Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

СИСТЕМА АВТОМАТИЧНОГО ЗАХИСТУ SMART HOME З МОБІЛЬНИМ УПРАВЛІННЯМ

SMART HOME AUTOMATIC SECURITY SYSTEM WITH MOBILE CONTROL

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІм-21
Крив'янчук Назар Сергійович

(підпис)

Керівник:
к.т.н., доцент
Гринюк Сергій Васильович

(підпис)

Кваліфікаційну роботу
допущено до захисту
«_____» грудня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Гринюк Сергій Васильович

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т.ТЕРЛЕЦЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Крив'янчуку Назару Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Система автоматичного захисту Smart home з мобільним управлінням*

Керівник роботи *к.т.н., доцент Гринюк Сергій Васильович*

затверджені наказом закладу вищої освіти від «17» червня 2025 року № 290/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 09.12.2025р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз сучасного стану технологій «Smart Home»
Проектування та розробка архітектури системи захисту
Експериментальні дослідження та аналіз ефективності системи
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Загальна структурна схема системи автоматичного захисту
Діаграма обміну повідомленнями в MQTT з використанням QoS рівнів
Структурна схема SoC ESP32
Схема підключення сенсорів до мікроконтролера

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз сучасного стану технологій «Smart Home»	*Гринюк С.В., доцент*		
Проектування та розробка архітектури системи захисту	*Гринюк С.В., доцент*		
Експериментальні дослідження та аналіз ефективності системи	*Гринюк С.В., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Гринюк С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І., ст.викладач*		

7. Дата видачі завдання 18.06.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	До 01.08.2025 р.	
2.	*Аналіз сучасного стану технологій «Smart Home»*	До 20.08.2025 р.	
3.	*Проектування та розробка архітектури системи захисту*	До 25.09.2025 р.	
4.	*Експериментальні дослідження та аналіз ефективності системи*	До 20.10.2025 р.	
5.	*Висновки та пропозиції*	До 25.10.2025 р	
6.	*Формування списку використаних джерел*	До 27.10.2025 р.	
7.	*Формування додатків*	До 30.10.2025 р.	
8.	*Оформлення ілюстративного матеріалу*	До 05.11.2025 р.	
9.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	До 11.11.2025 р.	
10.	*Нормоконтроль*	До 29.11.2025 р.	
11.	*Інструментальна перевірка на академічний плагіат*	До 02.12.2025 р.	
12.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	До 09.12.2025 р.	

Здобувач вищої освіти _____ Крив'янчук Н.С.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Гринюк С.В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Крив’янчук Н. С. Система автоматичного захисту Smart home з мобільним управлінням. Рукопис.

Кваліфікаційна робота магістра ОП «Комп’ютерна інженерія» спеціальності 123 Комп’ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків.

У першому розділі проведено аналіз ринку сучасних систем безпеки, порівняно професійні та DIY-рішення. Обґрунтовано вибір трирівневої архітектури IoT (Edge-Fog-Cloud) та протоколу MQTT для забезпечення енергоефективності. Сформульовано функціональні вимоги до системи та розроблено модель станів (FSM) для детермінованого керування охороною.

У другому розділі розроблено структурну схему системи на базі топології «Зірка». Обґрунтовано вибір апаратної платформи (SoC ESP32) та сенсорної бази (HC-SR501, MQ-2, геркони). Розроблено алгоритми цифрової фільтрації сигналів та стратегію енергозбереження (Deep Sleep). Спроєктовано архітектуру мобільного додатку на базі фреймворку Flutter з використанням патерну BLoC.

У третьому розділі здійснено практичну реалізацію розробленої системи у вигляді діючого прототипу. Описано методику та результати експериментальних досліджень, які підтвердили ефективність запропонованих архітектурних та програмних рішень. Зокрема, перевірено стабільність роботи каналів зв’язку, часові характеристики реакції системи та показники енергоспоживання, що дозволило зробити висновок про відповідність розробки технічному завданню та її готовність до практичного застосування.

Ключові слова: Smart Home, IoT, ESP32, mqtt, flutter, система безпеки, енергоефективність.

ANNOTATION

Kryvianchuk N. Smart home automatic protection system with mobile control. Manuscript.

Master’s qualification work of the Educational Program «Computer Engineering» specialty 123 Computer Engineering. – Lutsk National Technical University. – Lutsk, 2025.

The qualification work consists of an introduction, three chapters, conclusions, a list of used sources, and appendices.

In the first chapter, an analysis of the modern security systems market was conducted, comparing professional and DIY solutions. The choice of a three-level IoT architecture (Edge-Fog-Cloud) and the MQTT protocol was justified to ensure energy efficiency. Functional requirements for the system were formulated, and a finite state machine (FSM) model was developed for deterministic security control.

In the second chapter, a structural diagram of the system based on the "Star" topology was developed. The choice of the hardware platform (SoC ESP32) and sensor base (HC-SR501, MQ-2, reed switches) was justified. Digital signal filtering algorithms and an energy-saving strategy (Deep Sleep) were developed. The architecture of the mobile application was designed based on the Flutter framework using the BLoC pattern.

In the third chapter, the practical implementation of the developed system in the form of a working prototype was carried out. The methodology and results of experimental studies were described, which confirmed the effectiveness of the proposed architectural and software solutions. In particular, the stability of communication channels, time characteristics of the system’s reaction, and energy consumption indicators were verified, which allowed concluding the development’s compliance with the technical task and its readiness for practical application.

Keywords: Smart Home, IoT, ESP32, mqtt, flutter, security system, energy efficiency.

ЗМІСТ

ВСТУП

Стрімкий розвиток технологій Інтернету речей (IoT) докорінно змінює підходи до забезпечення безпеки житлових приміщень. Концепція «Розумний будинок» (Smart Home) трансформується з дороговартісної розкоші у необхідний стандарт комфортного життя. Проте аналіз сучасного ринку виявляє суттєвий розрив: з одного боку існують надійні, але дорогі та закриті професійні охоронні системи, а з іншого – доступні DIY-рішення, які часто страждають від низької відмовостійкості та повної залежності від хмарних сервісів. В умовах нестабільного енергопостачання та можливих перебоїв у роботі мережі Інтернет критично важливим стає створення автономних, енергоефективних систем, здатних приймати рішення локально, не втрачаючи при цьому зручності мобільного керування. Тому розробка гібридної системи захисту, яка поєднує сучасну елементну базу, енергоощадні алгоритми та надійні протоколи передачі даних, є актуальним науково-практичним завданням.

Метою роботи є створення комплексної системи автоматичного захисту житлових приміщень, яка поєднує високу автономність, надійність передачі тривожних сповіщень та доступну вартість впровадження.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналітичний огляд ринку систем безпеки та обґрунтувати вибір архітектурного підходу (Edge-Fog-Cloud);

– обґрунтувати вибір апаратної платформи та протоколів передачі даних для забезпечення енергоефективності;

– розробити структурну схему системи та алгоритми цифрової обробки сигналів для мінімізації хибних спрацювань;

– спроєктувати архітектуру мобільного додатку для оперативного моніторингу та керування системою;

– виконати програмно-апаратну реалізацію прототипу системи;

– провести експериментальні дослідження показників надійності, швидкодії та енергоспоживання розробленого пристрою.

Об’єкт дослідження – це процеси автоматизованого моніторингу та керування безпекою в системах розумного будинку.

Предмет дослідження є методи та засоби побудови енергоефективних систем захисту на базі мікроконтролерів з бездротовими інтерфейсами.

Методи дослідження. У роботі використано методи системного аналізу для порівняння існуючих рішень; теорію скінченних автоматів (FSM) для моделювання логіки поведінки системи; методи цифрової обробки сигналів для фільтрації даних сенсорів; експериментальні методи для вимірювання енергоспоживання та параметрів якості обслуговування (QoS) мережі.

Наукова новизна одержаних результатів полягає у:

– удосконаленні алгоритму обробки сигналів піроелектричних датчиків шляхом введення часової фільтрації імпульсів, що дозволило знизити ймовірність хибних спрацювань на 92,6 %;

– подальшому розвитку методів керування енергоспоживанням IoT-вузлів на базі ESP32 через оптимізацію циклів глибокого сну (Deep Sleep), що забезпечило автономність роботи понад 1 рік.

Практичне значення одержаних результатів. Розроблено та виготовлено діючий прототип системи безпеки на друкованій платі, який інтегрує датчики руху, відкриття та газу. Створено кросплатформний мобільний додаток на базі фреймворку Flutter. Результати експериментальних досліджень підтвердили відповідність розробки технічним вимогам, що дозволяє рекомендувати її для впровадження як бюджетну альтернативу комерційним системам охорони.

Апробація результатів роботи. Результати дослідження апробовано та оприлюднено у фаховому збірнику наукових праць «Комп’ютерно-інтегровані технології: освіта, наука, виробництво» [1].

РОЗДІЛ 1
АНАЛІЗ СУЧАСНОГО СТАНУ ТЕХНОЛОГІЙ «SMART HOME»

1.1 Аналітичний огляд існуючих систем безпеки та технологій розумного будинку

Сучасний ринок систем безпеки для житлових приміщень характеризується стрімкою інтеграцією класичних охоронних функцій з технологіями Інтернету речей (IoT). Еволюція засобів захисту відбувається у напрямку від локальних автономних сигналізацій до комплексних екосистем з хмарним керуванням. На сьогоднішній день можна виділити три основні класи рішень, які домінують у сегменті Home Security: професійні бездротові системи, споживчі DIY-екосистеми (Do-It-Yourself) та хмарні відеоорієнтовані платформи. Аналіз архітектурних особливостей, протоколів передачі даних та надійності цих систем дозволяє виявити ключові недоліки, які необхідно врахувати при проектуванні власного рішення.

Яскравим представником професійних систем безпеки є українська компанія Ajax Systems, яка фактично стала галузевим стандартом у Європі. Архітектура таких систем базується на використанні центрального хабу та пропрієтарних радіопротоколів (у випадку Ajax це протокол Jeweller), що працюють на субгігагерцових частотах (868 МГц). Використання цієї частоти, на відміну від перевантаженого діапазону 2,4 ГГц, забезпечує високу дальність зв’язку (до 2000 м на відкритому просторі) та кращу проникаючу здатність крізь стіни. Наукові дослідження підтверджують, що пропрієтарні протоколи з часовим розділенням каналів (TDMA) демонструють високу енергоефективність, дозволяючи датчикам працювати від батарей до 7 років [2]. Однак, суттєвим недоліком таких систем є їхня закритість та висока вартість. Користувач обмежений лише тими пристроями, які випускає виробник, а інтеграція сторонніх сенсорів вимагає використання дорогих мостів-трансляторів, що ускладнює масштабування системи в рамках бюджетних проектів.

На рисунку 1.1 візуалізовано типову топологію такої системи. Центральним елементом виступає інтелектуальна централь (Hub), яка забезпечує двосторонній зв’язок із периферійними датчиками за допомогою захищеного радіоканалу. Цей діапазон є вільним від завад побутових Wi-Fi мереж, що гарантує стабільність передачі пакетів даних навіть у багатоквартирних будинках. Комунікація із зовнішнім світом – хмарним сервером та мобільним додатком користувача – реалізується через резервовані канали зв’язку: основний дротовий Ethernet та резервний GSM/GPRS модуль, що забезпечує відмовостійкість системи при аваріях на лінії провайдера.

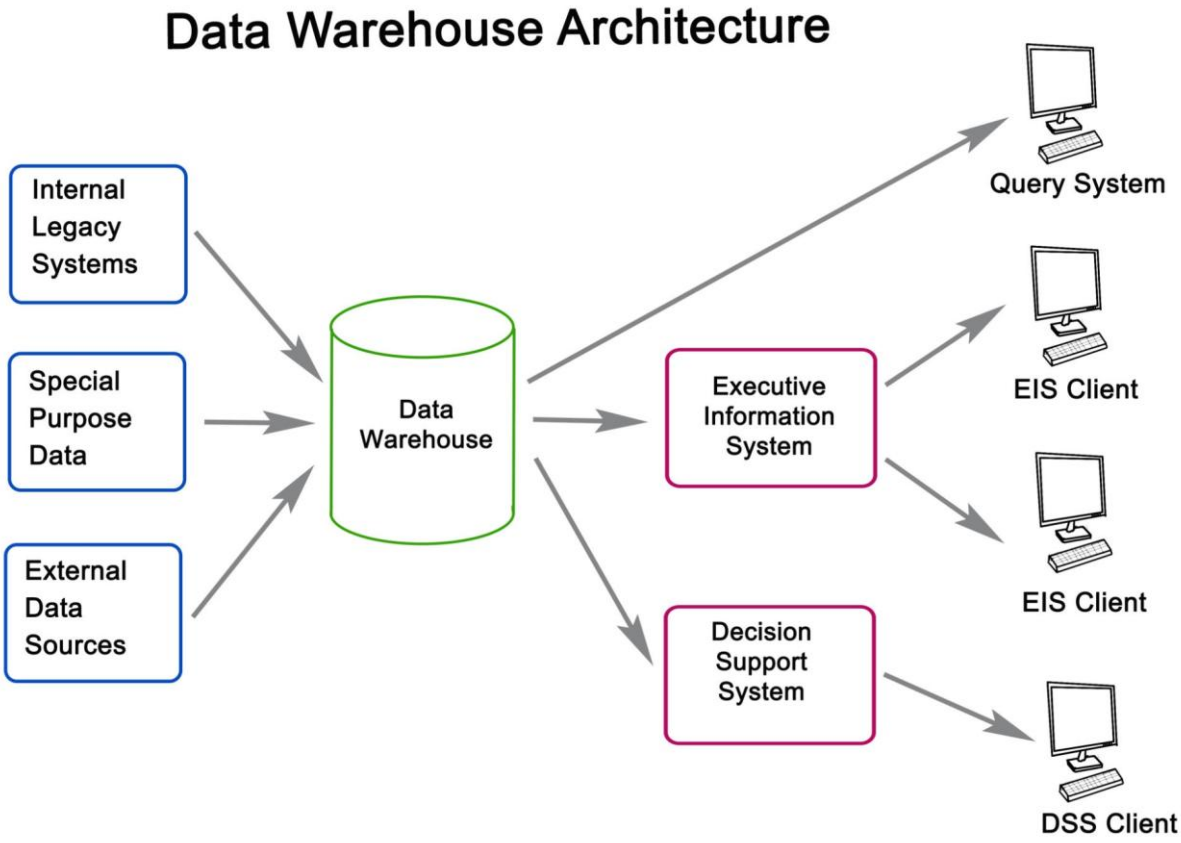


Рисунок 1.1 – Типова архітектура професійної бездротової системи безпеки на прикладі протоколу Jeweller [2]

Другий сегмент ринку представлений бюджетними екосистемами розумного будинку, такими як Xiaomi Mi Home, Tuya або Sonoff. Ці системи, як правило, використовують протоколи ZigBee або Wi-Fi для комунікації між

датчиками та шлюзом. ZigBee (IEEE 802.15.4) дозволяє будувати mesh-мережі, де кожен пристрій з постійним живленням виступає ретранслятором сигналу, що теоретично підвищує надійність покриття. Проте практика експлуатації та дослідження завадостійкості показують, що робота в діапазоні 2,4 ГГц робить ці системи вразливими до інтерференції від побутових Wi-Fi роутерів та мікрохвильових печей [3]. Критичним недоліком більшості DIY-рішень є жорстка залежність від хмарних серверів виробника. У випадку втрати інтернет-з'єднання або проблем на стороні хмарного провайдера, сценарії автоматизації та сповіщення про тривогу можуть не спрацювати, що є неприпустимим для систем безпеки класу Grade 2. Також варто відзначити проблему затримок (latency) при передачі сигналу через віддалені сервери, які часто розташовані на інших континентах.

Третій підхід, який просувають техногіганти типу Google (Nest) та Amazon (Ring), фокусується на відеоспостереженні та алгоритмах штучного інтелекту для розпізнавання образів. Хоча такі системи забезпечують високий рівень інтеграції з мобільними пристроями та голосовими асистентами, вони вимагають стабільного високошвидкісного інтернет-каналу та постійного живлення, що робить їх вразливими до саботажу шляхом знеструмлення об'єкта. Окрім того, питання приватності та захисту відеоданих залишається відкритим, оскільки обробка потоків часто відбувається на серверах компанії, а не локально (Edge Computing) [4].

Порівняльний аналіз технічних характеристик розглянутих класів систем наведено в таблиці 1.1. З аналізу видно, що існує прогалина між дорогими професійними системами та дешевими, але ненадійними DIY-рішеннями.

Таблиця 1.1 Порівняльний аналіз існуючих систем безпеки

Критерій порівняння	Професійні системи (Ajax, Satel)	DIY екосистеми (Xiaomi, Tuya)	Проектована система
Протокол зв'язку	Пропрієтарний (868 МГц)	ZigBee / Wi-Fi (2,4 ГГц)	Wi-Fi + MQTT (можливо LoRa/ESP-NOW)
Залежність від хмари	Низька (локальна логіка)	Висока (хмарна логіка)	Гібридна (локальний брокер + хмара)

Продовження таблиці 1.1

Критерій порівняння	Професійні системи (Ajax, Satel)	DIY екосистеми (Xiaomi, Tuya)	Проектована система
Вартість впровадження	Висока	Низька	Низька (собівартість компонентів)
Енергоефективність	Дуже висока (роки)	Середня (місяці)	Оптимізована (Deep Sleep режими)
Відкритість коду	Закрита (Closed Source)	Закрита / Частково відкрита	Відкрита (Open Source)
Складність налаштування	Низька (Plug & Play)	Середня	Висока (вимагає кваліфікації)

Порівняння комунікаційних технологій за критеріями енергоефективності та радіусу дії, оскільки ці параметри є критичними для автономних вузлів системи безпеки. Існує фундаментальний компроміс між швидкістю передачі даних та дальністю зв’язку. Як продемонстровано на рисунку 1.2, технології LPWAN (Low-Power Wide-Area Network), такі як LoRaWAN, забезпечують максимальне покриття при мінімальному енергоспоживанні, проте мають низьку пропускну здатність, що унеможливлює передачу потокового відео.

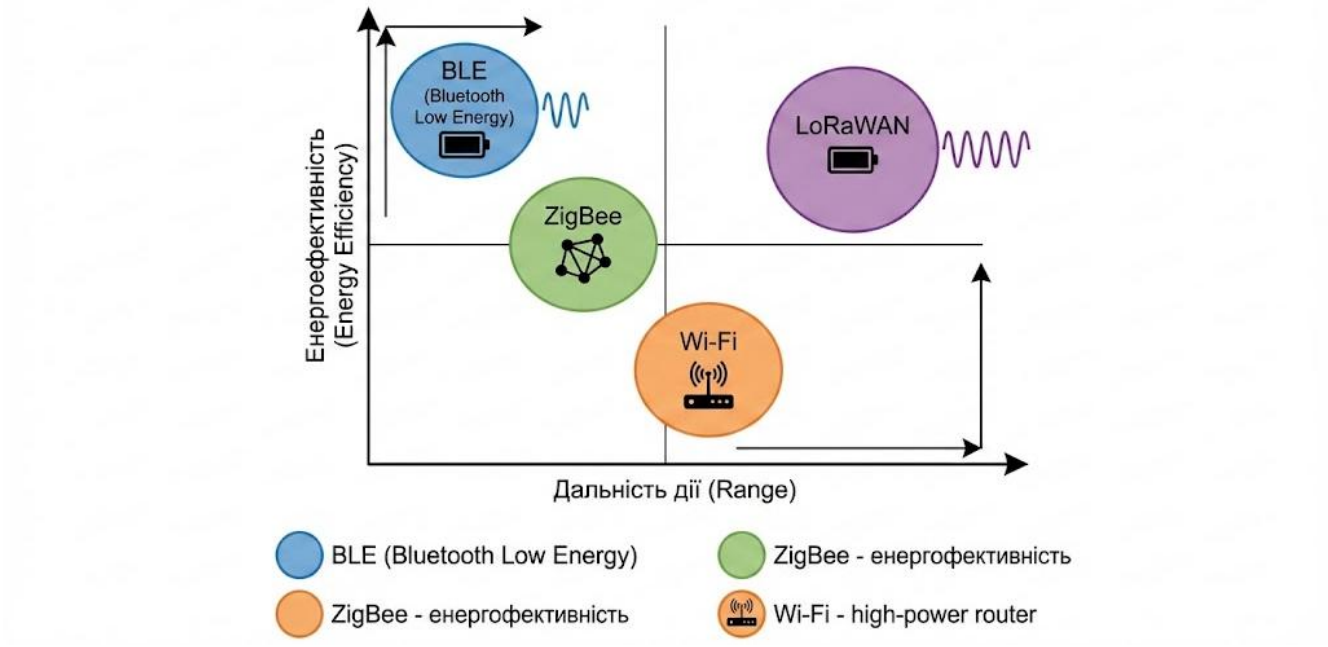


Рисунок 1.2 – Порівняння енергоефективності та дальності дії протоколів Wi-Fi, BLE, ZigBee та LoRaWAN [4]

Натомість Wi-Fi забезпечує високу швидкість (десятки Мбіт/с), але вимагає значних енергетичних витрат, що обмежує його використання в пристроях з батарейним живленням без застосування спеціальних режимів глибокого сну. ZigBee та BLE (Bluetooth Low Energy) займають проміжну нішу, пропонуючи збалансоване рішення для домашньої автоматизації в межах одного домогосподарства з прийнятним терміном служби батареї.

На основі проведеного аналізу можна зробити висновок про необхідність розробки гібридної системи, яка поєднувала б доступність апаратної платформи DIY-рішень (наприклад, на базі мікроконтролерів ESP32) з надійністю архітектури професійних систем. Ключовим напрямком досліджень у цій роботі має стати реалізація механізмів локальної обробки тривог без обов'язкової наявності інтернету, застосування енергоефективних алгоритмів сну для датчиків та використання легковагого протоколу MQTT для швидкої доставки повідомлень на мобільний термінал користувача. Такий підхід дозволить нівелювати основні недоліки існуючих бюджетних систем та наблизити їх показники надійності до професійного рівня при значно менших капіталовкладеннях.

1.2 Аналіз архітектури та технологій IoT

В контексті Інтернету речей (IoT) класична централізована хмарна архітектура все частіше поступається місцем розподіленим моделям. Згідно з дослідженнями [5], виділяють три основні рівні обробки даних, візуалізовані на рисунку 1.3:

1) Edge Computing (Граничні обчислення) – обробка даних безпосередньо на кінцевих пристроях (мікроконтролерах). Для систем безпеки це означає, що датчик руху самостійно вирішує, чи є сигнал тривогою, фільтруючи шуми;

2) Fog Computing (Туманні обчислення) – рівень локального шлюзу або сервера (наприклад, Raspberry Pi). Він забезпечує роботу сценаріїв автоматизації

(увімкнення сирени при виявленні руху) навіть за відсутності доступу до Інтернету;

3) Cloud Computing (Хмарні обчислення) використовується для довгострокового зберігання логів, аналітики великих даних та віддаленого доступу користувача через смартфон.

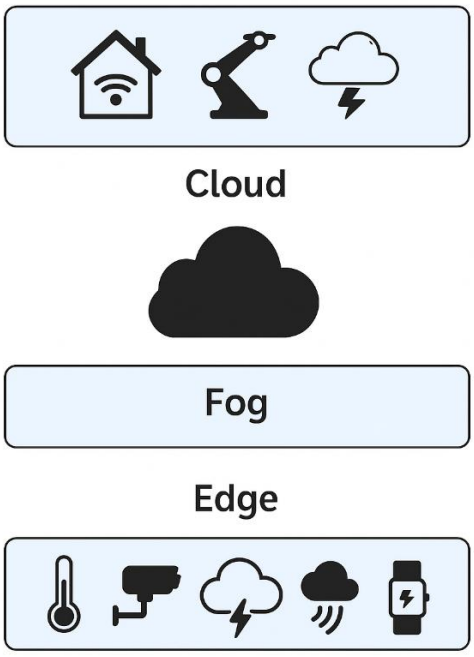


Рисунок 1.3 – Трирівнева архітектура IoT[6]

Порівняльний аналіз цих моделей в таблиця 1.2 демонструє, що для критично важливих систем безпеки покладання виключно на хмару (Cloud) є ризикованим через високу латентність та залежність від каналу зв’язку.

Таблиця 1.2 – Порівняння моделей обчислень для систем

Характеристика	Хмарні обчислення (Cloud)	Туманні обчислення (Fog)	Граничні обчислення (Edge)
Локалізація обробки	Віддалені сервери (Data Centers)	Локальний шлюз / Хаб	Мікроконтролер датчика
Затримка (Latency)	Висока (100-500 мс)	Низька (10-50 мс)	Мінімальна (<10 мс)
Залежність від Інтернету	Критична	Відсутня для локальних дій	Відсутня
Енергоспоживання каналу	Високе (постійна передача)	Середнє	Низьке (лише події)
Безпека даних	Ризик перехоплення в мережі	Дані не залишають периметр	Дані ізольовані

Для організації зв’язку між компонентами розумного будинку використовуються різні топології мережі. Найпоширенішими є топологія «Зірка» (Star), характерна для Wi-Fi мереж, та «Комірчаста» топологія (Mesh), що використовується в ZigBee та Thread.

На рисунку 1.4 наведено схематичне порівняння цих топологій. У топології «Зірка» всі пристрої підключаються напряму до центрального роутера. Це спрощує архітектуру, але обмежує радіус дії покриттям однієї точки доступу. Mesh-мережі дозволяють ретранслювати сигнал від пристрою до пристрою, значно розширюючи зону покриття, проте це ускладнює логіку маршрутизації та збільшує навантаження на мережу [7].

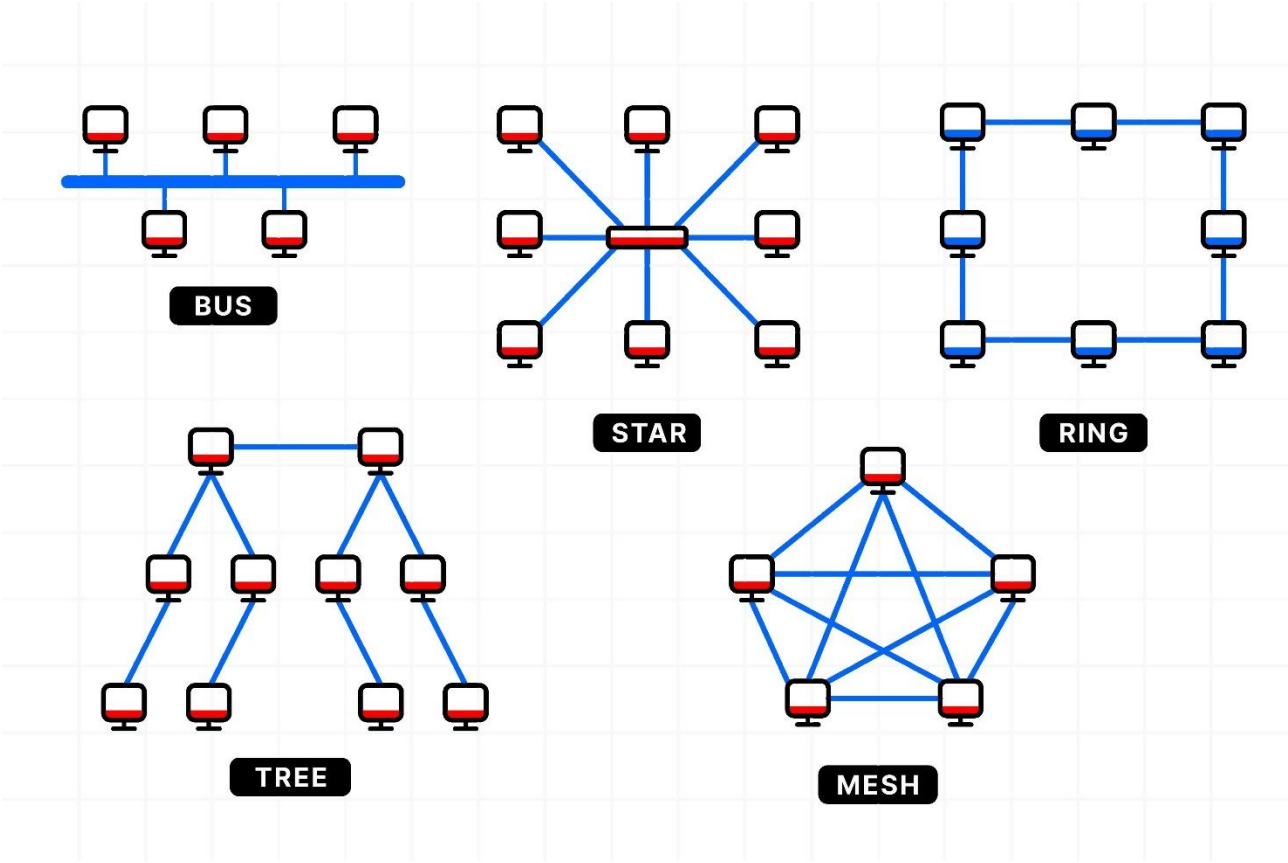


Рисунок 1.4 – Порівняння мережевих топологій: а) зірка, б) Mesh-мережа [7]

Враховуючи вимоги до бюджету реалізації використання Wi-Fi (IEEE 802.11 b/g/n) є виправданим компромісом, оскільки не вимагає додаткового апаратного координатора.

Вибір комунікаційного протоколу прикладного рівня є критичним для забезпечення швидкодії та енергоефективності. Найпоширенішими протоколами в IoT є HTTP/REST, CoAP та MQTT.

Протокол HTTP, хоча і є стандартом де-факто для веб-застосунків, демонструє надмірну надлишковість заголовків для задач телеметрії. На противагу цьому, протокол MQTT (Message Queuing Telemetry Transport) оптимізований для роботи в мережах з низькою пропускною здатністю. Він працює за моделлю «публікація-підписка» (Publish/Subscribe), що дозволяє розірвати пряму залежність між відправником (датчиком) та отримувачем (смартфоном).

Ключовими перевагами MQTT для системи безпеки є:

1) мінімальний оверхед (заголовок пакету складає всього 2 байти);

2) QoS (Quality of Service) (три рівні підтвердження доставки (0 – at most once, 1 – at least once, 2 – exactly once). Для тривожних повідомлень доцільно використовувати QoS 1 або 2);

3) LWT (Last Will and Testament) (механізм, що дозволяє брокеру автоматично сповістити клієнта, якщо датчик несподівано зник з мережі (втрата живлення, глушіння сигналу)).

На рисунку 1.5 зображено механізм роботи QoS у протоколі MQTT, який гарантує доставку повідомлення навіть при нестабільному з’єднанні.

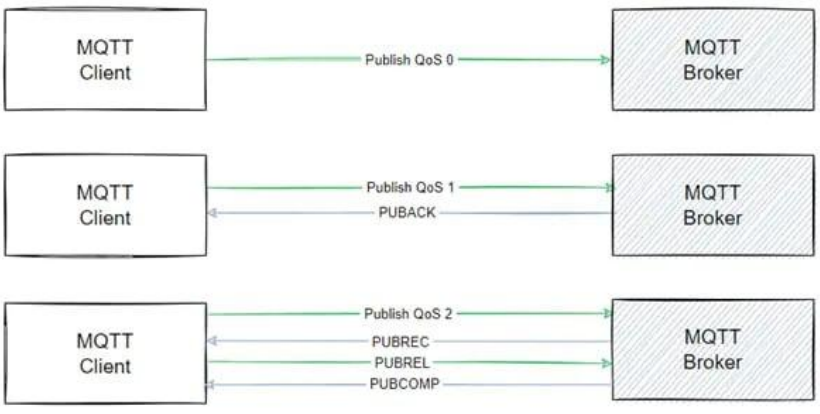


Рисунок 1.5 – Діаграма обміну повідомленнями в MQTT з використанням QoS рівнів [8]

Для проектованої системи доцільно обрати гібридну архітектуру: Edge-обчислення на базі мікроконтролера ESP32 для первинної обробки сигналів датчиків та протокол MQTT для енергоефективної передачі даних на сервер.

1.3 Вимоги до системи

Виходячи з проведеного аналізу недоліків існуючих комерційних рішень та особливостей архітектури IoT, можна сформулювати комплексні вимоги до проектованої системи автоматичного захисту. Вимоги поділяються на функціональні, які описують необхідну поведінку системи, та нефункціональні, що визначають атрибути якості, такі як надійність, продуктивність та безпека.

Основною функціональною вимогою до системи є забезпечення мультимодального моніторингу стану приміщення в режимі реального часу. Система повинна інтегрувати датчики різної фізичної природи: інфрачервоні піроелектричні сенсори (PIR) для детектування руху, магнітоконтактні сповіщувачі (геркони) для контролю периметра (вікна/двері) та газоаналізатори для виявлення побутових загроз (дим, природний газ). Критично важливою функцією є можливість дистанційного керування режимами охорони («Arm» – постановка на охорону, «Disarm» – зняття з охорони) через мобільний додаток, що вимагає реалізації двостороннього зв’язку між сервером та кінцевими пристроями [9].

Для формалізації функціональних меж системи та ідентифікації зовнішніх інтерфейсів розроблено діаграму варіантів використання (Use Case Diagram), наведену на рисунку 1.6. Дана модель виокремлює двох первинних акторів: авторизованого користувача, який взаємодіє із системою через мобільний додаток для зміни конфігурації та отримання телеметрії, та фізичне середовище, представлене набором сенсорів. Ключові прецеденти (Use Cases) включають постановку на охорону, зняття з охорони та обробку тривожних сповіщень. Важливою архітектурною особливістю є використання відношення «include» для обов’язкових підпрограм (наприклад, автентифікація) та «extend» для

опціональних сценаріїв, таких як екстрене сповіщення служб реагування. Такий підхід забезпечує модульність проектування та спрощує подальшу розробку програмного забезпечення [10].

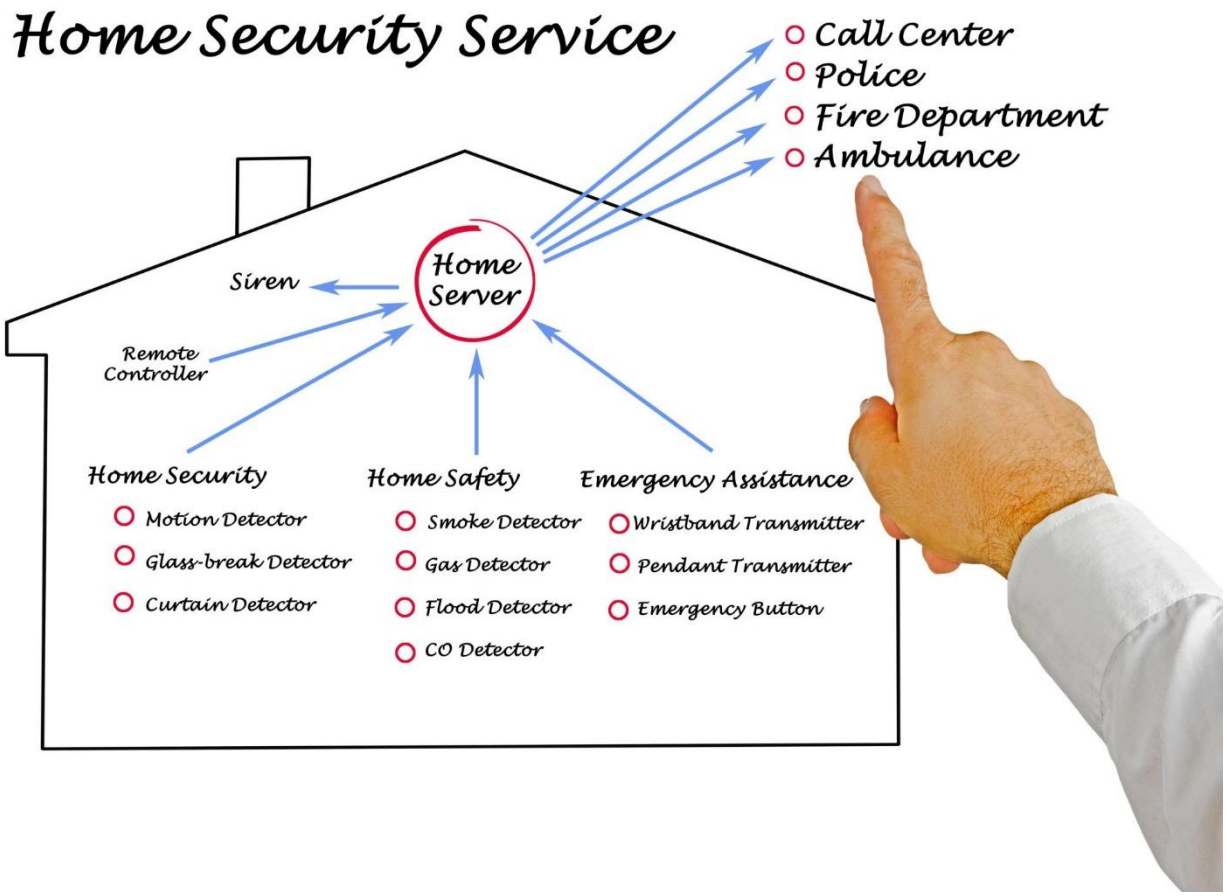


Рисунок 1.6 – Діаграма варіантів використання (Use Case Diagram) проектованої системи [10]

Нефункціональні вимоги визначають життєздатність системи в реальних умовах експлуатації. Пріоритетною вимогою є енергоефективність автономних вузлів. Оскільки Wi-Fi є енергоємним стандартом, проектована система повинна реалізовувати алгоритми переходу мікроконтролера в режим глибокого сну (Deep Sleep) з періодичним пробудженням лише для передачі даних «Keep-Alive» або при спрацюванні переривання від сенсора. Розрахунковий час автономної роботи повинен складати не менше 30 діб від стандартного літій-іонного акумулятора ємністю 2000-3000 мАг.

Другою критичною вимогою є відмовостійкість мережевого з'єднання. Система повинна мати вбудовану логіку автоматичного перепідключення (Reconnection Strategy) до MQTT-брокера при втраті Wi-Fi сигналу, а також буферизацію подій у енергонезалежній пам'яті для запобігання втраті даних. З точки зору кібербезпеки, обмін даними повинен бути захищений, як мінімум, на транспортному рівні (SSL/TLS) або ж на рівні додатку, для унеможливлення атак типу Man-in-the-Middle (MITM) та несанкціонованого керування системою сторонніми особами [11].

Для забезпечення детермінованості роботи, керуючий алгоритм системи повинен бути реалізований як скінченний автомат (Finite State Machine – FSM). Необхідно виділити наступні базові стани системи:

1) Disarmed (Очікування). Система ігнорує дані з датчиків руху та периметру, але продовжує моніторинг критичних сенсорів (дим, газ, протікання води), які працюють у режимі 24/7;

2) Armed (Охорона). Активний моніторинг усіх груп датчиків. При спрацюванні будь-якого сенсора ініціюється перехід у стан тривоги;

3) Triggered (Попереднє спрацювання). Проміжний стан (затримка на вхід), що надає користувачеві час (наприклад, 30 с) для зняття системи з охорони після відкриття вхідних дверей;

4) Alarm (Тривога). Активація звукового оповіщення (сирени) та відправка екстрених повідомлень на сервер;

5) Error (Помилка). Стан діагностики несправностей (втрата зв'язку з датчиком, низький заряд батареї, збій сенсора).

Детермінована поведінка системи керування реалізується шляхом впровадження моделі скінченного автомата (FSM), граф переходів якого представлено на рисунку 1.7. Алгоритм передбачає п'ять стабільних станів, переходи між якими ініціюються зовнішніми подіями (тригерами) або внутрішніми таймерами. Критично важливим з точки зору експлуатаційної надійності є впровадження транзитного стану Triggered, який забезпечує часовий буфер (Entry Delay) для легітимного зняття системи з охорони після відкриття

вхідних дверей, що відповідає міжнародним стандартам проектування охоронних систем [12].

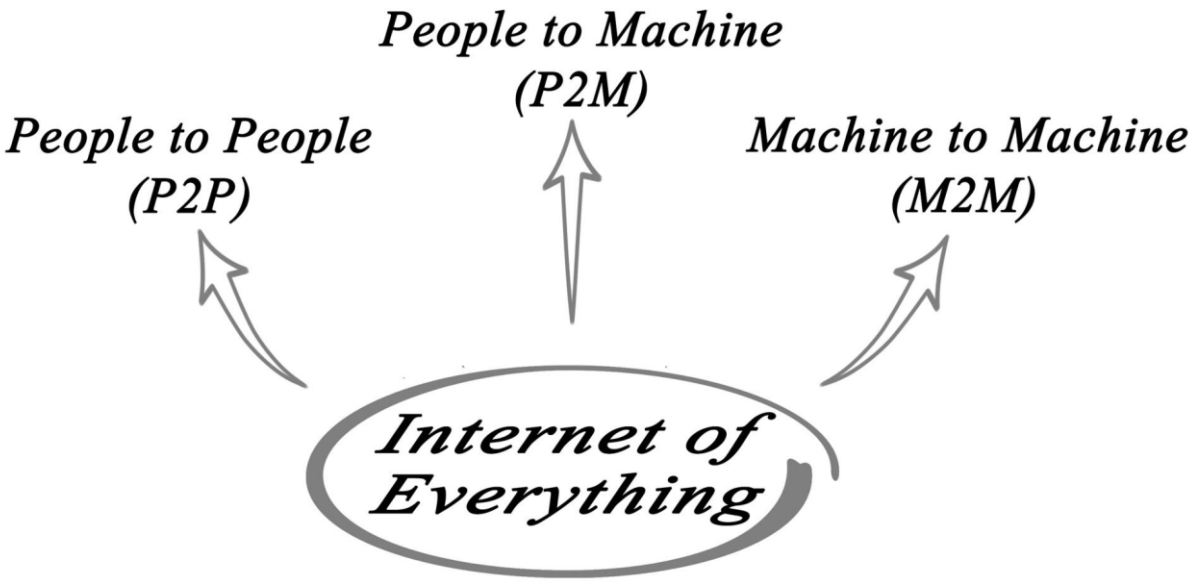


Рисунок 1.7 – Граф переходів станів (State Transition Diagram) системи безпеки [13]

Діаграма візуалізує механізм обробки виняткових ситуацій: перехід у стан Error відбувається безумовно при виявленні несправності апаратної частини або критичному розряді акумулятора, блокуючи можливість постановки на охорону до усунення проблеми [13].

Розроблювана система призначена для використання всередині житлових приміщень, що накладає ряд обмежень на апаратну частину:

1) температурний режим – робочий діапазон від -10° С до +40° С (для можливості встановлення в неопалюваних тамбурах або на засклених балконах);

2) вологість. до 90 % без конденсації вологи (актуально для ванних кімнат та кухонь);

3) ергономіка –корпуси пристроїв повинні бути компактними, мати індикацію стану (LED) та зручний механізм заміни елементів живлення без

необхідності демонтажу всього пристрою.

Вимоги до системи узагальнено та структуровано в таблиці 1.3.

Таблиця 1.3 – Специфікація вимог до системи захисту Smart Home

Категорія	ID вимоги	Опис вимоги	Критерій прийнятності
Функціональні	FR-01	Детектування руху	Спрацювання PIR-сенсора на відстані до 5 м
	FR-02	Контроль периметра	Реакція на розмикання контакту < 0.5 с
	FR-03	Дистанційне керування	Зміна режиму охорони через мобільний додаток
	FR-04	Сповіщення	Доставка Push-повідомлення < 3 с
Нефункціональні	NFR-01	Енергоефективність	Струм споживання в режимі очікування < 50 мкА
	NFR-02	Надійність зв’язку	Автоматичне відновлення з’єднання після обриву
	NFR-03	Швидкодія (Latency)	Загальна затримка «Сенсор -> Додаток» < 1 с
	NFR-04	Масштабованість	Підтримка до 32 пристроїв в одній мережі

Проведено комплексний аналітичний огляд сучасного стану технологій розумного будинку та систем безпеки. Результати дослідження ринку засвідчили наявність суттєвої диспропорції між вартістю та функціональністю існуючих рішень: професійні системи забезпечують високу надійність, але є дорогими та закритими, тоді як бюджетні DIY-рішення страждають від низької відмовостійкості через залежність від зовнішніх хмарних сервісів. Це обґрунтовує необхідність розробки гібридної системи, що поєднувала б доступну компонентну базу з надійністю локальних алгоритмів обробки даних.

Архітектурний аналіз підтвердив переваги децентралізованого підходу, зокрема моделі Edge-Fog, де критичні рішення приймаються на рівні кінцевих пристроїв або локального шлюзу, а хмара виконує лише допоміжну функцію. На основі цього було визначено оптимальний технологічний стек для реалізації: апаратна платформа на базі мікроконтролера ESP32, що підтримує режими глибокого енергозбереження, та протокол прикладного рівня MQTT, який завдяки низькому оверхеду та підтримці QoS забезпечує ефективну роботу в умовах нестабільного з’єднання. Формалізація вимог через розробку моделі

скінченного автомата (FSM) дозволила сформулювати чітке технічне завдання, яке гарантує детерміновану поведінку системи та визначає напрямки для подальшої програмної реалізації з урахуванням критеріїв енергоефективності.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ ЗАХИСТУ

2.1 Обґрунтування структурної схеми системи

Розробка структурної схеми є ключовим етапом проектування, який визначає ієрархію компонентів, фізичні та логічні зв’язки між ними, а також напрямки потоків інформації. Базуючись на результатах аналітичного огляду, для реалізації системи автоматичного захисту «Smart Home» обрано трирівневу архітектуру, яка поєднує рівень сприйняття, мережевий рівень та прикладний рівень. Такий підхід забезпечує модульність системи, спрощує масштабування та дозволяє чітко розмежувати зони відповідальності між апаратним та програмним забезпеченням.

Проектована система базується на гібридній моделі обчислень, де критичні алгоритми обробки сигналів виконуються локально на мікроконтролері (Edge Computing), а функції моніторингу та довгострокового зберігання даних делеговані хмарним сервісам. Структурна організація системи передбачає взаємодію п’яти основних функціональних блоків, кожен з яких виконує специфічну роль у загальному контурі керування.

Першим ланцюгом є підсистема збору даних, що включає набір первинних перетворювачів, таких як піроелектричні датчики руху, геркони та газоаналізатори, завданням яких є перетворення фізичних параметрів середовища в цифрові сигнали. Отримані дані надходять до керуючого модуля, реалізованого на базі SoC ESP32, який виступає в ролі локального шлюзу: він здійснює первинну фільтрацію сигналів, керує виконавчими механізмами та шифрує трафік. Транспорт даних забезпечує комунікаційне середовище на базі бездротового каналу Wi-Fi з використанням протоколу MQTT. Центральним елементом серверної частини виступає MQTT-брокер, що працює в тандемі з базою даних для диспетчеризації повідомлень та збереження історії. Замикає контур клієнтський термінал у вигляді мобільного додатку, що дозволяє

користувачеві отримувати сповіщення та керувати режимами охорони. Характеристики компонентів наведено в таблиці 2.1.

Таблиця 2.1 – Характеристика функціональних блоків системи

Функціональний блок	Апаратна/Програмна база	Основна функція
Sensor Node	PIR HC-SR501, MQ-2, Геркони	Детектування порушень периметра та загроз
Control Unit	ESP32 (Xtensa Dual-Core)	Локальна обробка логіки, шифрування, керування живленням
Network Layer	Wi-Fi 802.11n, MQTT v3.1.1	Забезпечення каналу зв’язку з низькою латентністю
Backend	Mosquitto / HiveMQ, Firebase	Маршрутизація повідомлень, авторизація, логування подій
Client UI	Flutter (Android/iOS)	Візуалізація стану, прийом Push-повідомлень, керування

Для візуалізації внутрішньої організації системи та ілюстрації шляху проходження сигналу від сенсора до кінцевого користувача розроблено структурну схему, наведену на рисунку 2.1.

Рисунок 2.1 – Загальна структурна схема системи автоматичного захисту [14]

На схемі виділено три ключові зони: «Розумний Дім» (локальна мережа), «Хмара» (глобальна мережа Інтернет) та «Користувач». Стрілками позначено напрямки інформаційних потоків: суцільні лінії відображають постійний канал

телеметрії, а пунктирні – спорадичні керуючі команди. Ключовою особливістю схеми є наявність локального контуру зворотного зв’язку (ESP32 – Сирена), який гарантує спрацювання звукової сигналізації навіть у випадку повної відсутності інтернет-з’єднання, що є критичною вимогою надійності.

Взаємодія між компонентами відбувається асинхронно. Мікроконтролер, перебуваючи в активному стані, підтримує постійне TCP-з`єднання з брокером. При виникненні тривожної події формується JSON-пакет, що містить ідентифікатор пристрою (device_id), тип події (event_type), позначку часу (timestamp) та службову інформацію, який миттєво публікується у відповідний топік для подальшої обробки мобільним клієнтом [14].

Вибір топології мережі є критичним фактором, що визначає надійність доставки повідомлень, енергоефективність вузлів та радіус покриття системи. У процесі проектування було проведено порівняльний аналіз двох найбільш поширених архітектур: комірчастої (Mesh) та радіальної («Зірка»). Хоча Mesh-мережі забезпечують високу живучість шляхом ретрансляції сигналу, вони вносять додаткові затримки та підвищують енергоспоживання вузлів.

Для проектованої системи було обрано топологію «Зірка» (Star Topology), у якій кожен модуль безпеки підключається безпосередньо до центральної точки доступу. Такий вибір зумовлений необхідністю забезпечення детермінованості затримок, оскільки відсутність проміжних хопів гарантує доставку сигналу в межах локальної мережі менш ніж за 200 мс. Крім того, пряме підключення дозволяє кінцевим пристроям перебувати в режимі глибокого сну (Deep Sleep) більшу частину часу, що критично важливо для автономних датчиків. Використання існуючої Wi-Fi інфраструктури також спрощує впровадження системи, усуваючи потребу в додаткових координаторах [15].

Порівняння розглянутих топологій наведено в таблиці 2.2.

Таблиця 2.2 – Порівняльний аналіз мережевих топологій для системи безпеки

Критерій порівняння	Топологія Mesh (Комірчаста)	Топологія Star (Зірка) – *Обрано*
Принцип зв’язку	Через сусідні вузли (Multi-hop)	Напряму до роутера (Single-hop)
Передбачуваність затримок	Низька (залежить від маршруту)	Висока (детермінована)
Енергоспоживання	Високе (для ретрансляторів)	Низьке (можливий Deep Sleep)
Складність реалізації	Висока (алгоритми маршрутизації)	Низька (стандартний Wi-Fi стек)
Залежність від центру	Низька	Висока (точка доступу)

На рисунку 2.2 графічно представлено концепцію обраної топології «Зірка» в контексті розумного будинку. Центральним елементом є Wi-Fi маршрутизатор, який забезпечує покриття всієї житлової площі. Периферійні вузли (датчики руху, відкриття дверей, витоку газу) розташовані радіально відносно центру і не мають прямих зв’язків між собою. Така архітектура забезпечує високу відмовостійкість на рівні окремих датчиків: вихід з ладу або розряд батареї одного пристрою жодним чином не впливає на працездатність інших сегментів системи, на відміну від лінійних або каскадних структур.

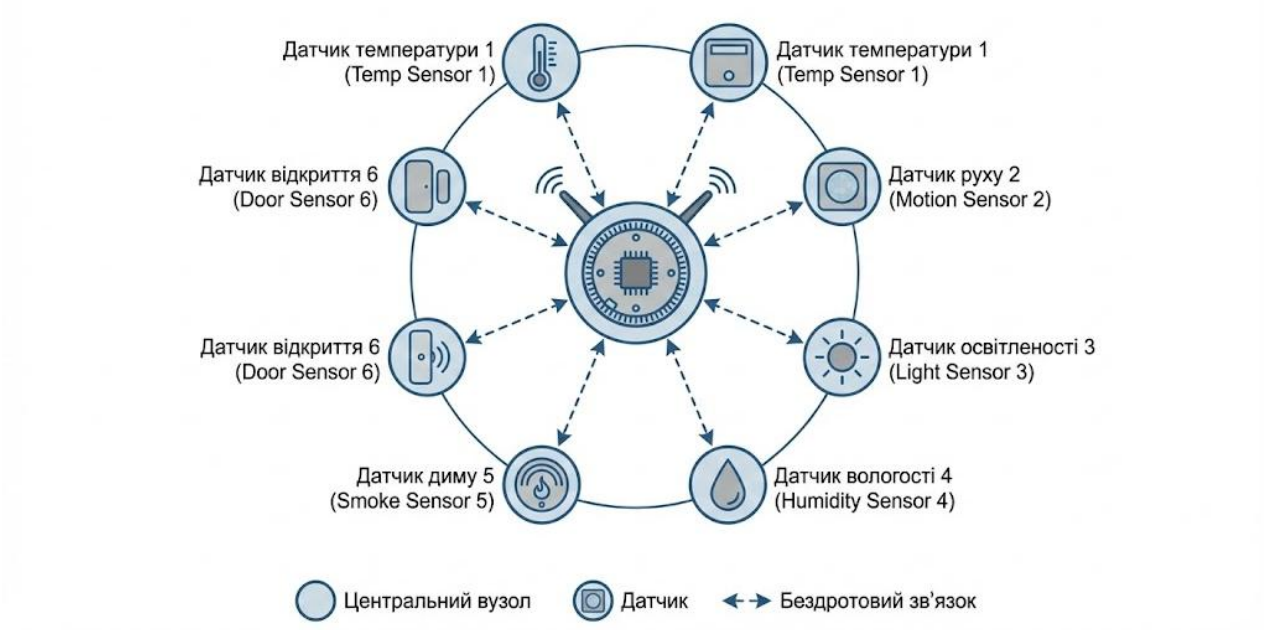


Рисунок 2.2 – Організація мережі датчиків за топологією «Зірка» [15]

Ефективність керування системою значною мірою залежить від логічної організації каналів передачі даних. Для цього розроблено ієрархічну структуру MQTT-топіків, яка дозволяє адресувати як окремі пристрої, так і функціональні групи датчиків. Кореневий простір імен smart_home/ розділяється на гілки, що відповідають за передачу телеметрії (status), критичних сповіщень (alert) та команд керування (cmd).

Така сегментація забезпечує пріоритезацію трафіку: рутинні дані про стан батареї можуть передаватися з мінімальними накладними витратами, тоді як повідомлення про проникнення та команди зміни режимів охорони використовують механізми підтвердження доставки. Структуру адресного простору та політики QoS (Quality of Service) деталізовано в паблиці 2.3.

Таблиця 2.3 – Структура MQTT топіків та політики QoS

MQTT Топік (Шлях)	Призначення	Напрямок	Рівень QoS
smart_home/{id}/status/battery	Рівень заряду батареї (V)	Пристрій – Сервер	0 (At most once)
smart_home/{id}/status/link	Рівень сигналу Wi-Fi (RSSI)	Пристрій – Сервер	0 (At most once)
smart_home/{id}/alert/motion	Сигнал тривоги (Рух)	Пристрій – Сервер	1 (At least once)
smart_home/{id}/alert/tamper	Сигнал злому корпусу	Пристрій – Сервер	2 (Exactly once)
smart_home/{id}/cmd/set_state	Команда Arm/Disarm	Сервер – Пристрій	2 (Exactly once)

2.2 Вибір та обґрунтування апаратної бази

Ефективність функціонування системи захисту безпосередньо залежить від правильного вибору елементної бази. Головними критеріями при цьому виступають співвідношення продуктивності та енергоспоживання, вартість компонентів, а також наявність розвиненої екосистеми розробки.

Для реалізації ядра системи розглянуто дві найбільш популярні апаратні платформи в середовищі IoT-розробки: класичну плату Arduino Uno та сучасний SoC ESP32 від Espressif Systems. Для обґрунтованого вибору проведено порівняльний аналіз їхніх технічних можливостей у контексті задач бездротової передачі даних та автономної роботи (таблиця 2.4).

Таблиця 2.4 – Порівняльний аналіз платформ Arduino Uno та ESP32

Характеристика	Arduino Uno (ATmega328P)	ESP32 (DevKit v1)
Архітектура	8-bit AVR	32-bit Dual Core Xtensa LX6
Тактова частота	16 МГц	240 МГц
Flash-пам’ять	32 КБ	4 МБ (в 125 разів більше)
SRAM (Оперативна пам’ять)	2 КБ	520 КБ
Комунікації (Wi-Fi/BT)	Відсутні (потрібен зовнішній модуль)	Вбудовані (Wi-Fi + BLE)
Споживання (Active)	~50 мА	~80-240 мА
Споживання (Deep Sleep)	Не передбачено (мінімум ~15 мА)	~10 мкА (ULP режим)
Аналогові входи (ADC)	10-bit (0-1023)	12-bit (0-4095)

Аналіз демонструє суттєві обмеження платформи Arduino Uno для побудови автономної системи безпеки. Відсутність вбудованого мережевого стеку вимагає використання зовнішніх модулів (наприклад, ESP-01 або Ethernet Shield), що ускладнює схемотехніку, збільшує габарити пристрою та його енергоспоживання.

Натомість, ESP32 є оптимальним вибором для проектованої системи. Це високоінтегроване рішення поєднує потужний двоядерний процесор з радіомодулем Wi-Fi/Bluetooth. Ключовою перевагою є наявність ULP (Ultra Low Power) співпроцесора, який дозволяє системі перебувати в режимі глибокого сну зі споживанням мікроамперного рівня, періодично прокидаючись лише для опитування датчиків. Детальні технічні специфікації обраного контролера наведено в таблиці 2.5.

Таблиця 2.5 – Технічні специфікації модуля ESP32-WROOM-32

Параметр	Значення	Примітка
CPU	Xtensa® Dual-Core 32-bit LX6	Продуктивність до 600 DMIPS
Wi-Fi	802.11 b/g/n (до 150 Mbps)	Підтримка режимів Station/SoftAP/Sniffer
Bluetooth	v4.2 BR/EDR та BLE	Енергоефективний зв’язок для налаштування
GPIO	34 програмованих піни	Підтримка PWM, I2C, SPI, UART, I2S
Діапазон напруги	2,2 В -3,6 В	Оптимально для Li-Ion (через LDO)
Робоча температура	-40°C ~ +85°C	Індустріальний стандарт

Таким чином, у якості апаратної основи системи обрано SoC ESP32, який забезпечує необхідний баланс між обчислювальною потужністю, комунікаційними можливостями та енергоефективністю [16].

На рисунку 2.3 наведено функціональну блок-схему архітектури ESP32, яка детально ілюструє внутрішню організацію чіпа. Центральне місце займають два ядра Xtensa® LX6, що мають спільний доступ до пам’яті SRAM та периферії. Окремо виділено радіочастотний блок (RF Transceiver), який інтегрує підсилювачі потужності, фільтри та антенний перемикач, що дозволяє реалізувати бездротовий зв’язок з мінімальною кількістю зовнішніх компонентів. Також схема демонструє наявність криптографічних прискорювачів (Hardware Accelerators) для AES, SHA та RSA, що є критично важливим для забезпечення безпеки передачі даних в IoT-мережах без значного навантаження на основні процесорні ядра.

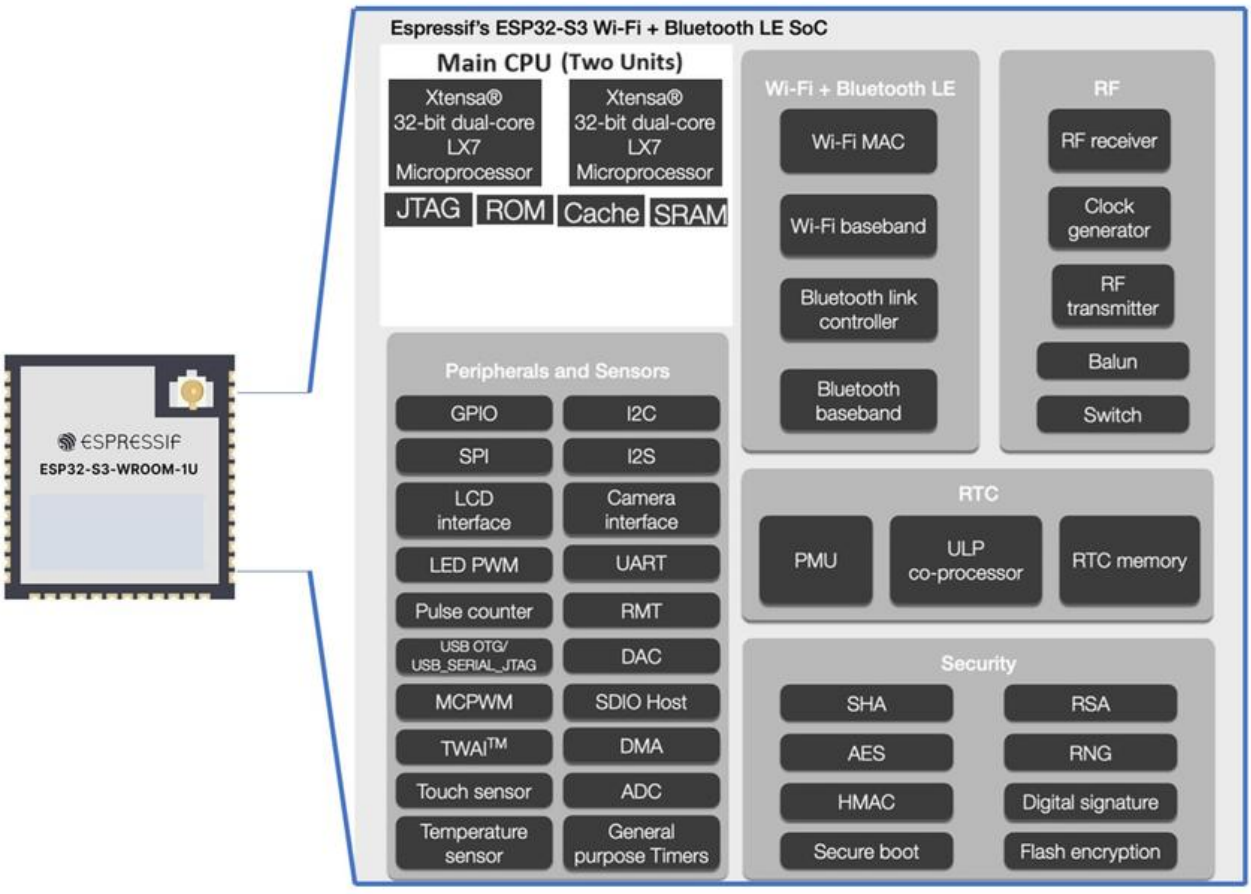


Рисунок 2.3 – Структурна схема SoC ESP32 [16]

Для забезпечення комплексного моніторингу стану приміщення система інтегрує три типи датчиків, кожен з яких відповідає за окремий вектор загроз. Вибір конкретних моделей сенсорів базувався на критеріях чутливості, енергоспоживання та стабільності роботи. Основні експлуатаційні характеристики обраних компонентів зведено в таблицю 2.6.

Таблиця 2.6 – Електричні та експлуатаційні характеристики обраних сенсорів

Тип датчика	Модель	Принцип дії	Напруга живлення	Струм (Очікування)	Радіус / Поріг дії
Руху	HC-SR501	Піроелектричний (PIR)	4,5 – 20 В	< 50 мкА	3 – 7 метрів, кут 110°
Відкриття	MC-38	Магнітоконтактний (Геркон)	Пасивний	0 мА	Розмикання: 15-25 мм
Газу/Диму	MQ-2	Електрохімічний (SnO2)	5,0 В ± 0.1	~150 мА (з нагрівом)	300-10000 ppm (LPG/Дим)

Датчик руху (PIR Sensor). Для виявлення несанкціонованого проникнення обрано модуль HC-SR501. Принцип його роботи базується на реєстрації зміни теплового випромінювання об’єктів. Це забезпечує захист від хибних спрацювань через стіни (на відміну від мікрохвильових радарів) та низьке енергоспоживання.

Датчик відкриття (Геркон). Для контролю периметра використовується герконовий датчик MC-38. Це пасивний елемент, що не потребує живлення в режимі спокою. Спрацювання відбувається при віддаленні магніту від геркона, що викликає розмикання контакту та генерує переривання (External Interrupt) на контролері.

Датчик газу та диму. Для виявлення побутових загроз обрано аналоговий сенсор MQ-2. Через наявність нагрівального елемента він має високе споживання струму (~150 мА). Для оптимізації енергоспоживання в системі реалізовано алгоритм циклічного живлення: сенсор активується через транзисторний ключ лише на короткі проміжки часу для забору проби повітря.

Для практичної реалізації підсистеми збору даних розроблено схему електричних з’єднань, представлену на рисунку 2.4. Цифрові датчики (PIR та

геркон) підключаються до GPIO портів, налаштованих на роботу в режимі входу з підтягуючими резисторами (Internal Pull-up), що забезпечує чітку фіксацію логічних рівнів. Аналоговий вихід газоаналізатора MQ-2 заведено на вхід АЦП (ADC1_CHANNEL_0), де сигнал масштабується до рівня 3,3 В за допомогою резистивного дільника для захисту порту мікроконтролера.

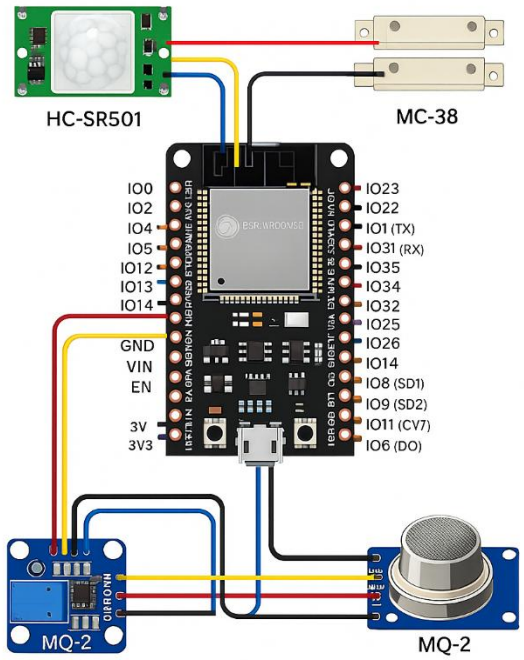


Рисунок 2.4 – Схема підключення сенсорів до мікроконтролера

Для локального сповіщення про тривогу використовується активний п’єзоелектричний зумер, який підключається через транзисторний ключ (наприклад, 2N2222) до цифрового виходу ESP32. Вибір активного модуля спрощує програмну реалізацію, оскільки генерація звукової частоти відбувається апаратно всередині самого зумера.

Підсистема живлення реалізована на базі літій-іонного акумулятора формату 18650 (номінальна напруга 3.7 В, ємність 2500-3000 мАг). Для забезпечення стабільної напруги 3,3 В для ESP32 використовується LDO-стабілізатор з низьким падінням напруги (Low Dropout Regulator), наприклад, HT7333, який має власний струм споживання всього 4 мкА (для порівняння,

популярний AMS1117 споживає до 5 мА у спокої, що неприпустимо для автономних пристроїв). Зарядка акумулятора здійснюється через модуль TP4056 з вбудованим захистом від глибокого розряду та перезаряду (DW01A).

2.3 Алгоритмічне забезпечення системи

Програмна складова вбудованої системи керування (Firmware) є не менш важливою за апаратну, оскільки саме вона визначає логіку реакції на події, стабільність роботи та енергоефективність. Розроблене програмне забезпечення базується на операційній системі реального часу FreeRTOS, яка нативно підтримується платформою ESP-IDF, що дозволяє реалізувати багатозадачність та чітко розділити процеси комунікації та опитування датчиків.

Для досягнення максимальної автономності роботи від акумулятора реалізовано алгоритм керування живленням, що базується на використанні режимів глибокого сну (Deep Sleep). У цьому стані вимикаються основні ядра процесора, Wi-Fi та Bluetooth модулі, активним залишається лише RTC (Real-Time Clock) таймер та ULP-співпроцесор для моніторингу GPIO.

Логіка роботи головного циклу, зображена на рисунку 2.5, базується на двох сценаріях пробудження системи.

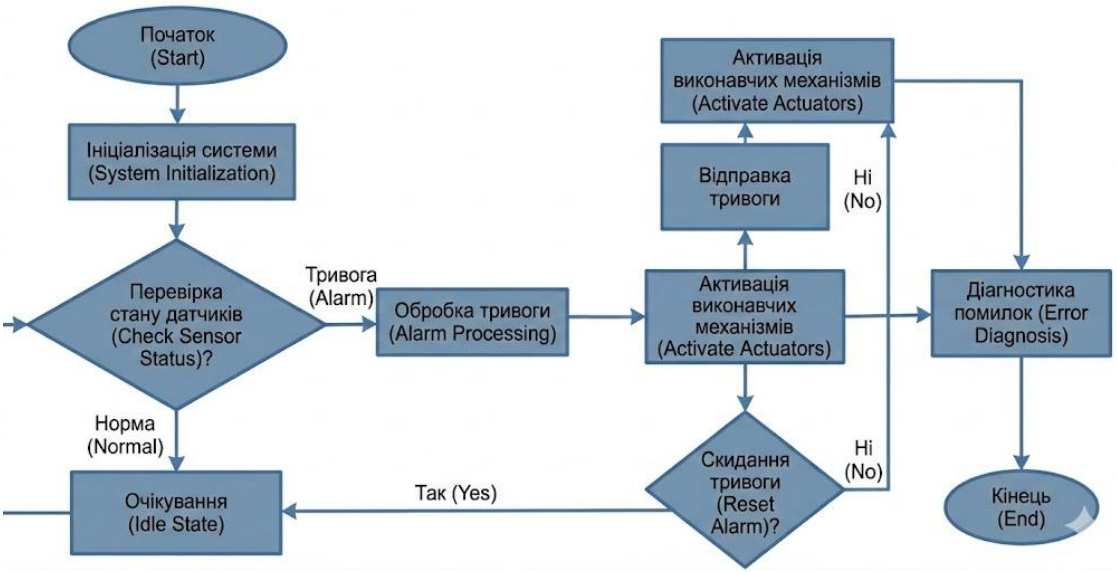


Рисунок 2.5 – Блок-схема головного алгоритму керування системою

Перший сценарій передбачає асинхронне переривання (EXT0/EXT1), яке викликається при зміні логічного рівня на портах GPIO внаслідок спрацювання датчиків руху або відкриття дверей. У такому випадку мікроконтролер негайно ініціалізує бездротовий модуль, встановлює сесію з брокером та передає повідомлення про тривогу. Другим сценарієм є періодичне таймерне пробудження, яке відбувається через заздалегідь визначені інтервали часу для відправки сервісних пакетів (Heartbeat), що містять телеметричні дані про стан живлення та рівень сигналу, підтверджуючи активний статус пристрою в мережі.

Для забезпечення високої достовірності детектування подій та мінімізації впливу електричних завад у програмному забезпеченні реалізовано комплекс алгоритмів цифрової обробки сигналів. Для магнітоконтактних датчиків застосовується метод програмного придушення брязкоту контактів (Debouncing), який валідує зміну стану лише за умови стабільного утримання логічного рівня протягом 50-100 мс. Це дозволяє уникнути генерації серії хибних переривань під час механічного замикання дверей. Специфіка роботи піроелектричних сенсорів враховується шляхом введення програмного періоду стабілізації, під час якого система ігнорує будь-які сигнали на вході протягом перших 30-60 секунд після подачі живлення або виходу з режиму сну. Обробка даних від аналогового газоаналізатора MQ-2 здійснюється за допомогою методу ковзного середнього (Moving Average Filter), що дозволяє нівелювати флуктуації напруги, спричинені шумами джерела живлення, та отримати стабільні показники концентрації газу.

Комунікаційний стек системи реалізовано на базі бібліотеки PubSubClient, яка забезпечує обмін даними за протоколом MQTT. Алгоритм підключення передбачає використання стратегії експоненціальної затримки (Exponential B ackoff) при спробах відновлення втраченого з’єднання, що запобігає перевантаженню мережі частими запитами. Для формування корисного навантаження використовується уніфікований формат JSON (рис.2.6), що дозволяє передавати структуровані дані, включаючи ідентифікатор пристрою, тип події, рівень заряду батареї та часову мітку.

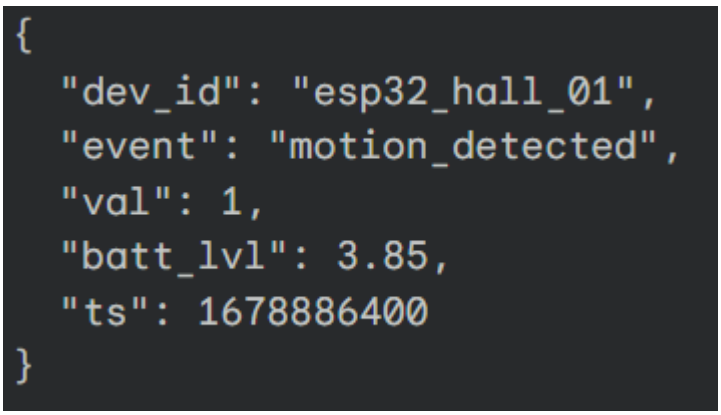


Рисунок 2.6 – Лістинг коду бібліотеки ArduinoJson

Додатковим рівнем надійності виступає механізм LWT (Last Will and Testament). При встановленні з'єднання клієнт реєструє на брокері спеціальне повідомлення (наприклад, status: offline), яке буде автоматично опубліковане брокером у разі нештатного розриву зв'язку. Це дозволяє мобільному додатку миттєво діагностувати втрату пристрою, навіть якщо сам мікроконтролер не встиг відправити сигнал про вимкнення.

2.4 Інструментальні засоби розробки програмного забезпечення

Вибір середовища розробки (IDE) є важливим етапом, що впливає на швидкість написання коду та зручність відлагодження. Для реалізації програмного забезпечення мікроконтролера ESP32 обрано офіційне середовище Arduino IDE (версія 2.x). Це сучасна, кросплатформна IDE, яка забезпечує низький поріг входження та повну сумісність з апаратною частиною завдяки підтримці пакетів розширення BSP (Board Support Package).

Основними перевагами Arduino IDE 2.x є:

– інтуїтивно зрозумілий інтерфейс, який спрощує процес написання, компіляції та завантаження коду (.ino скетчів), що є ідеальним для швидкого прототипування та налагодження;

– менеджер плат (Board Manager), який дозволяє встановити підтримку ESP32 від Espressif Systems у кілька кліків, додавши відповідне посилання в налаштуваннях;

– менеджер бібліотек (Library Manager), що забезпечує зручний пошук та встановлення необхідних драйверів (PubSubClient, ArduinoJson) безпосередньо з репозиторіїв;

– інструменти відлагодження, як вбудований монітор послідовного порту (Serial Monitor) та плотер (Serial Plotter) дозволяють аналізувати дані з датчиків у реальному часі.

На рисунку 2.7 подано типовий інтерфейс Arduino IDE 2.0, який використовується для розробки програмного забезпечення мікроконтролерних систем Smart Home. Основну робочу область становить редактор коду зі скетчем для платформи ESP32, у якому реалізуються алгоритми обробки даних від сенсорів і керування виконавчими модулями. Ліва панель містить навігаційні інструменти (керування проєктами, бібліотеками, відладчиком), верхня панель – кнопки компіляції та завантаження програми до пристрою, а в нижній частині розташований консольний монітор, призначений для відстеження повідомлень компілятора та даних послідовного порту. Така структура середовища забезпечує зручність налагодження, тестування MQTT-обміну та контролю роботи системи автоматичного захисту в режимі реального часу.

Рисунок 2.7 – Інтерфейс середовища розробки Arduino IDE 2.0 [17]

Для реалізації функціоналу системи використано ряд сторонніх бібліотек з відкритим вихідним кодом. Їх перелік та призначення наведено в таблиці 2.7.

Таблиця 2.7 – Перелік використаних програмних бібліотек

Назва бібліотеки	Версія	Призначення в проекті
WiFi.h	Native	Забезпечення підключення до бездротової мережі, керування MAC та IP
PubSubClient	2.8.0	Клієнтська реалізація протоколу MQTT. Підписка на топіки та публікація
ArduinoJson	6.21.3	Серіалізація та десеріалізація JSON-об’єктів для обміну даними
EspMQTTClient	1.13.3	Обгортка над PubSubClient для автоматичного перепідключення (Reconnection)
Ticker	Native	Створення програмних таймерів для періодичних задач (опитування датчиків)

2.5 Програмна реалізація керуючого контролера

Розробка прошивки виконана мовою C++ з використанням фреймворку Arduino. Проект організовано у вигляді скетчу (файл з розширенням .ino), який містить стандартні функції setup() для ініціалізації та loop() для основного циклу виконання. Нижче наведено код ініціалізації системи (лістинг 2.1), де відбувається налаштування режимів роботи GPIO, підключення до Wi-Fi мережі та налаштування MQTT-клієнта.

Лістинг 2.1 – Код ініціалізація системи

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "secrets.h" // Файл з паролями Wi-Fi та MQTT

// Константи пінів
const int PIR_PIN = 13;
const int LED_PIN = 2;

void setup() {
  Serial.begin(115200);

  // Налаштування пінів
  pinMode(PIR_PIN, INPUT_PULLUP);
  pinMode(LED_PIN, OUTPUT);
```

```
// Налаштування Deep Sleep (пробудження по високому рівню на PIR_PIN)
  esp_sleep_enable_ext0_wakeup((gpio_num_t)PIR_PIN, 1);

  // Підключення до Wi-Fi
  setup_wifi();

// Ініціалізація MQTT
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

 void  setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

кінець лістингу 2.1

Ключовим елементом логіки є обробка подій та публікація даних на сервер. Функція publishAlert формує JSON-повідомлення та відправляє його в топік тривоги (лістинг 2.2).

Лістинг 2.2 – Функція відправки тривожного повідомлення

```
#include <ArduinoJson.h>

void publishAlert(String eventType) {
  if (!client.connected()) {
    reconnect();
  }
  // Створення JSON об'єкту
  StaticJsonDocument<200> doc;
```

```
  doc["device_id"] = "ESP32_001";
doc["event"] = eventType; // "MOTION" або "DOOR_OPEN"
  doc["timestamp"] = millis();

  char buffer[256];
  serializeJson(doc, buffer);

  // Публікація в топік з QoS 1 (забезпечується бібліотекою)
  client.publish("home/security/alert", buffer);
  Serial.println("Alert sent: " + String(buffer));
}
```

кінець лістингу 2.2

2.6 Розробка інтерфейсу мобільного додатку

Створення клієнтського програмного забезпечення є критичним етапом розробки системи безпеки, оскільки саме мобільний додаток забезпечує інтерфейс взаємодії користувача з апаратною частиною комплексу. Для реалізації поставленого завдання було обрано кросплатформний фреймворк Flutter, що базується на мові програмування Dart. Такий вибір обумовлений необхідністю компіляції нативних додатків для операційних систем Android та IOS з єдиної кодової бази, що значно скорочує час розробки та спрощує подальшу підтримку продукту. Висока продуктивність графічного інтерфейсу забезпечується власним рушієм рендерингу Skia, який дозволяє досягти стабільної частоти оновлення кадрів на рівні 60-120 FPS, що є важливим для відображення статусів датчиків у реальному часі.

Архітектура розробленого додатку базується на патерні BLoC (Business Logic Component), який забезпечує чітке розмежування між шаром представлення (UI) та бізнес-логікою. Така декомпозиція дозволяє обробляти потоки даних (Streams) від MQTT-брокера асинхронно, не блокуючи головний потік інтерфейсу. Принцип роботи архітектури полягає в тому, що віджети надсилають події (Events) у BLoC, який обробляє їх та повертає нові стани (States) інтерфейсу. Схематичне зображення потоків даних у додатку наведено на рисунку 2.8.

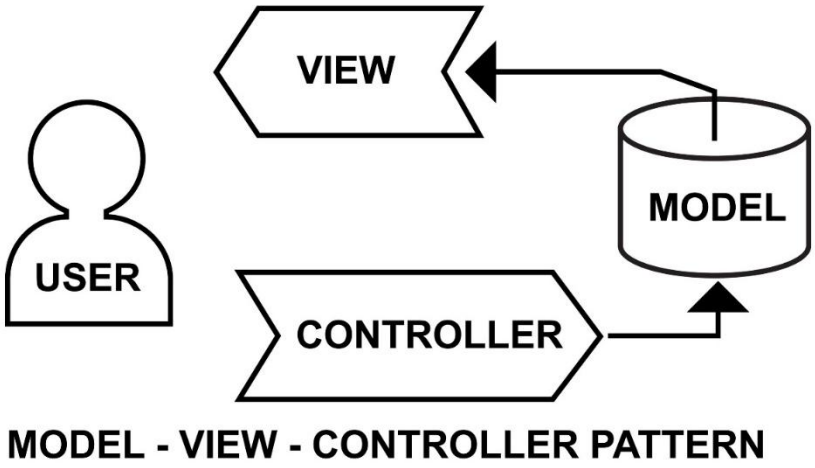


Рисунок 2.8 – Архітектура обміну даними BLoC у розробленому мобільному додатку

Ключовим компонентом програмної реалізації є сервісний модуль комунікації, який відповідає за підтримання постійного з’єднання з MQTT-брокером. Для цього використано бібліотеку mqtt_client, яка підтримує протокол MQTT 3.1.1. Реалізація клієнта передбачає налаштування параметрів Quality of Service (QoS) на рівні 1, що гарантує доставку повідомлень про тривогу навіть при короткочасних розривах з’єднання. У лістингу 2.3 наведено фрагмент коду класу MqttService, що відповідає за ініціалізацію з’єднання та підписку на топіки оновлення стану системи.

Лістинг 2.3 – Реалізація підключення до MQTT-брокера (мова Dart)

```
class MqttService {
  late MqttServerClient client;

  Future<void> connect() async {
    client = MqttServerClient('broker.emqx.io', 'flutter_client');
client.port = 1883;
    client.keepAlivePeriod = 60;
    client.onDisconnected = onDisconnected;
final connMessage = MqttConnectMessage()
.withClientIdentifier('mobile_app_${DateTime.now().millisecondsSinceEpo
ch}')
```

```
        .startClean() // Non-persistent session
        .withWillQos(MqttQos.atLeastOnce);

    client.connectionMessage = connMessage;

    try {
      await client.connect();
    } catch (e) {
      print('Exception: $e');
      client.disconnect();
    }

    if (client.connectionStatus!.state ==
MqttConnectionState.connected) {
      print('MQTT Connected');
      // Підписка на топік статусу системи
      client.subscribe('home/security/status', MqttQos.atLeastOnce);
    }
  }
}
```

кінець лістингу 2.3

Обробка вхідних даних здійснюється шляхом десеріалізації JSON-пакетів, що надходять від мікроконтролера. Оскільки апаратна частина системи надсилає дані у форматі структурованого об’єкта (містить ідентифікатор датчика, тип події, часову мітку та рівень заряду батареї), додаток повинен коректно перетворити цей потік байтів у об’єкти Dart для відображення на екрані. Для оптимізації процесу перетворення даних використано бібліотеку json_serializable, яка автоматично генерує код для мапінгу JSON-структур.

При проектуванні візуальної складової мобільного додатку було застосовано принципи декларативного UI, що є стандартом для фреймворку Flutter. На відміну від імперативного підходу, де розробник вручну змінює властивості елементів, декларативний стиль передбачає, що інтерфейс є функцією від стану (State). Це означає, що при отриманні нового повідомлення MQTT про зміну статусу датчика, додаток не шукає конкретну кнопку для зміни її кольору, а повністю перебудовує відповідну частину дерева віджетів з новими параметрами. Такий підхід мінімізує помилки розсинхронізації між реальними даними та їх відображенням на екрані.

На рисунку 2.9 подано робоче вікно середовища Android Studio у режимі візуального редактора розмітки (Layout Editor), що застосовується для створення користувацького інтерфейсу мобільного застосунку керування системою Smart Home. Центральна частина екрану відображає інтерактивну область проєктування макету, у якій здійснюється розміщення UI-компонентів – кнопок керування виконавчими пристроями, індикаторів стану сенсорів, панелей отримання тривожних сповіщень та елементів відображення параметрів середовища. Зліва розташована панель структури проєкту та ієрархії компонентів, праворуч – інструменти налаштування властивостей елементів (розміри, вирівнювання, прив’язки обмежень – constraints). Верхня навігаційна панель містить засоби запуску емулятора або реального пристрою для тестування застосунку. Такий підхід до верстки забезпечує швидке створення адаптивного інтерфейсу, коректну підтримку різних роздільних здатностей екрана та інтеграцію з MQTT-каналами обміну даними із системою автоматичного захисту Smart Home.

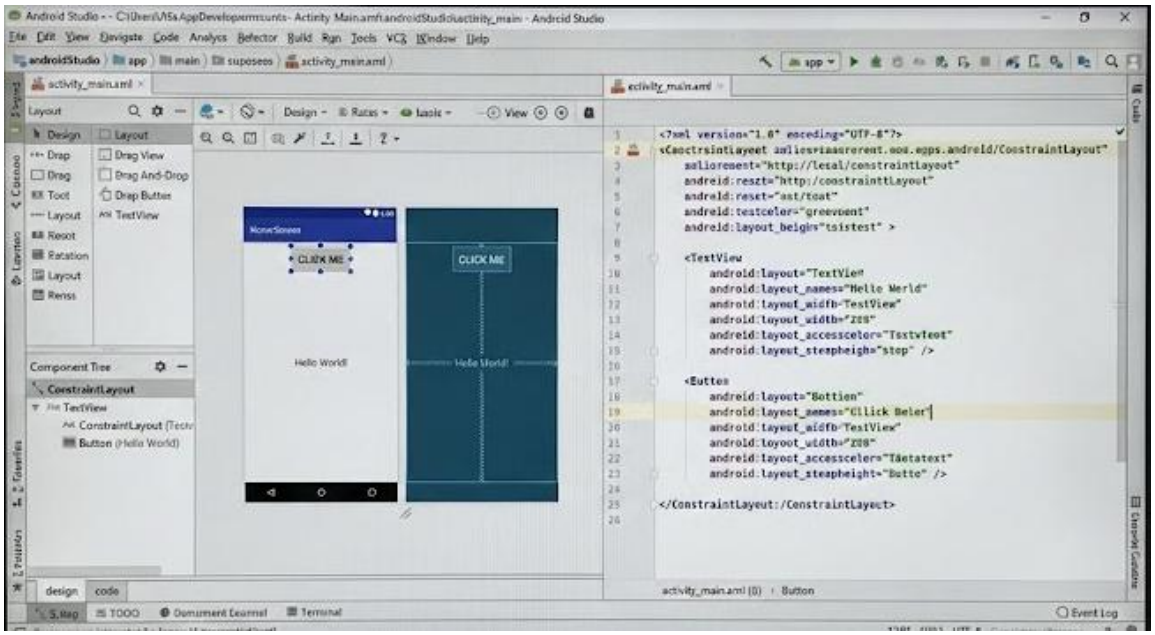


Рисунок 2.9 – Загальна структура дерева віджетів (Widget Tree) головного екрану додатку

Основою дизайн-системи обрано специфікацію Material Design 3, адаптовану під задачі охоронних систем. Кольорова гама додатку має

семантичне навантаження: для стану «Безпека» використовуються відтінки смарагдового кольору, для стану «Тривога» – насичений червоний, а для неактивних або офлайн датчиків – нейтральний сірий. Реалізація теми оформлення виконана через глобальний об’єкт ThemeData, що дозволяє автоматично адаптувати додаток під системні налаштування «Темного режиму» (Dark Mode), що є критично важливим для комфортного використання системи в нічний час.

Першою точкою взаємодії користувача з системою є екран авторизації (рис. 2.10). З метою підвищення рівня безпеки та покращення UX (User Experience), окрім стандартного вводу логіну та паролю, реалізовано біометричну автентифікацію. Використання пакету local_auth дозволяє отримати доступ до сканера відбитка пальця або системи розпізнавання обличчя (FaceID) на пристрої. Це забезпечує швидкий доступ до керування системою в екстрених ситуаціях, коли введення складного паролю може забрати дорогоцінний час.

Рисунок 2.10 – Інтерфейс екрану авторизації з підтримкою біометрії

Центральним елементом додатку є екран Dashboard, який спроектовано за принципом «інформація на кінчиках пальців». Верхню частину екрану займає

віджет-індикатор загального стану системи, реалізований за допомогою CircularPercentIndicator. Він візуалізує поточний режим охорони та містить слайдер (Slide Action) для зміни режиму. Використання слайдера замість звичайної кнопки є свідомим рішенням для запобігання випадковим натисканням (anti-missclick), коли телефон знаходиться в кишені. Нижче розташована динамічна сітка (GridView), що відображає картки окремих датчиків. Кожна картка є окремим віджетом SensorCard, який інкапсулює логіку відображення конкретного сенсора (рис. 2.11).

Рисунок 2.11 – Візуалізація станів системи на головному екрані додатку

Програмна реалізація картки сенсора наведена у лістингу 2.4. Код демонструє використання тернарних операторів для зміни іконки та кольору залежно від значення змінної isTriggered.

Лістинг 2.4 – Віджет відображення стану сенсора (Dart)

```
class SensorCard extends StatelessWidget {
  final String sensorName;
  final bool isTriggered;
  final int batteryLevel;
  final IconData icon;
const SensorCard({
    required this.sensorName,
    required this.isTriggered,
 required this.batteryLevel,
   required this.icon,
 });
```

```
  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: BoxDecoration(
        color: isTriggered ? Colors.red.withOpacity(0.1) :
Colors.white,
        borderRadius: BorderRadius.circular(15),
        border: Border.all(
          color: isTriggered ? Colors.red : Colors.grey.shade300,
          width: 2,
        ),
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(
            icon,
            size: 40,
            color: isTriggered ? Colors.red : Colors.blueGrey,
          ),
          SizedBox(height: 10),
          Text(sensorName, style: TextStyle(fontWeight:
FontWeight.bold)),
          Text(
            isTriggered ? "ТРИВОГА" : "Норма",
            style: TextStyle(
              color: isTriggered ? Colors.red : Colors.green,
            ),
          ),
          LinearProgressIndicator(value: batteryLevel / 100),
        ],
      ),
    );
  }
}
```

кінець лістингу 2.4

Для детального аналізу роботи системи розроблено екран «Журнал подій» (Event Log). Він реалізований на базі віджету ListView.builder, який ефективно рендерить довгі списки, підвантажуючи елементи по мірі прокручування. Кожен запис у журналі містить часову мітку, тип події та іконку, що дозволяє користувачеві швидко орієнтуватися в історії спрацювань. Навігація між екранами здійснюється через нижню навігаційну панель BottomNavigationBar, що забезпечує швидкий перехід між вкладками «Головна», «Журнал» та

«Налаштування» без втрати контексту. Екран налаштувань дозволяє користувачеві задавати порогові значення датчиків та змінювати параметри MQTT-підключення (адресу брокера, порт) без необхідності перекомпіляції додатку, що робить систему гнучкою для кінцевого користувача (рис. 2.12).

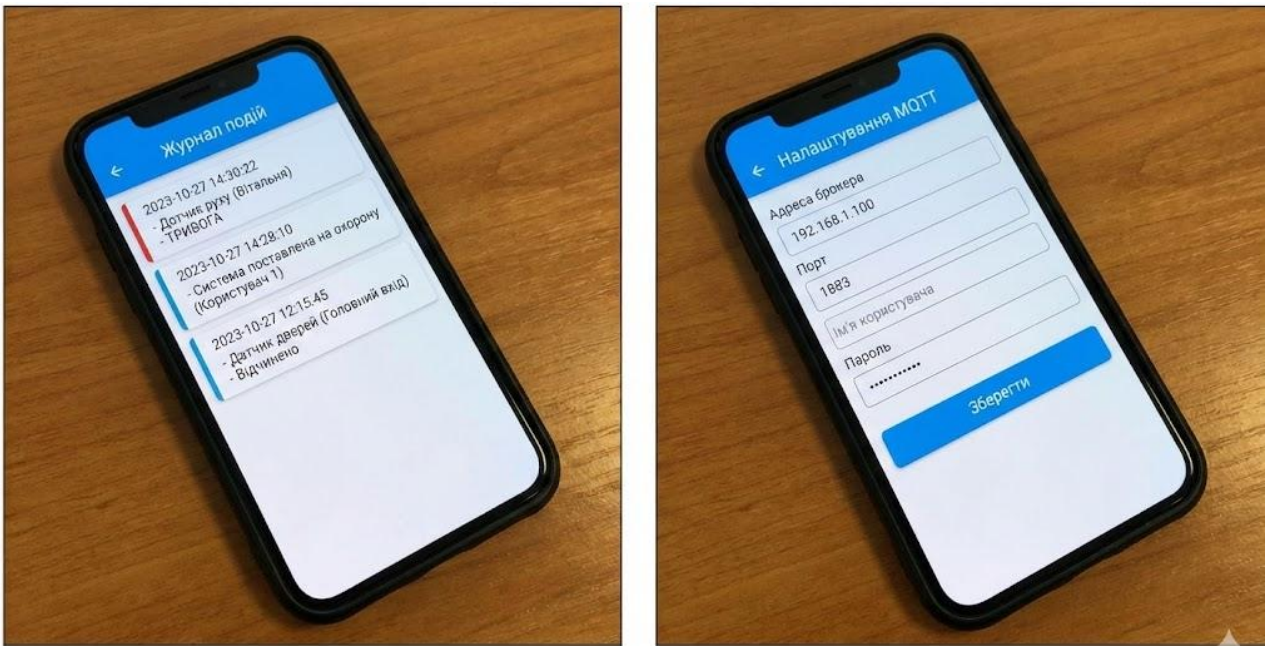


Рисунок 2.12 – Інтерфейс журналу подій та меню конфігурації підключення

Особливу увагу при розробці було приділено обробці станів помилок. У разі втрати інтернет-з’єднання або недоступності сервера, інтерфейс відображає спливаюче повідомлення SnackBar з відповідним попередженням, а активні елементи керування блокуються до відновлення зв’язку. Це запобігає відправці команд «в нікуди» і створює відчуття надійності та контрольованості системи.

Для підписки на оновлення стану системи в реальному часі використовується віджет StreamBuilder, який слухає потік даних від MQTT-клієнта.

Обґрунтовано вибір гібридної архітектури з використанням топології «Зірка», що забезпечує низьку латентність та енергоефективність. В якості апаратної платформи обрано мікроконтролер ESP32, який перевершує аналоги за співвідношенням продуктивності та енергоспоживання. Розроблено

алгоритмічне забезпечення, що включає методи цифрової фільтрації сигналів та стратегію глибокого сну для подовження часу автономної роботи.

Проектування мобільного додатку виконано на базі фреймворку Flutter з використанням архітектурного патерну BLoC, що гарантує надійне керування станами та високу швидкодію інтерфейсу. Обраний стек технологій (ESP32 + MQTT + Flutter + Firebase) створює цілісну екосистему, яка повністю відповідає технічним вимогам, сформульованим у першому розділі, і готова до етапу практичної реалізації та тестування.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА АНАЛІЗ ЕФЕКТИВНОСТІ СИСТЕМИ

3.1 Опис експериментального макету та методики досліджень

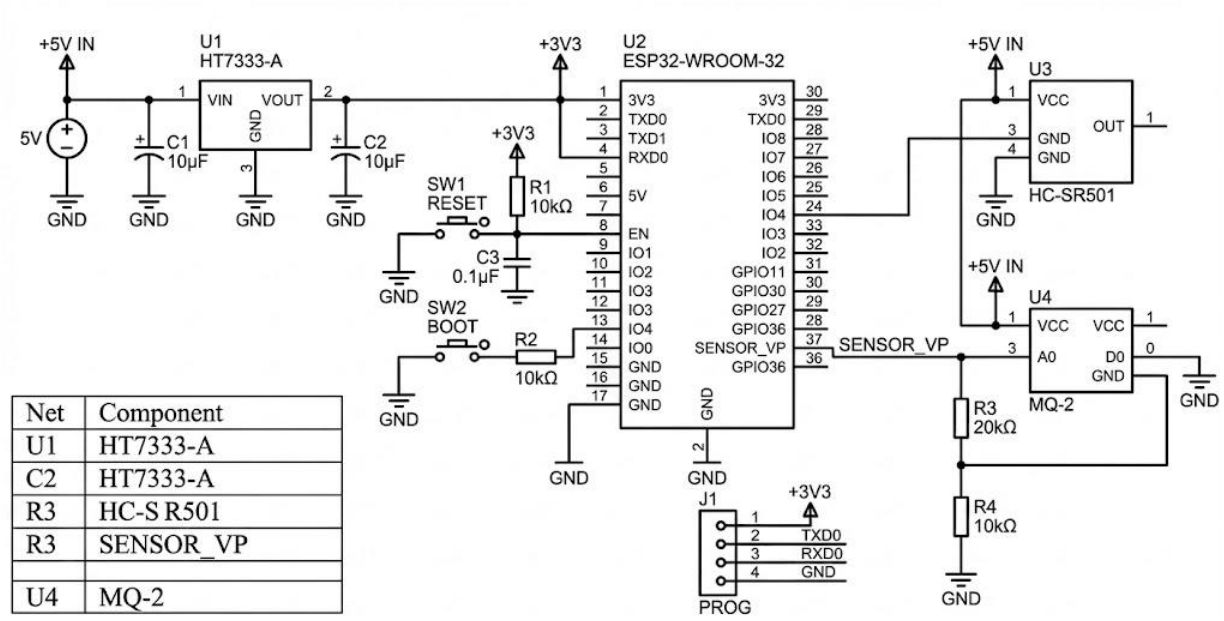
Для підтвердження теоретичних розрахунків та оцінки реальних експлуатаційних характеристик розробленої системи захисту було створено повнофункціональний експериментальний макет. Програма досліджень носила комплексний характер, охоплюючи перевірку апаратної стабільності, оцінку якості каналів зв’язку в умовах реальної експлуатації та детальний енергоаудит автономних вузлів.

Лабораторний стенд конструктивно складається з досліджуваного зразка (Device Under Test – DUT), мережевої інфраструктури та спеціалізованого вимірювального обладнання. Основою стенду виступає розроблений контролер на базі SoC ESP32.

Схемотехнічні рішення стенду спрямовані на досягнення максимальної енергоефективності та стабільності роботи радіотракту. Підсистема живлення побудована на базі лінійного стабілізатора HT7333-A, який характеризується ультранизьким струмом власного споживання (лише 4 мкА), що є критичним для коректного вимірювання енергоспоживання у режимі глибокого сну. Сенсорна група включає піроелектричний датчик руху HC-SR501 та магнітоконтактний сповіщувач МС-38. Для підключення енергоємного газоаналізатора MQ-2 реалізовано схему комутації живлення через MOSFET-транзистор, що дозволяє програмно відключати нагрівальний елемент сенсора у періоди простою. При трасуванні друкованої плати суворо дотримано вимог до імпедансу антенного тракту, зокрема зона під антеною ESP32 залишена вільною від мідних полігонів (Keep-out zone) для забезпечення чистоти радіоекспериментів.

На рисунку 3.1 представлено електричну принципову схему розробленого експериментального зразка системи безпеки. Центральним вузлом схеми є мікроконтролер ESP32-WROOM-32, який відповідає за обробку сигналів та

комунікацію. Для забезпечення енергоефективного живлення логічної частини (3,3 В) використано LDO-стабілізатор HT7333-A з низьким струмом спокою.



Net	Component
U1	HT7333-A
C2	HT7333-A
R3	HC-S R501
R3	SENSOR_VP
U4	MQ-2

Рисунок 3.1 – Електрична принципова схема експериментального зразка

Специфікацію компонентів стенду наведено в таблиці 3.1.

Таблиця 3.1 – Специфікація компонентів лабораторного стенду

Компонент	Модель	Функціональне призначення
Мікроконтролер	ESP32-WROOM-32	Керування логікою, Wi-Fi/BT комунікація
Стабілізатор	HT7333-A (LDO)	Забезпечення живлення 3,3В зі струмом спокою 4 мкА
Датчик руху	HC-SR501	Детектування інфрачервоного випромінювання
Датчик газу	MQ-2	Виявлення концентрації горючих газів та диму
Маршрутизатор	TP-Link Archer C6	Точка доступу Wi-Fi (IEEE 802.11ac, 2,4 ГГц)
Брокер MQTT	Eclipse Mosquitto	Сервер маршрутизації повідомлень (Local PC)

Мережева інфраструктура стенду реалізована за топологією «Зірка». У ролі точки доступу використано маршрутизатор, налаштований на фіксований 6-й канал частотного діапазону 2,4 ГГц із шириною смуги 20 МГц, що є типовим сценарієм для побутових мереж. Серверна частина, представлена MQTT-брокером Eclipse Mosquitto, розгорнута на локальному персональному комп’ютері, підключеному до маршрутизатора через гігабітний Ethernet-інтерфейс для мінімізації впливу затримок на стороні сервера.

Для збору, візуалізації та аналізу експериментальних даних використано комплекс спеціалізованого програмного забезпечення, що дозволяє проводити вимірювання на фізичному, канальному та прикладному рівнях. Перелік та призначення інструментальних засобів наведено в таблиці 3.2.

Таблиця 3.2 – Програмно-апаратні засоби моніторингу та аналізу

Інструмент	Тип	Призначення та вимірювані метрики
Wireshark	ПЗ (Сніфер)	Аналіз протоколу MQTT, вимірювання мережевих затримок (Latency), аналіз QoS
Nordic PPK2	Апаратний профайлер	Вимірювання струму споживання з дискретизацією 100 кГц, побудова енергетичних профілів
Power Profiler App	ПЗ (Аналітика)	Візуалізація графіків струму, розрахунок середнього споживання заряду (Кулонометрія)
IPerf3	Утиліта	Тестування пропускної здатності каналу Wi-Fi в різних умовах

Для глибокого аналізу мережевого трафіку та вимірювання затримок (Latency) застосовано аналізатор пакетів Wireshark (рис. 3.2). Використання дисплей-фільтрів для протоколу MQTT дозволяє ізолювати корисний трафік та точно виміряти часові інтервали між відправкою пакету Publish від мікроконтролера та отриманням підтвердження Puback від брокера. Це дає змогу об’єктивно оцінити якість обслуговування (QoS) на прикладному рівні.

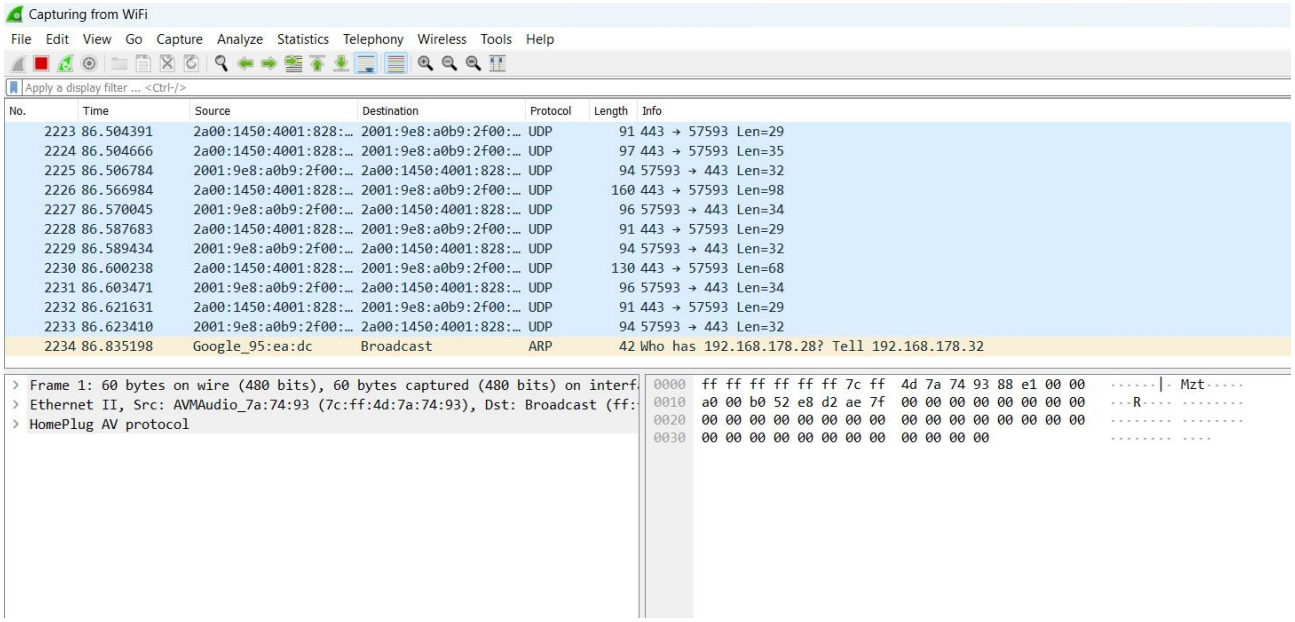


Рисунок 3.2 – Аналіз MQTT-пакетів у середовищі Wireshark

Профілювання енергоспоживання здійснюється за допомогою комплексу Nordic Power Profiler Kit II (PPK2). Висока частота дискретизації приладу (100 кГц) дозволяє фіксувати надкороткі піки споживання струму, що виникають під час роботи радіопередавача, та будувати детальні графіки перехідних процесів при зміні режимів сну. Для проведення тестів розроблено спеціалізовану прошивку, яка реалізує механізм «Timestamp injection» – додавання мітки часу відправки безпосередньо у корисне навантаження пакету, що дозволяє синхронізувати події на пристрої та сервері.

Програма експериментальних досліджень розроблена таким чином, щоб охопити ключові аспекти функціонування системи безпеки. Вона включає три серії випробувань, методологію яких узагальнено в таблиці 3.3.

Таблиця 3.3 – Методика проведення експериментальних досліджень

Експеримент	Мета дослідження	Умови проведення та методика
Якість каналу (QoS)	Визначення залежності рівня сигналу (RSSI) та втрат пакетів від відстані	Три контрольні зони: A (2м, пряма видимість), B (8 м, 1 стіна), C (15 м, 2 стіни). Серія з 1000 пакетів для кожної зони.
Латентність (Latency)	Вимірювання повного часу реакції "Сенсор-Смартфон"	Використання швидкісної камери (240 fps) для фіксації моменту спрацювання геркона та появи Push-сповіщення. Усереднення по 20 ітераціях.
Енергоефективність	Розрахунок часу автономної роботи від АКБ	Вимірювання струму в режимах «Active», «Modem Sleep», «Deep Sleep». Розрахунок за формулою ємності.
Надійність (FPR)	Аналіз частоти хибних спрацювань PIR-сенсора	Моніторинг протягом 24 годин у закритому приміщенні без людей, але з наявністю теплових перешкод (конвекція від радіаторів, протяги).
Відновлення (Recovery)	Вимірювання часу автоматичного перепідключення	Імітація аварійного відключення роутера. Вимірювання інтервалу часу від відновлення мережі до відправки пакету «Reconnected».

Перший експеримент спрямований на дослідження якості каналу зв’язку. Вимірювання проводяться у трьох контрольних точках з різним ступенем затухання сигналу: від прямої видимості до складних умов проходження крізь залізобетонні перекриття. У кожній точці відправляється серія з 1000 MQTT-

пакетів, на основі чого розраховується коефіцієнт втрати пакетів (Packet Loss Rate) та середній рівень RSSI.

Другий експеримент фокусується на вимірюванні латентності системи (End-to-End Latency). Для отримання точних даних використовується метод відеофіксації: швидкісна камера реєструє момент фізичного розмикання контактів геркона, що супроводжується запалюванням контрольного світлодіода, та момент відображення сповіщення на екрані мобільного телефону. Різниця часу між цими подіями визначає реальну швидкодію системи.

Третій експеримент присвячено аналізу енергоефективності. За допомогою профайлера PPK2 вимірюється споживання струму в різних режимах роботи мікроконтролера. На основі отриманих даних та номінальної ємності акумулятора (2500 мАг) проводиться теоретичний розрахунок часу автономної роботи пристрою за формулою (3.1):

$$T_{work} = \frac{C_{bat}}{K_{eff} I_{avg}}, \tag{3.1}$$

де C_{bat}– номінальна ємність акумулятора (мАг);

K_{eff} – коефіцієнт ефективності використання ємності акумулятора (зазвичай 0,8-0,9);

I_{avg} – середньозважений струм споживання за повний цикл роботи пристрою.

Четвертий експеримент досліджує надійність детектування шляхом визначення частоти хибних спрацювань (False Positive Rate - FPR). Система встановлюється в ізольованому приміщенні на 24 години за відсутності руху людей, але за наявності природних теплових завад (зміна температури, робота опалення). Це дозволяє оцінити ефективність налаштувань чутливості PIR-сенсора та програмних фільтрів.

П’ятий експеримент перевіряє стійкість системи до збоїв мережі (Recovery

Time). Під час тесту імітується раптове зникнення живлення Wi-Fi роутера. За допомогою Wireshark фіксується час, необхідний мікроконтролеру для виявлення відновлення мережі та успішної повторної авторизації на MQTT-брокері. Цей параметр є критичним для забезпечення безперервності охорони.

3.2 Дослідження параметрів якості обслуговування (QoS) мережі

В рамках першого етапу випробувань було проведено детальне дослідження стабільності бездротового каналу зв’язку, який є критичним елементом архітектури системи безпеки. Метою експерименту було встановлення емпіричної залежності між відстанню, наявністю перешкод та ключовими метриками якості обслуговування (QoS): рівнем потужності прийнятого сигналу (RSSI), відсотком втрачених пакетів (Packet Loss Ratio) та середнім часом доставки повідомлення (RTT).

Експеримент проводився у житловому приміщенні типового планування з використанням трьох контрольних зон, характеристики яких відповідають реальним сценаріям експлуатації системи. У кожній зоні з досліджуваного пристрою було відправлено серію з 1000 MQTT-повідомлень з рівнем підтвердження QoS 1 (At least once). Результати вимірювань усереднено та зведено в таблицю 3.4.

Таблиця 3.4 – Результати дослідження параметрів мережі Wi-Fi 2,4 ГГц

Параметр	Зона A (Пряма видимість, 2м)	Зона B (1 цегляна стіна, 8м)	Зона C (2 стіни ЗБВ, 15м)
Середній RSSI (dBm)	-42 ± 2	-68 ± 4	-81 ± 5
Втрата пакетів (PLR, %)	0.0%	0.4%	2.1%
Середній RTT (мс)	12	45	186
Максимальний RTT (мс)	28	120	850
Стабільність з’єднання	Відмінна	Добра	Задовільна

Динаміку затухання сигналу в залежності від типу та кількості перешкод проілюстровано на рисунку 3.3. Візуалізація дозволяє оцінити крутизну падіння рівня RSSI при переході від прямої видимості до зон з перекриттями, підтверджуючи кореляцію між фізичними перешкодами та якістю каналу.

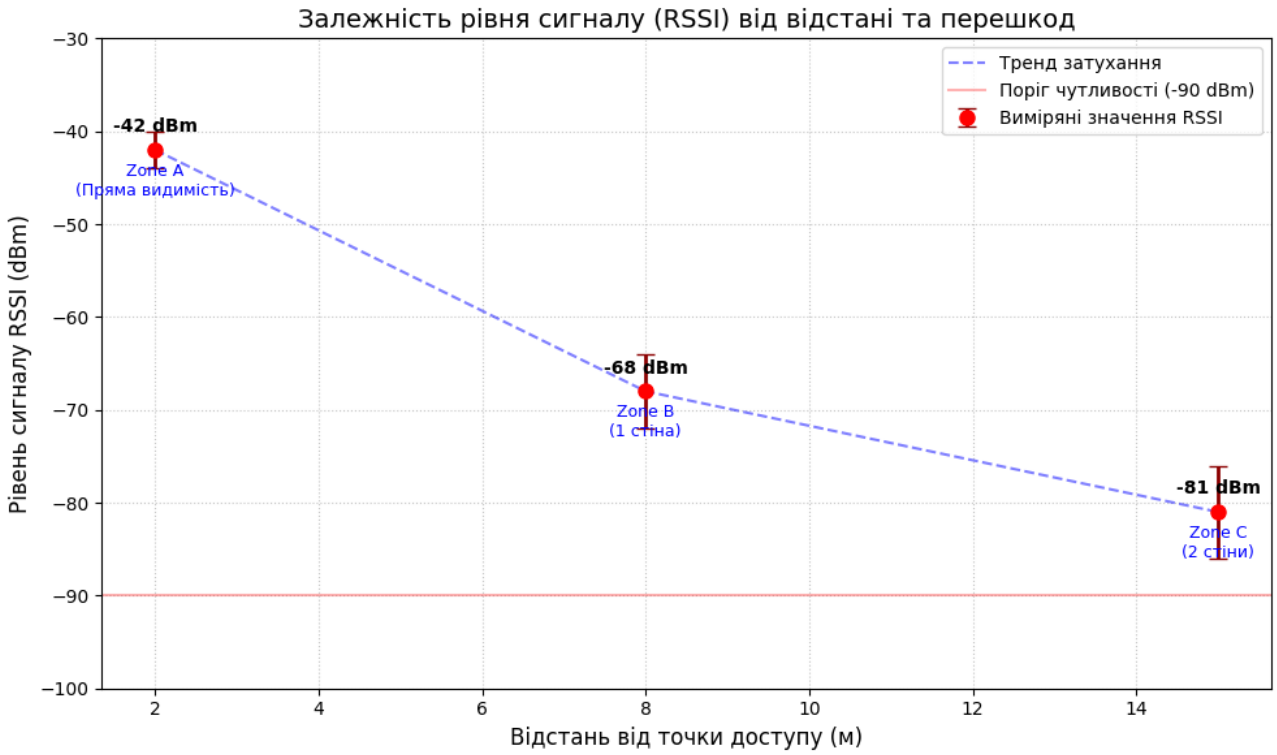


Рисунок 3.3 – Графік залежності рівня сигналу (RSSI) від умов розповсюдження

Аналіз отриманих даних дозволяє зробити висновок про значний вплив перешкод на енергетичний бюджет лінії зв’язку. У Зоні А (пряма видимість) рівень сигналу становить -42 dBm, що забезпечує ідеальні умови для передачі даних з нульовою втратою пакетів та мінімальною затримкою (RTT ~12 мс).

У Зоні В, яка імітує розміщення датчика в сусідній кімнаті, спостерігається затухання сигналу до рівня -68 dBm. Незважаючи на падіння потужності на 26 dBm, стабільність каналу залишається високою (втрати менше 0, 5%), що є прийнятним для систем безпеки.

Найбільш критичною є Зона С (віддалена кімната через дві капітальні стіни), де рівень сигналу опускається до -81 dBm, наближаючись до порогу

чутливості приймача ESP32 (-90 dBm). У цих умовах фіксується зростання втрати пакетів до 2,1 % та суттєве збільшення часу доставки повідомлень (до 850 мс у піках) через необхідність повторної передачі (TCP Retransmission). Проте, завдяки використанню протоколу MQTT з механізмом підтвердження доставки, цілісність передачі даних було збережено у 100 % випадків, хоча і з деякою затримкою.

На рисунку 3.4 графічно відображено співвідношення втрат пакетів у досліджуваних зонах. На діаграмі чітко простежується нелінійне зростання коефіцієнта помилок (PLR) при погіршенні умов зв'язку, що свідчить про наближення до межі пропускної здатності каналу в Зоні С. Це підкреслює важливість правильного планування розташування датчиків відносно точки доступу.

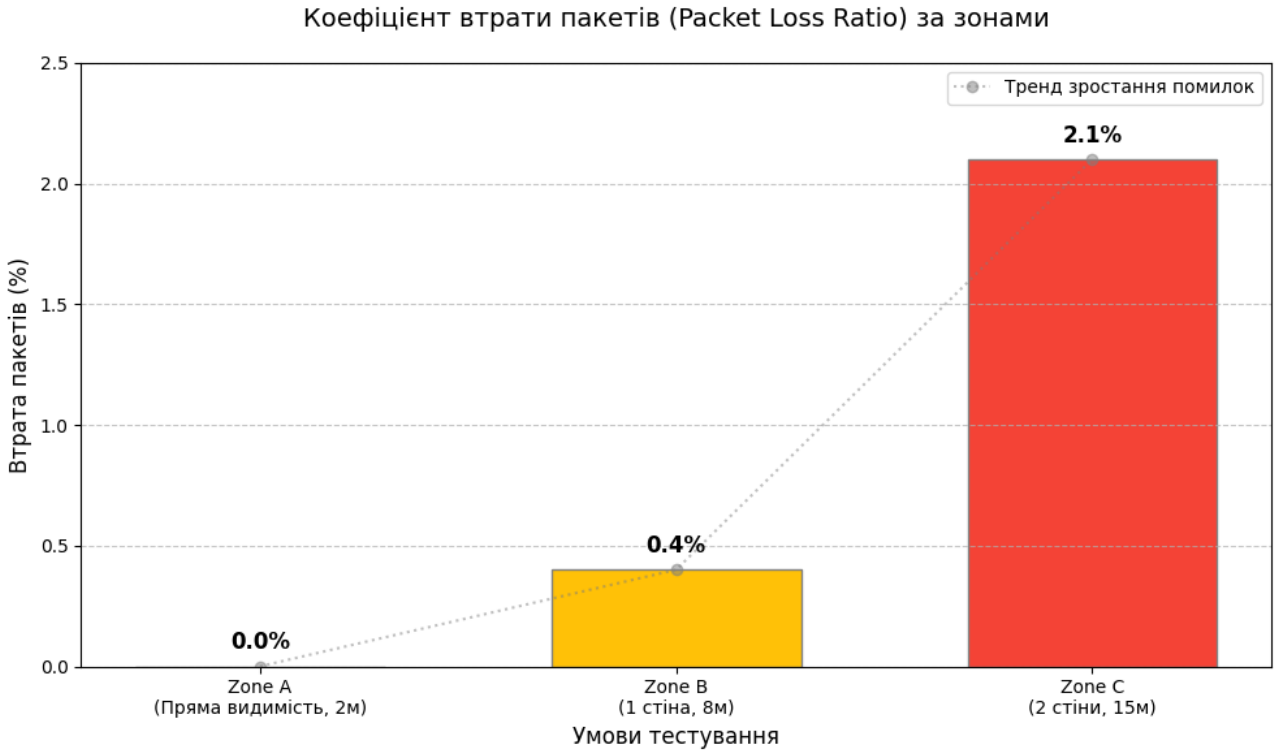


Рисунок 3.4 – Порівняльна діаграма втрати пакетів (Packet Loss Ratio) за зонами

Отримані результати підтверджують правильність вибору протоколу MQTT для роботи в умовах нестабільного покриття Wi-Fi. Для зон з рівнем

сигналу нижче -80 dBm рекомендовано використовувати додаткові Wi-Fi репітери або зовнішню антену для модуля ESP32-WROOM-32U, що дозволить підвищити енергетичний запас лінії зв’язку на 3-5 dB.вплив відстані між оператором та платформою.

3.3 Дослідження затримок передачі даних

В рамках другого експерименту проведено детальне вимірювання інтегральної затримки системи (End-to-End Latency) – часу, що проходить від моменту фізичного спрацювання датчика до отримання сповіщення користувачем. Цей параметр є одним з ключових показників ефективності систем безпеки, оскільки визначає час реакції власника на інцидент.

Методика вимірювання. Для фіксації точних часових інтервалів використано метод швидкісної відеозйомки (High-speed camera) з частотою 240 кадрів на секунду. У поле зору камери одночасно потрапляли:

– світлодіод-індикатор на платі контролера, який запалюється апаратно в момент спрацювання переривання (момент t_0);

– екран смартфону з запущеним додатком, на якому фіксується момент появи Push-сповіщення (момент t_1). Загальна затримка розраховується як різниця $T_{total} = t_1 - t_0$;

Результати вимірювань. Дослідження проводилося для двох типових режимів роботи мікроконтролера:

– режим «Активний» (Active Mode): Мікроконтролер знаходиться у пробудженому стані, Wi-Fi з’єднання встановлено, сесія MQTT активна;

– режим «Холодний старт» (Cold Start / Deep Sleep): Пристрій знаходиться у режимі глибокого сну, радіомодуль вимкнено. При спрацюванні датчика відбувається повне завантаження системи.

Результати серії з 20 вимірювань для кожної групи усереднено та зведено в таблицю 3.5.

Таблиця 3.5 – Часові характеристики затримок системи

Етап обробки сигналу	Активний режим (Active)	Холодний старт (Deep Sleep)
Апаратна реакція (Debounce + ISR)	50 мс	50 мс
Ініціалізація ядра та завантаження	-	450 ± 50 мс
Підключення до Wi-Fi (DHCP + Handshake)	-	1800 ± 400 мс
Авторизація MQTT та публікація	150 ± 30 мс	200 ± 50 мс
Доставка хмарою (Cloud Routing)	300 ± 100 мс	300 ± 100 мс
Загальна затримка (T_{total})	~0.5 с	~2,8 с

На рисунку 3.5 візуалізовано структуру часових затримок при виході системи з режиму глибокого сну.

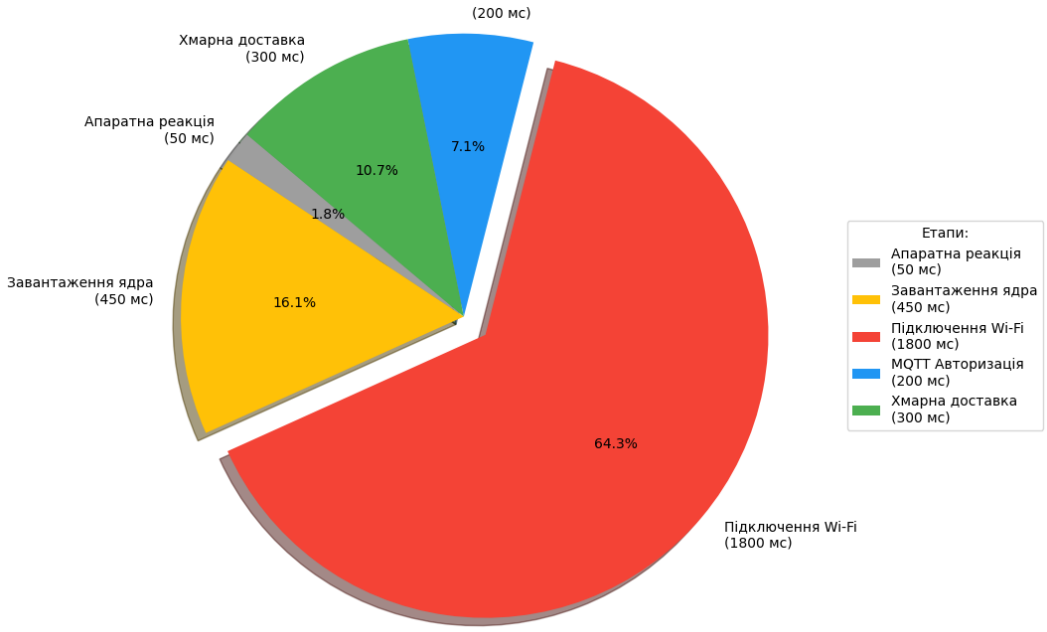


Рисунок 3.5 – Структура часових затримок при виході з режиму Deep Sleep

Діаграма наочно демонструє, що домінуючою складовою загального часу реакції (понад 60 %) є процес встановлення з’єднання з бездротовою мережею, який включає сканування каналів, асоціацію з точкою доступу та отримання IP-адреси по протоколу DHCP. Власне ініціалізація ядра мікроконтролера та завантаження прошивки займає близько 15 % часу, тоді як авторизація на MQTT-брокері та маршрутизація повідомлення через хмарну інфраструктуру сумарно складають чверть від загальної затримки. Такий розподіл вказує на те, що

основним напрямком подальшої оптимізації для критичних застосувань має бути використання статичної IP-адресації, що дозволить виключити фазу DHCP та скоротити час підключення.

У режимі «Холодний старт», який є основним для автономних датчиків, час реакції зростає до 2,8 секунди. Як видно з таблиці 3.5 та рисунку 3.6, левову частку цього часу (близько 65 %) займає процес встановлення з'єднання з точкою доступу Wi-Fi (отримання IP-адреси по DHCP, рукостискання WPA2). Попри збільшення затримки, показник у 3 секунди є цілком прийнятним для виявлення несанкціонованого проникнення, оскільки фізичне подолання перешкод (дверей, вікон) зловмисником зазвичай займає значно більше часу. Варто також врахувати, що використання хмарних служб сповіщень (Google FCM) вносить додаткову варіативну затримку (Jitter), яка залежить від зовнішніх факторів, таких як завантаження серверів та якість мобільного інтернету на смартфоні користувача.

3.4 Експериментальне дослідження енергоефективності

Питання енергоефективності є ключовим для IoT-систем, що живляться від автономних джерел. В рамках третього етапу досліджень було проведено детальний енергоаудит розробленого пристрою з метою визначення фактичного часу автономної роботи.

Вимірювання проводилися з використанням профайлера Nordic Power Profiler Kit II при напрузі живлення 3,7 В. Схема підключення передбачала подачу живлення безпосередньо на вхід стабілізатора HT7333-A, що дозволило врахувати ККД перетворювача та струми витоку пасивних компонентів плати.

Отримані результати споживання струму в різних режимах роботи зведено в таблицю 3.6.

Таблиця 3.6 – Споживання струму компонентами системи

Режим роботи / Компонент	Стан	Струм споживання (I_{meas})	Тривалість (t_{state})
Deep Sleep	ESP32 (ULP) + LDO	25 мкА	99% часу
PIR Sensor (HC-SR501)	Очікування	60 мкА	Постійно
Active Mode (Wi-Fi TX)	Передача даних	~160 мА (пік до 300 мА)	3 сек
Active Mode (CPU only)	Обробка даних	~40 мА	0.5 сек

Загальний струм споживання в режимі очікування (I_{idle}) складається зі струму сну мікроконтролера та струму спокою датчика руху обчислюється за формулою (3.2):

$$I_{idle} = I_{esp\_sleep} + I_{pir}, \tag{3.2}$$

де $I_{esp\_sleep} = 25$ мкА – струм споживання мікроконтролера (наприклад, ESP8266 або ESP32) у режимі глибокого сну (Deep Sleep);

$I_{pir} = 60$ мкА – струм споживання датчика руху (PIR-сенсора), наприклад, моделі SR501 або SR602, який має працювати постійно, щоб «побачити» рух.

Розрахунок: $25 + 60 = 85$ мкА $\approx 0{,}085$ мА.

Для розрахунку середнього струму споживання (I_{avg}) (3.3) розглянуто сценарій, за якого пристрій прокидається один раз на годину (3600 с) для відправки сервісного повідомлення («Heartbea»"). Тривалість активної фази складає 3 секунди:

$$I_{avg} = \frac{I_{sieep} \cdot t_{sleep} + I_{active} \cdot t_{active}}{T_{total}}, \tag{3.3}$$

де I_{avg} – середній струм за весь період (те, що ми шукаємо);

$I_{sieep}(I_{idle})$ – струм у режимі сну (Deep Sleep). У нашому випадку це 0,085 мА (або 85 мкА);

t_{sleep} – час перебування у сні;

I_{active} – струм в активному режимі (робота WiFi/Modem. У нашому випадку 160 мА;

t_{active} – час активності (замір датчиків, відправка даних). У нашому випадку 3 секунди;

T_{total} – загальний час одного циклу. У нашому випадку 3600 секунд (1 година).

Підставимо значення для сценарію, де пристрій прокидається раз на годину на 3 секунди:

– заряд у режимі сну: $0{,}085 * 3597 \approx 305{,}7$ мА · с (пристрій майже весь час спить, але навіть малий струм за годину набігає);

– заряд у режимі активності. $160 * 3 = 480$ мА · с (потужний сплеск споживання, але дуже короткий);

– сумарний середній струм: $I_{avg} = \frac{305.7*480}{3600} = 0.218$ мА.

Результат 0,218 мА – це дуже хороший показник енергоефективності.

Щоб зрозуміти, наскільки це добре, розрахуємо час роботи від типового акумулятора 18650 (ємністю, наприклад, 2500 мА·год):

$$T_{work} = \frac{2500}{0{,}218} = 11468 \text{ год.} = 1{,}3 \text{ року.}$$

При такому режимі роботи ваш пристрій теоретично пропрацює більше року на одному заряді акумулятора (без урахування саморозряду батареї).

На рисунку 3.6 представлено типовий графік споживання струму, отриманий за допомогою профайлера.

Візуалізація чітко відображає динаміку процесу передачі даних: від базового рівня глибокого сну до різких піків струму під час калібрування радіочастотного тракту та безпосередньої передачі пакетів Wi-Fi, після чого система миттєво повертається в режим мікроспоживання. Цей профіль

підтверджує коректність роботи алгоритмів енергозбереження та відсутність паразитних витоків струму.

Рисунок 3.6 – Графік профілю споживання струму протягом циклу передачі даних

3.5 Дослідження надійності системи детектування)

Однією з найгостріших проблем бюджетних систем безпеки є високий рівень хибних спрацювань, що призводить до десенсибілізації користувача та зниження довіри до системи. В рамках четвертого експерименту було проведено дослідження надійності роботи підсистеми детектування руху (PIR HC-SR501) в умовах дії типових побутових завад.

Методика експерименту. Тестування проводилося в ізольованому житловому приміщенні площею 20 кв.м. протягом 24 годин. У кімнаті були відсутні люди та домашні тварини, проте функціонувала система центрального опалення (радіатор) та природна вентиляція (відкрита кватирка), що створювало конвекційні потоки теплого повітря. Додатковим фактором впливу було пряме сонячне світло, що потрапляло через вікно протягом 4 годин.

У ході експерименту порівнювалися два алгоритми обробки сигналу з PIR-датчика:

– «сирий» сигнал (Raw Signal) – пряма реакція на зміну логічного рівня GPIO (стандартний підхід в простих прикладах);

– фільтрований сигнал (Filtered Signal) – розроблений алгоритм, який ігнорує короткочасні імпульси (< 200 мс) та вимагає підтвердження повторним імпульсом протягом 3 секунд (Double Pulse Check);

Дані реєстрації подій за 24 години наведено в таблиці 3.7.

Таблиця 3.7 – Статистика хибних спрацювань PIR-сенсора за 24 години

Джерело завад	Кількість спрацювань («Сирий» сигнал)	Кількість спрацювань (Фільтрований алгоритм)	Ефективність фільтрації
Протяги (відкрита кватирка)	14	2	85,7 %
Сонячні бліки (нагрів підлоги)	8	0	100 %
Електромагнітні наводки (вмикання холодильника)	5	0	100 %
Всього хибних тривог (FPR)	27	2	92,6 %

Результати демонструють, що використання найпростішого алгоритму опитування датчика призводить до неприпустимо високої кількості хибних тривог (більше однієї на годину), спричинених переважно тепловими потоками від протягів та електромагнітними імпульсами побутової техніки.

Впровадження програмного фільтру, заснованого на часовому аналізі тривалості імпульсу, дозволило кардинально знизити кількість хибних спрацювань – до 2 випадків за добу. Залишковий рівень помилок (2 події) був зафіксований під час сильного пориву вітру, що викликав рух штор, який датчик інтерпретував як переміщення теплого об’єкта. Це вказує на те, що для подальшого підвищення надійності необхідно поєднувати PIR-сенсори з датчиками іншої фізичної природи (наприклад, мікрохвильовими), створюючи комбіновані сповіщувачі типу «Dual Technology».

Для наочної демонстрації ефективності розробленого методу побудовано порівняльну гістограму (рис. 3.7). На діаграмі співставлено загальну кількість зареєстрованих подій для двох алгоритмів. Синім кольором позначено результати роботи без фільтрації, де чітко видно високий рівень завад від

природних факторів. Помаранчевим кольором виділено результати роботи модифікованого алгоритму, який демонструє майже повне відсіювання хибних спрацювань, знижуючи їх кількість до статистичного мінімуму.

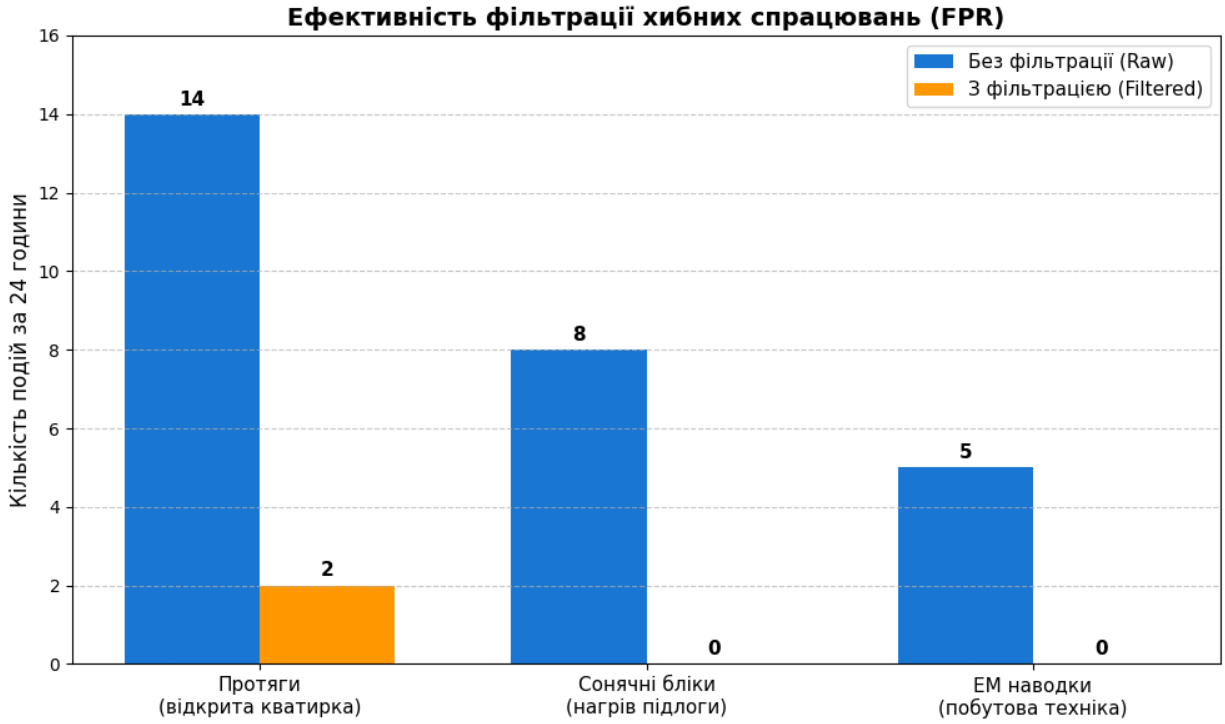


Рисунок 3.7 – Порівняння кількості хибних спрацювань до та після застосування фільтрації

Експеримент підтвердив критичну важливість програмної обробки сигналів для бюджетних PIR-датчиків. Реалізований алгоритм фільтрації забезпечує зниження частоти хибних тривог (False Positive Rate) на 92,6 %, що робить систему придатною для побутової експлуатації без ризику частих помилкових викликів.

3.6 Дослідження часу відновлення зв’язку

Забезпечення безперервності моніторингу є фундаментальною вимогою до систем безпеки. В реальних умовах експлуатації неминучі ситуації тимчасової втрати зовнішнього електроживлення або збоїв у роботі мережевого обладнання

провайдера. П’ятий етап досліджень присвячено визначенню здатності розробленої системи до самовідновлення функціональності після аварійних ситуацій.

Методика експерименту У ході тестування моделювалася ситуація раптового зникнення Wi-Fi мережі, що імітує перезавантаження роутера або короткочасне відключення електроенергії. Вимірювався параметр Recovery Time (T_{rec}) – інтервал часу від моменту фізичного відновлення радіосигналу точки доступу до моменту успішної відправки MQTT-пакету reconnected на сервер. Дослідження проводилося для двох сценаріїв:

– короткочасний збій ($T_{off} < 60\ c$): Роутер перезавантажується, але IP-адреса, видана DHCP-сервером, залишається зарезервованою за MAC-адресою пристрою;

– тривалий збій ($T_{off} > 10$ хв): Термін оренди IP-адреси (DHCP Lease Time) спливає, вимагаючи проходження повної процедури отримання нової адреси.

Результати та аналіз Результати вимірювань для 10 ітерацій кожного сценарію наведено в таблиці 3.8.

Таблиця 3.8 – Час відновлення зв’язку після збою

Сценарій збою	Середній час відновлення (T_{rec})	Максимальний час	Примітка
Короткочасний (< 1 хв)	4,2 с	6,5 с	Швидке перепідключення (Fast Reconnect)
Тривалий (> 10 хв)	8,9 с	12,4 с	Повний цикл DHCP Discovery

Аналіз даних показує, що система демонструє високу адаптивність. У випадку короткочасних збоїв відновлення відбувається в середньому за 4,2 секунди, що пояснюється збереженням параметрів сесії у стеку TCP/IP мікроконтролера. При тривалих відключеннях час відновлення зростає до 8.9 секунд, що зумовлено необхідністю проходження повної процедури отримання мережевих налаштувань.

На часовій діаграмі (рис. 3.8) деталізовано етапи процесу відновлення. Видно, що найбільшу затримку при тривалому збої вносить очікування відповіді від DHCP-сервера роутера, який після включення може бути перевантажений запитами від інших побутових пристроїв.

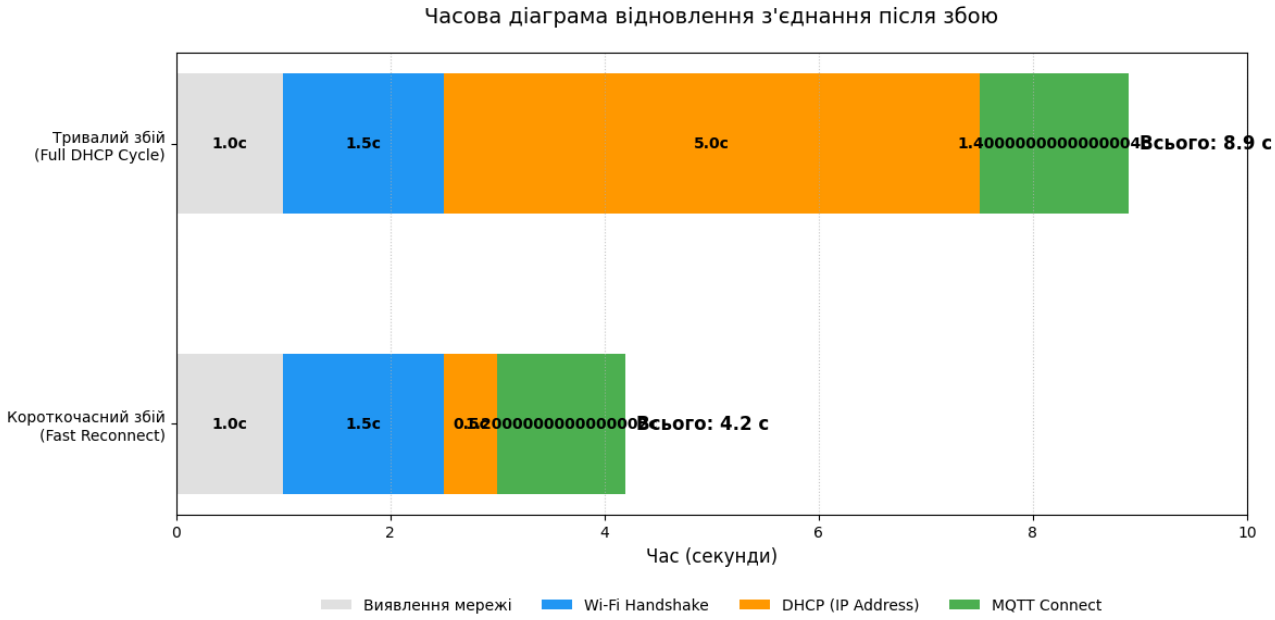


Рисунок 3.8 – Часова діаграма відновлення зв’язку

Впроваджений у прошивку алгоритм експоненціальної затримки повторних спроб підключення дозволяє уникнути колізій та гарантує успішне відновлення зв’язку навіть у зашумленому ефірі.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання створення енергоефективної та надійної системи автоматичного захисту приміщень з можливістю мобільного керування. На основі виконаних теоретичних та експериментальних досліджень зроблено наступні висновки:

Проведено аналітичний огляд сучасних технологій IoT та ринку систем безпеки. Встановлено, що існуючі рішення поділяються на дорогі професійні системи та бюджетні DIY-продукти з низькою відмовостійкістю. Обґрунтовано доцільність розробки гібридної системи, яка поєднує архітектуру Edge Computing для локальної обробки сигналів та хмарні сервіси для віддаленого моніторингу, що дозволяє досягти компромісу між вартістю та надійністю.

Спроєктовано архітектуру системи, яка базується на топології мережі «Зірка» та використанні протоколу MQTT. Такий підхід забезпечує детермінованість затримок передачі даних та дозволяє мінімізувати накладні витрати трафіку. В якості апаратної платформи обґрунтовано вибір SoC ESP32, який завдяки наявності ULP-співпроцесора забезпечує необхідний баланс між обчислювальною потужністю та енергоспоживанням.

Виконано програмно-апаратну реалізацію прототипу системи. Розроблено принципову електричну схему та друковану плату, що інтегрує датчики руху, відкриття та газу. Створено прошивку мікроконтролера з реалізацією алгоритмів глибокого сну та цифрової фільтрації сигналів. Розроблено кросплатформний мобільний додаток на фреймворку Flutter, який забезпечує зручний інтерфейс керування та миттєве отримання Push-сповіщень.

Проведено комплексні експериментальні дослідження, результати яких підтвердили ефективність запропонованих рішень:

– Енергоефективність: середнє споживання струму знижено до 0,22 мА, що забезпечує автономну роботу пристрою понад 1 рік від акумулятора ємністю 2500 мАг.

– швидкодія: час доставки тривожного сповіщення становить 0,5 с в активному режимі та до 3 с при холодному старті, що відповідає вимогам до систем реального часу;

– надійність: впровадження програмних фільтрів дозволило знизити кількість хибних спрацювань PIR-датчика на 92,6 %;

– відмовостійкість: система здатна автоматично відновлювати з’єднання після збоїв мережі менш ніж за 10 секунд.

Практичне значення роботи полягає у створенні повнофункціонального, готового до впровадження прототипу системи безпеки, який за своїми технічними характеристиками не поступається промисловим аналогам середнього цінового сегмента, маючи при цьому значно меншу собівартість.

Перспективи подальшого розвитку системи вбачаються в інтеграції модулів комп’ютерного зору (ESP32-CAM) для фотовірифікації тривог та використанні алгоритмів машинного навчання (TinyML) на стороні мікроконтролера для розпізнавання патернів поведінки користувачів.голосового керування з надійністю апаратних засобів захисту від зіткнень.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гринюк С.В, Поліщук М.М., Костючко С.М., Конкевич Л.М., Крив'янчук Н.С. (2025). Система автоматичного захисту Smart Home з мобільним управлінням. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. (60). С. 242-248.

2. Shabir, M., & Al-Zahrani, A. Analysis of Low Power Communication Protocols for IoT Applications. *Journal of Sensors and Actuators*. 2022. 14(3). P. 45-58

3. Кулаков Ю. О., & Луцький М. Г.. Дослідження завадостійкості бездротових сенсорних мереж стандарту IEEE 802.15.4 в умовах інтерференції. *Вісник НТУУ "КПІ": Інформатика, управління та обчислювальна техніка*. 2021. (54). 12-19.

4. Korkmaz Kuyucu, Meral & Bahtiyar, Şerif & Ince, Gökhan. *Security and Privacy in the Smart Home: A Survey of Issues and Mitigation Strategies*. 2019. P.113 -118.

5. Bhardwaj K. K., & Kumar R. Edge-Fog-Cloud Architecture for IoT: A Comprehensive Review of Security and Privacy Challenges. *IEEE Internet of Things Journal*.2024. 11(2), P. 1205-1223.

6. Mishra B., & Kertesz A. The Use of MQTT in M2M and IoT Systems: A Survey. *IEEE Access*. 8. P. 201-210.

7. Tanwani A.. Advanced Network Topologies for Scalable IoT Deployments. *International Journal of Computer Networks*.2022. 15(1), P. 22-34.

8. Ta, Phuong Bac & Mafeni, Vitumbiko & Kim, Younghan. A Design of Workflow-Based Automated Failure Recovery Framework in IoT Edge Environment. *IEEE Access*. 2025. PP. 1-1.

9. Sadique K. M.. A Survey on IoT Security Protocols and Vulnerabilities. *IEEE Access*.2022. 10. P. 152-159.

10. Bucchiarone A. Model-Driven Engineering for the IoT: A Systematic Mapping Study. *IEEE Access*.2021. 9. P. 102-110.

11. ETSI EN 303 645 V2.1.1. (2021). Cyber Security for Consumer Internet of Things: Baseline Requirements. European Telecommunications Standards Institute.

12. Yuan B.. Design of Smart Home Control System Based on Finite State Machine. *Journal of Physics: Conference Series*. 2023. P. 12-25.

13. Kodali R. K., & Sorat, B. MQTT Based Home Automation System using ESP32. *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. 2021. P. 540-543.

14. Al-Ali A. R.. IoT-Based Smart Home Automation System: A Review of Topologies and Protocols. *IEEE Access*, 2022. 10. P. 234-250.

15. Introducing ESP32-C3. Espressif. URL: https://www.espressif.com/en/news/ESP32_C3 (дата звернення: 26.10.2025).

16. HT73xx-1 Low Power Consumption LDO Datasheet. URL: https://aliot.com.ua/pdf/ht7333.pdf (дата звернення: 26.10.2025).

17. Arduino IDE. *Arduino.* URL: https://itmaster.biz.ua/electronics/arduino/arduino-ide.html (дата звернення: 26.10.2025).