досліджуваних алгоритмів генерації ПВЧ

На основі результатів аналізу існуючих рішень було обрано три алгоритми генерації ПВЧ: алгоритм Xoroshiro128++ (версія 1.0), алгоритм ISAAC, а також алгоритм потокового шифрування Salsa20, що може працювати в режимі генератора ПВЧ. Такий перелік можна пояснити наступним чином. Жоден з розглянутих методів не входить до складу стандартних або спеціалізованих бібліотек для створення програмного забезпечення для розглянутих платформ, тому, якщо в завданні з проектування був вказаний один із вибраних алгоритмів, розробнику довелося б створювати відповідне програмне забезпечення самостійно. Алгоритм Xoroshiro128++ не відноситься до числа криптографічно безпечних, але за іншими критеріями є одним із найбільш перспективних генераторів для систем з обмеженими ресурсами. Інші два алгоритми є криптографічно безпечними, причому алгоритм Salsa20, по суті, є алгоритмом потокового шифрування, який, за необхідності, можна використовувати як генератор ПВЧ.

Алгоритм генерації псевдовипадкових чисел Xoroshiro128++ дозволяє отримувати рівномірно розподілені 32-бітні цілі числа без знаку. Він був розроблений Себастьяном Вігна разом із Девідом Блекманом як один із варіантів у загальному сімействі алгоритмів xoshiro/xoroshiro [21]. Алгоритм побудований за допомогою трьох основних операцій: побітового виключного-АБО (xor), лінійного (shift) та циклічного (rotate) зсувів. Число в назві позначає розрядність внутрішнього стану. Існують також версії з 256-бітовим, 512-

бітовим та 1024-бітовими станами, проте саме генератор із 128-бітовим станом рекомендується для використання у вбудованих системах з обмеженими ресурсами. Генератори цього сімейства вже активно використовуються як стандартні для багатьох мов програмування та програмних бібліотек (Java, JavaScript, .NET, Rust, Lua, Julia), а також є генераторами за замовчуванням у операційних системах Mbed та Zephyr, спеціально розроблених для застосування в IoT [22].

Алгоритм ISAAC (скорочення від Indirection, Shift, Accumulate, Add, and Count) — криптографічно безпечний генератор псевдовипадкових чисел, розроблений Робертом Дженкінсом у 1993 році [23]. Структура алгоритму у багатьох відношеннях подібна до алгоритму потокового шифрування RC4, який іноді також використовується як генератор ПВЧ. Алгоритм містить масив з 256 32-розрядних цілих чисел як внутрішній стан, який частково змішується кожного разу під час генерації наступного числа. Для отримання випадкового числа з масиву обираються 2 числа з довільним значенням лічильника зі зсувом на 128 позицій одне відносно одного. Отримання результату відбувається приблизно за 19 операцій над 32-розрядними числами, що є досить швидким для 32-розрядних платформ. Основним недоліком алгоритму є відносно великий розмір стану для систем з обмеженими ресурсами, для його зберігання потрібно виділити постійно приблизно 1 КБ пам'яті, а також досить тривала процедура початкової ініціалізації. Однак сам робочий процес дуже швидкий та оптимізований для 32-розрядної архітектури, сам генератор є криптографічно безпечним: вихідні значення не виявляють жодних статистичних залежностей або закономірностей, за будь-якою послідовністю вихідних значень неможливо відновити початковий внутрішній стан, ймовірність виникнення коротших циклів є мізерною.

Алгоритм Salsa20 був розроблений Деніелом Берштейном, перш за все, як потоковий шифр, приблизно у 2005 році [24]. Він був представлений на конкурсі вибору кращих алгоритмів шифрування eSTREAM, який відбувався з 2004 по 2011 рік, і у результаті увійшов до числа фіналістів. Тим не менше, алгоритм може бути також використаний для отримання ПВЧ: його структура побудована на принципі ГПВЧ, які використовує набір операцій "додавання" (add), "циклічний зсув" (rotate), "виключне-АБО" (xor) над 32-розрядними операндами. Алгоритм використовує як початкові параметри 128- або 256-бітний ключ, 64-бітне значення вектора ініціалізації та 64-бітне значення лічильника. Далі всі дії виконуються над 512-бітним внутрішнім станом. Один етап отримання кінцевого результату в алгоритмі Salsa20 займає 20 циклів проходження по внутрішньому стану, після кожного другого виконується додаткова операція незворотного перемішування. Однак для генерації псевдовипадкових чисел можна використовувати спрощену версію алгоритму на 8 циклів. Під час виконання етапу створюється копія стану із зміненими даними, так що на момент виконання алгоритм потребує мінімум 128 байтів (1 Кбіт) оперативної пам'яті, проте між етапами достатньо зберігати лише поточний стан у 64 байта (512 біт). Якщо реалізація алгоритму Salsa20 є обов'язковим компонентом криптографічного стеку вбудованого пристрою, то використання алгоритму у повному чи скороченому варіанті як генератора ПВЧ дозволить суттєво економити пам'ять, а також дозволить отримати швидкий і надійний ГПВЧ.

Особливості реалізації та вибір мови програмування

Тест вимірювання швидкості є складним з багатьох причин. Вимірювання за допомогою вбудованого годинника МК може бути недостатньо точним через невисоку тактову частоту роботи пристрою. Деякі МК мають розвинуті можливості до паралелізму, таким чином відстеження початку і завершення виконання інструкцій є дещо недетермінованим і сильно залежить від інших подій. Сучасні процесори мають механізм зміни робочої частоти: вони цілеспрямовано стають повільнішими або швидшими, залежно від потреби здійснення обчислень: при відсутності навантаження або при виконанні певних команд вони можуть значно знижувати робочу частоту, що знижує енергоспоживання і подовжує можливий час роботи від обмеженого джерела живлення.

Ще одним важливим аспектом є вибір мови програмування для реалізації програми на МК. Зазвичай, цей вибір обумовлено підтримкою допоміжного апаратного забезпечення та периферійних пристроїв, таких, як інтерфейс введення/виведення сигналів загального призначення (GPIO), аналогові вхідні лінії та аналого-цифрові перетворювачі сигналів (ADC), провідні цифрові інтерфейси (SPI, I2C) та комунікаційні модулі (Wi-Fi, Bluetooth), які потрібні для вирішення задачі. Розробники МК майже завжди пропонують використовувати певні бібліотеки абстрактного уявлення апаратних ресурсів (HAL) для доступу до певних регістрів і периферійних пристроїв для деяких мов програмування (здебільшого це C/C++). Іншим аспектом є підтримка тої чи іншої мови високого рівня. Дуже часто занадто складні можливості мови програмування високого рівня обмежені або взагалі не реалізовані у версії компілятора для вбудованих апаратних платформ. Важливо також брати до уваги компілятор та інструменти, які будуть використовуватися для компіляції коду. Для вбудованих середовищ дуже важливою є здатність оптимізувати не тільки продуктивність скомпільованого застосунку, а ще й розмір його коду.

Виконання коду здійснюється під керуванням операційної системи, яка також виконує деякі інші завдання, наприклад обробку подій приймально-передавальних пристроїв, таких як модулі Wi-Fi або Bluetooth/BLE. Все це може приводити до варіативності часу виконання однакового коду.

Найбільш доцільнішим для реалізації алгоритмів та програм тестування ГПВЧ у відповідності до зазначених умов буде обрання мов програмування C/C++. Розробка на цих мовах програмування може здійснюватися для більшості платформ, вони забезпечують прямий контроль над розподілом пам’яті, а також для цих мов є необхідні бібліотеки, які підтримують обладнання та периферійні пристрої. Для порівняння, мова Python (MicroPython) не забезпечує

контроль за розподілом пам’яті, та має дещо низку продуктивність, а мова Go (TinyGo) також не має підтримки бездротових периферійних модулів (Wi-Fi та BLE) на платформі ESP32.

Тестування на платформі ESP32

Тестування проводилося на платформі MCU ESP32-WROOM v3.1: частота процесору 160МГц, програмне оточення ESP-IDF 5.01.0001. Програма написана на мові C++, компілювалася за допомогою компілятору GCC з налаштуванням на продуктивність (Optimize for performance -O2). Програма завантажувалась до Flash пам'яті МК та виконувалась під керуванням ОС FreeRTOS [25] з часом квантування 1 мс.

Було проведено заміри часу виконання 1 мільйону операцій генерації випадкових чисел розрядністю 32 біти за допомогою різних генераторів rnd(). Отримані числа додавалися до лічильника counter який гарантовано було розташовано у динамічній пам'яті МК (атрибут DRAM_ATTR), за допомогою операції “виключне-АБО”. Це потрібно для того щоб компілятор при оптимізації програми не відкидав виклики функцій результат яких не використовується. Шаблон коду для тестування:

```
static uint32_t DRAM_ATTR counter = 0;
void test(int steps)
{
    counter = 0;
    for(int i = 0; i < steps; ++i)
    {
        counter ^= rnd();
    }
}
```

Вимірювання часу виконувалося за допомогою внутрішнього лічильника МК (функція esp_timer_get_time()) з роздільною здатністю до 1 мкс. Також було виміряно час виконання циклу за шаблоном без генерування чисел, і отриманий час віднімався від виміряного по кожному генератору, для того щоб прибрати частку, яку безпосередньо не займає генерація (вона сягає в середньому 6304 мкс). Додатково було виміряно час, який потрібен для отримання значення з внутрішнього лічильника МК (він становить приблизно 1 мкс), і з'ясовано, що він суттєво не впливає на результати тестування. Всі тести запускалися з одного потоку операційної системи. Всі модулі МК (Wi-Fi, BLE та ін.), які потребують додаткових процесів ОС були відключені. Заміри часу для початкової ініціалізації генераторів випадкових чисел не проводилися, та час, потрібний на її виконання, в кінцевих результатах не враховувався. Всі змінні, необхідні для зберігання стану генераторів, також були розташовані у динамічній пам'яті МК (за допомогою вказання атрибуту DRAM_ATTR), або на стеку, для того щоб зменшити вплив затримки запиту даних з різних видів пам'яті МК. Результати тестування наведено у табл. 2.

Додатково було проведене тестування програмних реалізацій алгоритмів на частоті 240МГц, в якому з'ясовано, що для апаратного генератору випадкових чисел кількість циклів для отримання одного числа збільшилася з 118 до 126, а для решти методів залишилася незмінною, час отримання одного числа зменшився у 1.5 рази, що є пропорційним збільшенню частоти.

Таблиця 2 – **Результати порівняння швидкодії різних ГВЧ на ESP32 (1 ядро, 160 МГц)**

№	Тип генератора	Середній час виконання 1 млн. операцій, мкс	Час генерування 1 числа, мкс	Час генерування 1 числа, тактів
1	Порожній цикл	6304	0.006	1
2	Апаратний генератор випадкових чисел	744610	0.738	118
3	Вбудований генератор псевдо-випадкових чисел мови C++ (LCG/ MT19937/ SWC)	795060	0.789	126
4	Xoroshiro128++ 1.0	157369	0.151	26
5	ISAAC	329962	0.324	52
6	Salsa20	1019197	1.013	162

Аналіз ефективності та висновки

У даній роботі було виконано аналіз існуючих методів отримання псевдовипадкових чисел та бітових послідовностей для вбудованих пристроїв у складі систем з обмеженими ресурсами: систем IoT, мереж бездротових сенсорів, та пристроїв загального призначення з автономним живленням, які здатні об’єднуватись у локальні бездротові мережі. На базі проведеного аналізу було сформовано основні вимоги до таких методів. Разом з тим було розглянуто основні найбільш вживані платформи для реалізації пристроїв у складі систем з обмеженими ресурсами та обрано найбільш перспективну з них. Проведене експериментальне дослідження реалізації типових алгоритмів для отримання псевдовипадкових чисел показало що Xoroshiro128++ та ISAAC є найбільш продуктивнішими алгоритмами, які випереджають вбудований ГПВЧ в 4.8 – 2.4 рази. Отримані результати дозволяють зробити свідомий вибір алгоритму генерації псевдовипадкових чисел за необхідності їх реалізації у пристроях з обмеженими ресурсами. Слід зазначити, що, час виконання обраних алгоритмів для тестування, практично не змінювалися на протязі тестових запусків (похибка складала не більше 5 мкс). Окрім вбудованого ГПВЧ, для якого коливання часу складали до 100 мкс, що обумовлено особливостями реалізації доступу до змінних, які він використовує.

СПИСОК ЛІТЕРАТУРИ

1. Shafique K., Khawaja B. A., Sabir F., Qazi S., Mustaqim M. "Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios," in IEEE Access, vol. 8, pp. 23022-23040, 2020, DOI: 10.1109/ACCESS.2020.2970118.

2. Yang Y., Wu L., Yin G., Li L., Zhao H. "A Survey on Security and Privacy Issues in Internet-of-Things," in IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1250-1258, Oct. 2017, DOI: 10.1109/JIOT.2017.2694844.
3. Совин Я. Р., Наконечний Ю. М., Опірський І. Р., Стахів М. Ю. Аналіз апаратної підтримки криптографії у пристроях інтернету речей / Безпека інформації, Том 24 № 1 (2018), с. 36-48. DOI: 10.18372/2225-5036.24.12491
4. Kietzmann P., Schmidt T.C., Wählisch M. Kietzmann P., Schmidt T.C., Wählisch M. "A Guideline on Pseudorandom Number Generation (PRNG) in the IoT", ACM Computing Surveys (CSUR), Volume 54, Issue 6, 2020. pp. 1-38. DOI: 10.1145/3453159
5. Kübra Seyhan, Sedat Akleylek. "Classification of random number generator applications in IoT: A comprehensive taxonomy", Journal of Information Security and Applications, Vol. 71, Issue C, Dec 2022, DOI: 10.1016/j.jisa.2022.103365
6. Orue A. B., Hernandez-Encinas L., Fernandez V., Montoya F. "A review of cryptographically secure PRNGs in constrained devices for the IoT", International Joint Conference SOCO'17-CISIS'17-ICEUTE'17 León, Spain, 2017, Publ. in Advances in Intelligent Systems and Computing, vol 649. Springer, Cham. pp. 672–682. DOI: 10.1007/978-3-319-67180-2_65
7. Хомік М. А., Гарасимчук О. І. Застосування генераторів псевдовипадкових чисел та послідовностей в кібербезпеці, методи їх побудови та оцінки якості. Захист інформації. Т. 25 № 3 (2023). С. 147-159. DOI: 10.18372/2410-7840.25.17940
8. Orue A., Hernandez E. L., Martín A., Vitini F. "A Lightweight Pseudorandom Number Generator for Securing the Internet of Things", IEEE Access, vol. 5, pp. 27800-27806, 2017. DOI: 10.1109/ACCESS.2017.2774105
9. Bikram Paul, Apratim Khobragade, Soumith Javvaji Sai, Sushree Sila P. Goswami, Sunil Dutt, Gaurav Trivedi. " Design and Implementation of Low-Power High-throughput PRNGs for Security Applications", 2019 32nd Int. Conf. on VLSI Design and 2019 18th Int. Confe. on Embedded Systems (VLSID), 2019, Delhi, India, pp. 535-536. DOI: 10.1109/VLSID.2019.00123
10. Поперешняк С. В., Райчев О. О. "Модель легковагового генератора псевдовипадкових чисел для інтернету речей", Science-based technologies, 50(2), 2021, сс. 122-129. DOI: 10.18372/2310-5461.50.15670.
11. Miroschnyk M, Korytchinko T., Demihev O., Krylova V., Karaman D., Filippenko I. " Practical methods for de Bruijn sequences generation using non-linear feedback shift registers", 2018 14th Int. Conf. on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Eng. (TCSET), Lviv-Slavske, Ukraine, pp. 1157-1161. DOI: 10.1109/TCSET.2018.8336400.
12. Trujillo-Toledo D. A., López-Bonilla O. R., García-Guerrero E. E., Tlelo-Cuautle E., López-Mancilla D., Guillén-Fernández O., Inzunza-González E. " Real-time RGB image encryption for IoT applications using enhanced sequences from chaotic maps", Chaos, Solitons & Fractals, Volume 153, Part 2, December 2021, 111506. DOI: 10.1016/j.chaos.2021.111506
13. ESP32 Technical Reference Manual. Version 5.0 Espressif Systems Copyright © 2023. https://www.espressif.com.
14. ESP32 Series Datasheet. Version 4.3 Espressif Systems Copyright © 2023. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
15. Maier, A.; Sharp, A.; Vagapov, Y. Comparative Analysis and Practical Implementation of the ESP32 Microcontroller Module for the Internet of Things. In Proceedings of the 2017 Internet Technologies and Applications (ITA), Wrexham, UK, 12–15 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 143–148.
16. Hangan, A.; Chiru, C.-G.; Arsene, D.; Czako, Z.; Lisman, D.F.; Mocanu, M.; Sebestyen, G. Advanced Techniques for Monitoring and Management of Urban Water Infrastructures—An Overview. Water 2022, 14, 2174. DOI: 10.3390/w14142174
17. Allafi, I.; Iqbal, T. Design and Implementation of a Low Cost Web Server Using ESP32 for Real-Time Photovoltaic System Monitoring. In Proceedings of the 2017 IEEE Electrical Power and Energy Conference (EPEC), Saskatoon, SK, Canada, 22–25 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–5.
18. Carducci, C.G.C.; Monti, A.; Schraven, M.H.; Schumacher, M.; Mueller, D. Enabling ESP32-Based IoT Applications in Building Automation Systems. In Proceedings of the 2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT), Naples,Italy, 4–6 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 306–311.
19. Tastan, M.; Gökozan, H. Real-Time Monitoring of Indoor Air Quality with Internet of Things-Based E-Nose. Appl. Sci. 2019, 9, 3435. DOI: 10.3390/app9163435
20. Sangeethalakshmi, K.; Preethi Angel, S.; Preethi, U.; Pavithra, S.; Shanmuga Priya, V. Patient Health Monitoring System Using IoT. Mater. Today Proc. 2021. DOI: 10.1016/j.matpr.2021.06.188
21. Blackman D., Vigna S. "Scrambled Linear Pseudorandom Number Generators", ACM Transactions on Mathematical Software, Volume 47, Issue 4, Article No.: 36, pp. 1-32. DOI: 10.1145/3460772
22. Vigna S. "xoshiro / xoroshiro generators and the PRNG shootout", https://prng.di.unimi.it/
23. Robert J. Jenkins Jr. "ISAAC and RC4", 1993-1996. http://burtleburtle.net/bob/rand/isaac.html
24. Bernstein D.J. "The Salsa20 Family of Stream Ciphers." In: Robshaw, M., Billet, O. (eds) New Stream Cipher Designs. Lecture Notes in Computer Science, vol 4986. Springer, Berlin, Heidelberg, 2008. DOI: 10.1007/978-3-540-68351-3_8
25. FreeRTOS. Available online: https://docs.espressif.com/ projects/esp-idf/en/latest/esp32/api-reference/*system/ freertos.html (accessed on 15 November 2022).*

Received (Надійшла) 25.09.2023
Accepted for publication (Прийнята до друку) 29.11.2023

Software implementation of specialized algorithms for generating pseudorandom numbers on embedded systems platforms

A. Zuev, D. Karaman

Abstract. The article presents an analysis of software implementations of various pseudorandom number (PRN) and pseudorandom bit sequence (PRBS) generation algorithms suitable for use in embedded systems with constrained resources: Internet of Things (IoT) devices, wireless sensor networks (WSN), short-range radio communication complexes, and radio frequency identification systems, among others. A brief overview and analysis of existing platforms for embedded system development are provided, and a solution is chosen for further research. The requirements for PRN generation algorithms suitable for implementation in elements of resource-constrained systems are analyzed. The most representative algorithms from various classes are selected for implementation. Experimental research is conducted, and the results are intended to assist developers in choosing the most suitable algorithm for generating pseudorandom numbers in systems with limited resources.

Keywords: pseudorandom numbers and bit sequences, pseudorandom number generators, embedded systems with constrained resources, IoT devices, cryptographic algorithms, ESP32.