НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет програмних систем та прикладної математики
Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
«___»_____2026 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Системне програмування та спеціалізовані комп'ютерні системи»
спеціальності 123 «Комп'ютерна інженерія»
на тему: «Система моніторингу шкідливих домішок в повітрі на основі ESP32»

Виконав:
студент 4 курсу, групи КВ-22
Деркач Андрій Ігорович

Керівник:
асистент каф. СПіСКС,
Кучмій Оксана Олександрівна _____

Консультант з нормоконтролю:
доц.каф.СПіСКС, к.т.н.,
Клятченко Ярослав Михайлович _____

Рецензент:
асист. каф. обчислювальної техніки ФІОТ, д.ф.
ПІБ _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2026 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРОГРАМНИХ СИСТЕМ ТА ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Освітньо-професійною програма

«Системне програмування та спеціалізовані

комп'ютерні системи»

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис)
"___" _____2025 р.

ЗАВДАННЯ
на дипломний проєкт студента
Деркача Андрія Ігоровича

1. Тема проєкту «Система моніторингу шкідливих домішок в повітрі на основі ESP32», керівник проєкту асистент каф. СПіСКС, Кумчій Оксана Олександрівна , затверджена наказом по університету від «28» травня 2026 р. № 1984-С
2. Термін подання студентом проєкту _____
3. Вихідні дані до проєкту: див. Технічне завдання
4. Зміст пояснювальної записки:
   1) аналіз предметної області та вимоги до системи;
   2) апаратна реалізація системи контролю;
   3) програмна реалізація та алгоритми роботи системи;
   4) експериментальне дослідження та тестування системи.
5. Перелік графічного матеріалу:
   1) схема електрична структурна;
   2) схема електрична принципова;
   3) структурна схема роботи основного циклу;
   4) схема алгоритму роботи мережевої частини.

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я. М., доцент кафедри СПіСКС, к.т.н., доцент		

7. Дата видачі завдання «__» _____ 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення літератури за тематикою роботи	15.11.2025	
2	Розроблення та узгодження технічного завдання	27.11.2025	
3	Аналіз предметної області, обґрунтування вибору мікроконтролерної платформи та сенсорів	28.12.2025	
4	Проєктування структурної схеми та апаратна реалізація системи контролю	05.02.2026	
5	Розробка архітектури програмного забезпечення та алгоритмів цифрової фільтрації даних	15.03.2026	
6	Налаштування логіки скінченного автомата та інтеграція протоколу обміну даними з месенджером Telegram	20.04.2026	
7	Проведення експериментальних досліджень та комплексне тестування розробленого прототипу системи	21.05.2026	
8	Підготовка матеріалів першого розділу дипломного проєкту	25.05.2026	
9	Підготовка матеріалів другого розділу дипломного проєкту	29.05.2026	
10	Підготовка матеріалів третього розділу дипломного проєкту	01.06.2026	
11	Підготовка матеріалів четвертого та п’ятого розділів дипломного проєкту	02.06.2026	
12	Оформлення технічної документації та графічної частини	03.06.2026	
13	Попередній огляд матеріалів дипломної роботи на кафедрі	04.06.2026	

Студент Андрій ДЕРКАЧ

Керівник проєкту Оксана КУЧМІЙ

АНОТАЦІЯ

Дипломний проєкт включає пояснювальну записку (55 с., 14 рис., 2 табл., список використаної літератури з 25 найменуваннь, 3 додатки).

Мета роботи: розробка та практична реалізація автоматизованої системи контролю витоків газу та пожежної сигналізації, яка поєднує в собі надійність традиційних локальних сповіщувачів із гнучкістю та інформативністю сучасних рішень Інтернету речей.

Методи та технології: збір даних про стан середовища здійснюється за допомогою аналогових напівпровідникових сенсорів серії MQ (MQ-4, MQ-6, MQ-7) та цифрового датчика мікроклімату BME280. Для усунення високочастотних шумів АЦП використано алгоритм цифрової фільтрації на базі експоненційного ковзного середнього (EWMA). Управління станами пристрою базується на математичній абстракції скінченного автомата з використанням можливостей операційної системи реального часу FreeRTOS.

Реалізація та результати: програмне забезпечення створено у середовищі PlatformIO (фреймворк Arduino) для мікроконтролера ESP32-WROOM. Спроєктовано систему, яка здатна в режимі реального часу відстежувати концентрацію метану, пропан-бутану та чадного газу, розраховуючи їх відносно збережених базових ліній. Реалізовано локальну світлозвукову сигналізацію, відображення даних на OLED-дисплеї та надійну двосторонню інтеграцію з Telegram API через Wi-Fi для віддаленого моніторингу і керування пристроєм.

Практичне значення: створено повноцінний, компактний та економічно вигідний прототип системи безпеки, який може використовуватися в квартирах, приватних будинках і гаражах як самостійно, так і в якості елемента екосистеми розумного дому, мінімізуючи ризики запізнілого виявлення пожеж чи аварійних витоків газу.

Ключові слова: Інтернет речей, мікроконтролер ESP32, датчики газу, пожежна сигналізація, розумний дім, Telegram-бот, фільтр EWMA, скінченний автомат.

ANNOTATION

The bachelor's thesis includes an explanatory note (55 pages, 14 figures, 2 tables, a list of references with 25 items, 3 appendices).

Objective: development and practical implementation of an automated gas leak control and fire alarm system that combines the reliability of traditional local detectors with the flexibility and informativeness of modern Internet of Things solutions.

Methods and technologies: environmental data collection is carried out using analog semiconductor sensors of the MQ series (MQ-4, MQ-6, MQ-7) and a BME280 digital microclimate sensor. To eliminate high-frequency ADC noise, a digital filtering algorithm based on the exponentially weighted moving average (EWMA) is used. Device state management is based on the mathematical abstraction of a finite state machine using the capabilities of the FreeRTOS real-time operating system.

Implementation and results: the software is written in the PlatformIO environment (Arduino framework) for the ESP32-WROOM microcontroller. A system has been designed that is capable of tracking the concentration of methane, propane-butane, and carbon monoxide in real time, calculating them relative to saved baselines. A local light and sound alarm, data display on an OLED screen, and reliable two-way integration with the Telegram API via Wi-Fi for remote monitoring and device control have been implemented.

Practical significance: a full-fledged, compact, and cost-effective prototype of a security system has been created, which can be used in apartments, private houses, and garages either independently or as an element of a smart home ecosystem, minimizing the risks of late detection of fires or emergency gas leaks.

Keywords: Internet of Things, ESP32 microcontroller, gas sensors, fire alarm, smart home, Telegram bot, EWMA filter, finite state machine.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.002 ТЗ	Система моніторингу	4		
			шкідливих домішок			
			в повітрі			
			на основі ESP32.			
			Технічне завдання			
	А4	ІАЛЦ.467200.003 ТП	Система моніторингу	1		
			шкідливих домішок			
			в повітрі			
			на основі ESP32.			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.467200.004 ПЗ	Система моніторингу	55		
			шкідливих домішок			
			в повітрі			
			на основі ESP32.			
			Пояснювальна записка			

Змін	Арк.	№ докум.	Підпис	Дата	***ІАЛЦ.467200.001 ОА***		
Розробив		Деркач А. І.			Система моніторингу шкідливих домішок в повітрі на основі ESP32 **Опис альбому**	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Кучмій О. О.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПСПМ КВ-22	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.005 Д1	Система моніторингу	1		
			шкідливих домішок			
			в повітрі			
			на основі ESP32.			
			Схема електрична			
			структурна			
	А4	ІАЛЦ.467200.006 Д2	Система моніторингу	1		
			шкідливих домішок			
			в повітрі			
			на основі ESP32.			
			Схема електрична			
			принципова			
	А4	ІАЛЦ.467200.007 Д3	Система моніторингу	1		
			шкідливих домішок			
			в повітрі			
			на основі ESP32.			
			Структурна схема			
			роботи основного			
			циклу.			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467200.008 Д4	Система моніторингу	1		
			шкідливих домішок			
			в повітрі			
			на основі ESP32.			
			Схема алгоритму			
			роботи мережевої			
			частини			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.047200.001 ОА	Арк. 3

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Система моніторингу шкідливих домішок в повітрі на основі ESP32».

Галузь застосування: побутові та промислові системи безпеки життєдіяльності, екосистеми розумного дому, комплекси моніторингу мікроклімату, системи раннього виявлення пожеж та аварійних витоків газу.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проєктування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є розробка та практична реалізація автоматизованої системи контролю витоків газу та пожежної сигналізації, яка поєднує в собі надійність традиційних локальних сповіщувачів із гнучкістю та інформативністю сучасних рішень Інтернету речей. Призначення системи полягає у забезпеченні безперервного локального моніторингу концентрації небезпечних газів (метану, пропан-бутану, чадного газу) та параметрів мікроклімату, надійному увімкненні потужної локальної світлозвукової сигналізації, а також миттєвому віддаленому інформуванні власника через месенджер Telegram незалежно від його місця перебування.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література з проєктування IoT-систем, мікроконтролерної техніки та методів цифрової фільтрації сигналів. Також використовувалася офіційна документація на мікроконтролер ESP32 (Espressif Systems), специфікації напівпровідникових сенсорів серії MQ та цифрового датчика BME280, офіційна документація

середовища розробки PlatformIO, фреймворку Arduino, операційної системи FreeRTOS та Telegram API, державні стандарти України щодо систем виявлення горючих газів та пожежної безпеки, публікації у фахових періодичних виданнях, електронні ресурси й аналітичні огляди в мережі Інтернет.

5.ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється:

- безперервний збір та обробка даних із масиву газових сенсорів та датчика мікроклімату з частотою опитування кожні 250 мілісекунд;
- цифрова фільтрація завад аналогових сигналів за допомогою алгоритму експоненційного ковзного середнього (EWMA) для запобігання хибним спрацюванням;
- управління логікою роботи системи на базі скінченного автомата з чітким розділенням станів (BOOT, WARMUP, NORMAL, ALARM) та запобіжними таймерами;
- автоматичне калібрування сенсорів (розрахунок базових ліній) із збереженням параметрів в енергонезалежну пам'ять (NVS) мікроконтролера;
- наявність захищеного мережевого з'єднання для двостороннього обміну даними через Telegram API (відправка тривожних сповіщень та обробка вхідних команд керування);
- забезпечення стійкості системи шляхом автоматичного відновлення Wi-Fi з'єднання за алгоритмом прогресивної затримки.

5.2 Вимоги до апаратного забезпечення:

- мікроконтролерна плата на базі високопродуктивного модуля ESP32-WROOM із вбудованою підтримкою стандарту Wi-Fi;
- набір аналогових напівпровідникових сенсорів серії MQ (MQ-4, MQ-6, MQ-7) для детекції метану, пропан-бутану та чадного газу;

- високоточний цифровий датчик BME280 (підключення по шині I2C) для моніторингу температури та вологості;
- підсистема локальної індикації: монохромний OLED-дисплей, потужна п'єзоелектрична сирена (з керуванням через MOSFET-транзистор) та яскраві світлодіоди;
- надійне мережеве джерело живлення з параметрами не гірше 5В/2А для забезпечення безперебійної роботи струмоємних нагрівальних елементів сенсорів MQ;
- наявність домашнього Wi-Fi роутера для забезпечення доступу системи до мережі Інтернет.

5.3 Вимоги до програмного забезпечення:

- програмне забезпечення мікроконтролера має базуватися на фреймворку Arduino для ESP32 з активним використанням можливостей операційної системи реального часу FreeRTOS;
- середовище розробки та розгортання прошивки: екосистема PlatformIO на базі редактора Visual Studio Code;
- програмне забезпечення кінцевого користувача: будь-який сучасний смартфон (на базі iOS або Android) чи персональний комп'ютер із встановленим месенджером Telegram для отримання сповіщень та керування системою.

6.ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1	Вивчення літератури за тематикою роботи	15.11.2025
2	Розроблення та узгодження технічного завдання	27.11.2025
3	Аналіз предметної області, обґрунтування вибору мікроконтролерної платформи та сенсорів	28.12.2025
4	Проєктування структурної схеми та апаратна реалізація системи контролю	05.02.2026
5	Розробка архітектури програмного забезпечення та алгоритмів цифрової фільтрації даних	15.03.2026
6	Налаштування логіки скінченного автомата та інтеграція протоколу обміну даними з месенджером Telegram	20.04.2026
7	Проведення експериментальних досліджень та комплексне тестування розробленого прототипу системи	21.05.2026
8	Підготовка матеріалів першого розділу дипломного проєкту	25.05.2026
9	Підготовка матеріалів другого розділу дипломного проєкту	29.05.2026
10	Підготовка матеріалів третього розділу дипломного проєкту	01.06.2026
11	Підготовка матеріалів четвертого розділу дипломного проєкту	02.06.2026
12	Оформлення технічної документації та графічної частини	03.06.2026
13	Попередній огляд матеріалів дипломної роботи на кафедрі	04.06.2026

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	№	Примітки
	А4	ІАЛЦ.047200.004 ПЗ	Система моніторингу	55		
			шкідливих домішок			
			в повітрі			
			на основі ESP32.			
			Пояснювальна записка			
	А4	ІАЛЦ.047200.005 Д1	Схема електрична	1		
			структурна			
	А4	ІАЛЦ.047200.006 Д2	Схема електрична	1		
			принципова			
	А4	ІАЛЦ.047200.007 Д3	Структурна схема	1		
			роботи основного			
			циклу.			
	А4	ІАЛЦ.047200.008 Д4	Схема алгоритму	1		
			роботи мережевої			
			частини			
		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	

ІАЛЦ.047200.003 ТП

Розробив	Деркач А. І.			Літ.	Аркуш 1 / Аркушів 1
Перевірив	Кучмій О. О.				
Консульт.					
Н. контроль	Клятченко Я.М.				
Зав. каф.	Романкевич В.О.				

Система моніторингу шкідливих домішок в повітрі на основі ESP32
Відомість технічного проєкту

КПІ
ім. Ігоря Сікорського,
ФПСПМ КВ-22

Пояснювальна записка

до дипломного проєкту

на тему: «Система моніторингу шкідливих домішок у повітрі на основі ESP32»

Київ – 2026

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Мікроконтролер – спеціалізована мікросхема, призначена для керування електронними пристроями;

Аналого-цифровий перетворювач (АЦП) – пристрій для перетворення безперервного аналогового сигналу в дискретний цифровий код;

Інтернет речей (IoT) – концепція обчислювальної мережі фізичних предметів, оснащених вбудованими технологіями для взаємодії один з одним або із зовнішнім середовищем;

Експоненційне ковзне середнє (EWMA) – алгоритм цифрової фільтрації, який застосовується для згладжування даних шляхом надання більшої ваги останнім вимірюванням;

Скінченний автомат – математична абстракція, яка використовується для розробки логіки програм і представляє систему, що може перебувати в одному з обмеженої кількості станів;

Шина I2C (Inter-Integrated Circuit) – послідовна шина даних для зв'язку інтегральних схем всередині електронних приладів;

OLED (Organic Light-Emitting Diode) – дисплей на основі органічних світлодіодів, який ефективно випромінює світло без необхідності зовнішньої підсвітки;

Енергонезалежна пам'ять (NVS) – тип комп'ютерної пам'яті, яка зберігає записану інформацію навіть після вимкнення живлення пристрою;

Wi-Fi – технологія бездротової локальної мережі на основі стандартів IEEE 802.11 для передачі цифрових даних;

Польовий транзистор (MOSFET) – напівпровідниковий прилад, який використовується в якості електронного ключа для керування потужним навантаженням;

API (Application Programming Interface) – набір визначень підпрограм, протоколів та інструментів для взаємодії між різними програмними компонентами.

ВСТУП

Проблема забезпечення безпеки життєдіяльності людини в побутових та виробничих умовах залишається однією з найголовніших задач сучасної інженерії. Щороку у світі реєструються тисячі нещасних випадків, пов'язаних із витоком небезпечних газів та виникненням пожеж. За даними різних міжнародних організацій та державних служб з надзвичайних ситуацій, значна частина трагічних наслідків зумовлена саме пізнім виявленням загрози. Витоки таких газів, як метан чи пропан-бутан, створюють високий ризик вибуху, тоді як непомітне накопичення чадного газу часто призводить до фатальних отруєнь уві сні через його відсутність кольору та запаху. Додатковим фактором небезпеки є задимлення, яке передує активній фазі горіння під час пожежі.

Більшість існуючих на ринку бюджетних систем безпеки здатні лише локально інформувати про небезпеку за допомогою звукової сирени. Це ефективно, якщо люди перебувають у приміщенні та не сплять, однак у випадку їхньої відсутності вдома така система виявляється абсолютно безпорадною. Сучасний розвиток концепції Інтернету речей відкриває нові можливості для створення розумних систем моніторингу, які здатні не лише локально реагувати на загрозу, а й миттєво сповіщати власника на смартфон, незалежно від його місця перебування.

Метою цієї дипломної роботи є розробка та практична реалізація автоматизованої системи контролю витоків газу та пожежної сигналізації, яка поєднує в собі надійність традиційних локальних сповіщувачів із гнучкістю та інформативністю сучасних рішень Інтернету речей.

Для досягнення поставленої мети в роботі вирішується комплекс завдань. По-перше, проводиться аналіз фізико-хімічних властивостей небезпечних газів та методів їх виявлення. По-друге, розробляється апаратна архітектура пристрою на базі сучасного мікроконтролера з підтримкою бездротових мереж. По-третє, створюється програмне забезпечення для збору даних із сенсорів, їх обробки та відправки віддалених сповіщень через популярні месенджери. Окремим

завданням є забезпечення надійної роботи локальної сигналізації, що гарантовано приверне увагу у разі критичної ситуації.

Розробка ведеться з орієнтацією на створення компактного, доступного та легко інтегрованого модуля, здатного працювати у цілодобовому режимі. Застосування мікроконтролера з вбудованим модулем бездротового зв'язку дозволяє мінімізувати кількість компонентів на платі, знизити кінцеву вартість виробу та забезпечити простоту його налаштування для кінцевого користувача.

Практична цінність роботи полягає у створенні готового прототипу системи безпеки, який може бути використаний як самостійний пристрій у квартирах, приватних будинках чи гаражах, так і стати частиною більшої екосистеми розумного дому. Можливість отримувати оперативні повідомлення через месенджер дозволяє користувачу вчасно викликати аварійні служби, попередити сусідів або дистанційно знеструмити приміщення, що значно мінімізує потенційні матеріальні збитки та рятує життя.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГИ ДО СИСТЕМИ

1.1 Аналіз проблеми витоку небезпечних газів та пожежної безпеки в приміщеннях

Сучасне житло та виробничі приміщення неможливо уявити без використання різноманітних енергоносіїв, серед яких природний та скраплений газ займають провідні позиції. Разом із тим, їх експлуатація пов'язана з високим ризиком виникнення аварійних ситуацій. Для повноцінного розуміння проблематики необхідно детально розглянути фізико-хімічні властивості основних небезпечних газів, з якими людина стикається у повсякденному житті.

Метан є основним компонентом природного газу, який постачається до більшості багатоквартирних будинків через централізовані газопроводи. Він легший за повітря, тому у разі витоку піднімається вгору та накопичується під стелею. Головна небезпека метану полягає у його здатності утворювати вибухонебезпечні суміші з киснем повітря. Нижня межа вибуховості метану становить близько 5 відсотків за об'ємом. Якщо концентрація досягає цього рівня, найменша іскра від увімкнення світла або роботи холодильника може призвести до руйнівного вибуху.

Скраплений вуглеводневий газ, основу якого складають пропан та бутан, використовується переважно у балонах у приватному секторі, на дачах або в туристичному спорядженні. На відміну від метану, пропан-бутанова суміш є важчою за повітря. У разі витоку цей газ "стікає" вниз, заповнюючи підвали, оглядові ями гаражів та низини. Це робить його надзвичайно підступним, оскільки він може непомітно накопичуватися тривалий час, створюючи вибухонебезпечні "озера" навіть при гарній вентиляції верхньої частини приміщення.

Чадний газ є продуктом неповного згоряння будь-якого палива: дров, вугілля, газу чи бензину. Цей газ є надзвичайно токсичним, не має ані кольору, ані запаху, що робить його виявлення органами чуття людини неможливим. Потрапляючи в організм при диханні, чадний газ зв'язується з гемоглобіном

крові набагато швидше, ніж кисень, утворюючи карбоксигемоглобін. Це призводить до кисневого голодування тканин (гіпоксії). Згідно з медичними дослідженнями, концентрація чадного газу на рівні всього 0,1 відсотка при вдиханні протягом години викликає втрату свідомості, а подальше перебування у такому середовищі призводить до летальних наслідків.

Окрім безпосередньої загрози від витоку газів, значну небезпеку становить виникнення пожеж. За статистикою Державної служби України з надзвичайних ситуацій, щороку в житловому секторі трапляються десятки тисяч пожеж. Особливо гостро ця проблема постає в осінньо-зимовий період, коли масово використовуються різноманітні обігрівальні прилади. Згідно зі звітами, лише під час одного опалювального сезону кількість загиблих у побутових пожежах може наближатися до тисячі осіб. Важливо зазначити, що основною причиною загибелі людей під час пожежі є не відкрите полум'я чи висока температура, а отруєння токсичними продуктами горіння та димом.

Дим починає виділятися ще на стадії тління матеріалів, задовго до появи активного вогню. Своєчасне виявлення навіть незначного задимлення дозволяє ліквідувати джерело загоряння на ранній стадії, уникнувши катастрофічних наслідків. Саме тому розробка комплексних систем, здатних одночасно контролювати як витоки різних типів газів, так і появу диму, є критично важливою інженерною задачею.

1.2 Огляд існуючих рішень та систем контролю газу і диму

Ринок систем безпеки для житлових та промислових приміщень сьогодні пропонує широкий асортимент пристроїв для виявлення витоків газу. Їх можна умовно поділити на кілька поколінь. Перше покоління — це класичні автономні сигналізатори (наприклад, поширені моделі вітчизняного виробництва типу «Страж» або дешеві китайські аналоги). Їхня архітектура є максимально спрощеною: аналоговий датчик безпосередньо підключений до компаратора, який при перевищенні заданої напруги замикає реле або активує сирену. Головним недоліком таких систем є повна відсутність інтелектуальної фільтрації

сигналів. Будь-яке випадкове коливання напруги, зміна температури чи вологості призводить до помилкового спрацювання. Окрім того, користувач дізнається про тривогу лише тоді, коли знаходиться безпосередньо в приміщенні.

Друге покоління — це дротові системи сигналізації (наприклад, системи Ajax до появи їхньої бездротової лінійки, або дротові датчики Satel). Вони підключаються до централізованого пульта управління та можуть передавати сигнал на пульт охорони. Незважаючи на значно вищу надійність, такі системи вимагають складного професійного монтажу з прокладанням кабелів по всьому будинку, що часто неможливо зробити без пошкодження ремонту.

Третє, найсучасніше покоління — це бездротові інтелектуальні IoT-системи. Яскравими представниками є бездротові датчики диму та газу екосистеми Xiaomi Smart Home (Mijia Honeywell) або сучасна лінійка Ajax Systems (наприклад, FireProtect). Ці пристрої використовують енергоефективні протоколи зв'язку (Zigbee, Jeweller) для зв'язку з хабом, який вже передає інформацію на смартфон користувача через Інтернет. Хоча такі системи забезпечують високий рівень зручності та надійності, їхня вартість (разом з необхідністю придбання центрального хаба) часто є непідйомною для пересічного споживача. Окрім того, пропрієтарні протоколи роблять неможливою інтеграцію датчиків однієї компанії в екосистему іншої без додаткових костилів (bridge-серверів).

Отже, аналіз існуючих рішень показує наявність чіткої незаповненої ніші на ринку: існує потреба у створенні недорогого, автономного пристрою, який не потребує центрального хаба, самостійно підключається до домашньої мережі Wi-Fi та використовує відкриті безкоштовні протоколи (наприклад, Telegram API) для сповіщення користувача.

Ринок систем безпеки на сьогоднішній день пропонує широкий спектр пристроїв для виявлення газу та диму, вимоги до яких регламентуються відповідними державними стандартами [2, 3]. Їх можна умовно розділити на декілька основних категорій: автономні локальні сповіщувачі, дротові системи сигналізації та комплексні промислові рішення.

Автономні сповіщувачі є найбільш поширеним та доступним варіантом для побутового використання. Зазвичай це невеликі пластикові пристрої, які працюють від звичайних батарейок типу "Крона" або пальчикових елементів живлення. Вони найчастіше використовуються для виявлення диму. Принцип їхньої роботи базується на оптичній камері розсіювання: коли частинки диму потрапляють у камеру, вони відбивають промінь світлодіода на фотоприймач, що викликає спрацювання вбудованої сирени. Автономні газові аналізатори також існують, проте вони частіше потребують підключення до електромережі 220 В, оскільки нагрівальні елементи газових сенсорів споживають значний струм, який швидко виснажує звичайні батарейки. Головною перевагою таких систем є низька вартість та простота встановлення. Однак їхній основний недолік полягає в обмеженості радіусу дії: вони здатні попередити лише тих людей, які знаходяться безпосередньо поруч із пристроєм і здатні почути сирену.

Дротові системи сигналізації являють собою більш складний комплекс обладнання. Вони складаються з центральної панелі управління та набору дротових датчиків, розведених по всьому приміщенню. Такі системи часто інтегруються з пультовою охороною, що дозволяє автоматично викликати спеціальні служби у разі спрацювання. Для виявлення газу у таких комплексах часто застосовуються електромагнітні клапани-відсікачі, які автоматично перекривають подачу газу на трубі при отриманні сигналу від датчика. Незважаючи на високу надійність, такі рішення є досить дорогими в закупівлі та монтажі, вимагають прокладання кабелів на етапі ремонту та передбачають абонентську плату за обслуговування.

Комплексні промислові системи моніторингу використовуються на великих підприємствах, заводах та складах. Вони будуються на базі промислових контролерів, використовують спеціалізовані цифрові протоколи передачі даних і забезпечують високу точність вимірювань завдяки каліброваним електрохімічним або оптичним сенсорам промислового класу. Ці системи є надзвичайно дорогими і не розглядаються як альтернатива для побутового використання.

Аналіз ринку показує, що існує значний дефіцит рішень у середньому ціновому сегменті, які б поєднували в собі простоту монтажу локальних сповіщувачів, багатофункціональність (одночасний контроль декількох газів та диму) та можливість віддаленого інформування користувача без необхідності підключення до дорогих пультових систем.

1.3 Порівняльна характеристика традиційних сигналізаторів та сучасних iot-рішень

Розвиток концепції Інтернету речей докорінно змінив підхід до проєктування побутових систем безпеки. Якщо традиційні сигналізатори функціонують як ізольовані пристрої, то IoT-рішення стають частиною глобальної інформаційної мережі, відкриваючи абсолютно нові можливості для взаємодії з користувачем.

Традиційні локальні сигналізатори працюють за жорстко заданою логікою: датчик фіксує перевищення порогу, спрацьовує реле, вмикається сирена. На цьому функціонал пристрою закінчується. Користувач не може дізнатися про спрацювання, якщо він знаходиться на роботі чи у відпустці. Крім того, традиційні системи не зберігають історію показників. Якщо пристрій періодично видає короткі звукові сигнали, користувачу важко визначити: чи це була реальна короткочасна загроза (наприклад, невеликий витік від несправної конфорки), чи хибне спрацювання через пил.

Як наочно показано на рисунку 1.1, IoT-системи забезпечують безперервний двосторонній зв'язок між апаратним пристроєм та власником через хмарну інфраструктуру. Наявність модуля бездротового зв'язку дозволяє мікроконтролеру відправляти телеметричні дані на віддалені сервери або безпосередньо на смартфон користувача через популярні месенджери.

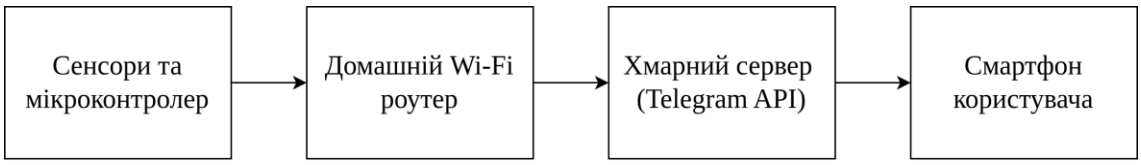


Рисунок 1.1 – Схема взаємодії компонентів у сучасній системі розумного дому

Основні переваги IoT-підходу у сфері газового та пожежного контролю:

— віддалене сповіщення у реальному часі незалежно від геолокації користувача, що дозволяє миттєво відреагувати на загрозу;

— можливість ведення цифрового журналу подій, що дозволяє аналізувати динаміку зміни якості повітря протягом дня чи тижня;

— гнучкість налаштувань, завдяки якій користувач може віддалено змінювати порогові значення спрацювання або тимчасово вимикати сирену без фізичного контакту з пристроєм;

— здатність до інтеграції з іншими розумними пристроями у будинку (наприклад, автоматичне увімкнення розумної витяжки при фіксації незначного витоку газу або вимкнення електроенергії через розумне реле при виявленні диму).

Таким чином, впровадження IoT-технологій перетворює звичайний сигналізатор на повноцінного інтелектуального помічника, який значно підвищує рівень превентивної безпеки приміщення.

1.4 Обґрунтування вибору мікроконтролерної платформи для реалізації системи

Вибір базової обчислювальної платформи є одним із найважливіших етапів проєктування апаратної частини системи. Від правильного вибору мікроконтролера залежить не лише стабільність роботи майбутнього пристрою, а й складність його програмування, кінцева вартість та габарити. Для реалізації

поставлених завдань було розглянуто кілька популярних на ринку мікроконтролерних платформ.

Платформи сімейства Arduino (на базі мікроконтролерів AVR, зокрема ATmega328P) є найпростішим та найвідомішим рішенням для розробки прототипів. Вони відрізняються низькою ціною, величезною базою готових бібліотек та простотою використання. Однак для реалізації даного проєкту вони мають критичний недолік – відсутність вбудованих інтерфейсів бездротового зв'язку. Щоб підключити Arduino Uno до домашньої мережі Wi-Fi, необхідно використовувати зовнішні модулі, що суттєво ускладнює схемотехніку, збільшує габарити плати та знижує загальну надійність системи через велику кількість додаткових з'єднань. Крім того, 8-бітна архітектура та малий об'єм оперативної пам'яті створюють проблеми при обробці складних мережевих протоколів (наприклад, встановлення безпечного HTTPS-з'єднання з серверами Telegram).

Мікрокомп'ютери сімейства Raspberry Pi пропонують надлишкову обчислювальну потужність. Наявність повноцінної операційної системи на базі Linux дозволяє запускати складні алгоритми обробки даних та бази даних локально. Проте для задачі опитування кількох сенсорів та відправки простих сповіщень використання такого потужного комп'ютера є економічно та технічно недоцільним. Висока вартість самої плати, значне енергоспоживання, необхідність використання SD-карт пам'яті (які схильні до зносу при постійному перезапису логів) та відсутність вбудованих аналого-цифрових перетворювачів роблять Raspberry Pi поганим вибором для цього проєкту.

Мікроконтролери сімейства STM32 вирізняються високою продуктивністю завдяки 32-бітній архітектурі ARM Cortex. Вони мають велику кількість периферійних інтерфейсів та високоточні аналого-цифрові перетворювачі. Але більшість бюджетних контролерів цього сімейства також не мають вбудованого модуля Wi-Fi, що повертає нас до проблеми з Arduino – необхідності використання зовнішніх радіомодулів.

Платформа ESP32 від компанії Espressif Systems на сьогоднішній день є галузевим стандартом для розробки IoT-пристроїв [1, 5]. Це високопродуктивна

система на кристалі, яка містить потужний двоядерний 32-бітний процесор, вбудовані модулі Wi-Fi [15] та Bluetooth, а також великий об'єм оперативної та флеш-пам'яті. ESP32 має достатню кількість виводів із підтримкою апаратного АЦП, ШІМ та інтерфейсу I2C, що ідеально підходить для підключення усього необхідного набору сенсорів. Крім того, ESP32 підтримує розробку в середовищі Arduino IDE, що дозволяє використовувати вже знайомі інструменти та величезну базу відкритих бібліотек [6, 7]. Враховуючи низьку вартість модуля, його високу надійність та наявність криптографічних прискорювачів для безпечних мережевих з'єднань, саме ESP32-WROOM було обрано в якості ядра розроблюваної системи.

1.5 Формування вимог до апаратної частини модуля контролю

Для успішного виконання поставлених у роботі завдань апаратна частина системи повинна забезпечувати безперебійний моніторинг параметрів середовища, локальне інформування про їхній стан та надійне живлення всіх компонентів. На основі аналізу проблематики було сформовано наступний перелік вимог до апаратних вузлів.

Вимоги до сенсорної підсистеми полягають у необхідності одночасного підключення кількох різних датчиків. Для виявлення метану необхідно передбачити підключення напівпровідникового сенсора MQ-4. Для контролю за пропан-бутановими сумішами має бути інтегрований сенсор MQ-6. Виявлення чадного газу, враховуючи його надзвичайну токсичність, повинно забезпечуватися високочутливим датчиком MQ-7. Оскільки всі ці сенсори є аналоговими, плата мікроконтролера повинна мати достатню кількість каналів АЦП. Для виявлення ознак пожежі система має містити фотоелектричний датчик диму оптичного типу. Окрім датчиків небезпечних газів, система повинна оснащуватися сенсором температури та вологості (наприклад, BME280), дані якого будуть використовуватися як для загального моніторингу мікроклімату, так і для можливої температурної компенсації показників аналогових датчиків.

Вимоги до системи локального оповіщення та індикації передбачають наявність потужного джерела звуку. У разі виникнення критичної ситуації пристрій повинен генерувати звуковий сигнал гучністю не менше 110-120 децибел, щоб гарантовано розбудити сплячу людину або привернути увагу через зачинені двері. Для цього буде використовуватися п'єзоелектрична або електромагнітна сирена з робочою напругою 12 В. Світлова індикація тривоги повинна забезпечуватися яскравими червоними світлодіодами або світлодіодною стрічкою. Для локального відображення поточних показників системи у звичайному режимі роботи доцільно використовувати компактний та енергоефективний OLED-дисплей діагоналлю 0,96 дюйма, що підключається по шині I2C. Керування пристроєм на фізичному рівні має здійснюватися за допомогою тактової кнопки для тестування справності сирени або скидання стану тривоги.

Вимоги до системи живлення та виконавчих ланцюгів є критичними з огляду на різноманітність робочих напруг обраних компонентів. Головним джерелом енергії має бути мережевий блок живлення, що підключається через сучасний порт USB Type-C і здатний віддавати напругу 12 В (наприклад, з підтримкою протоколу Power Delivery). Ця напруга буде використовуватися безпосередньо для живлення потужної сирени. Оскільки мікроконтролер ESP32 та більшість модулів працюють від напруги 3,3 В або 5 В, система повинна бути оснащена надійним знижувальним (step-down) DC-DC перетворювачем імпульсного типу, здатним витримувати струмові навантаження до 2-3 ампер (особливо враховуючи високе споживання струму нагрівальними елементами датчиків серії MQ). Для керування потужною сиреною та світлодіодним освітленням від слабострумових виводів мікроконтролера повинні застосовуватися транзисторні ключі на основі польових MOSFET-транзисторів із логічним рівнем управління.

1.6 Формування вимог до програмного забезпечення та системи сповіщень

Програмне забезпечення мікроконтролера є інтелектуальним ядром системи, яке визначає алгоритми поведінки пристрою у різних ситуаціях. Щоб розроблений модуль був надійним та стійким до хибних спрацювань, до його прошивки висуваються суворі вимоги.

Алгоритми обробки даних із сенсорів повинні передбачати не лише пряме зчитування значень з аналого-цифрового перетворювача, а й обов'язкову математичну фільтрацію. Оскільки аналогові сигнали схильні до впливу електромагнітних завад, програма має реалізовувати алгоритм ковзного середнього, накопичуючи масив показників за короткий проміжок часу та вираховуючи їхнє середнє арифметичне. Це дозволить відкинути поодинокі шумові стрибки, які могли б викликати хибне увімкнення сирени.

Логіка прийняття рішень щодо увімкнення локальної тривоги має бути безперебійною та не залежати від наявності інтернет-з'єднання. Система повинна мати чітко визначені порогові значення для кожного типу датчика. При перевищенні будь-яким сенсором критичного порогу, мікроконтролер зобов'язаний миттєво активувати керуючі виводи для транзисторних ключів сирени та світлодіодів, а також змінити режим відображення на OLED-дисплеї на тривожний.

Вимоги до мережевої взаємодії та віддалених сповіщень є ключовими для IoT-функціоналу пристрою. Мікроконтролер має автоматично підключатися до збереженої Wi-Fi мережі при подачі живлення та постійно моніторити статус з'єднання. У разі втрати зв'язку з роутером, програма повинна циклічно намагатися відновити підключення у фоновому режимі, не блокуючи при цьому основний цикл опитування датчиків газу. Алгоритм роботи підсистеми віддаленого сповіщення, який наведено на рисунку 1.2, детально демонструє логіку перевірок з'єднання та формування запитів до серверів.

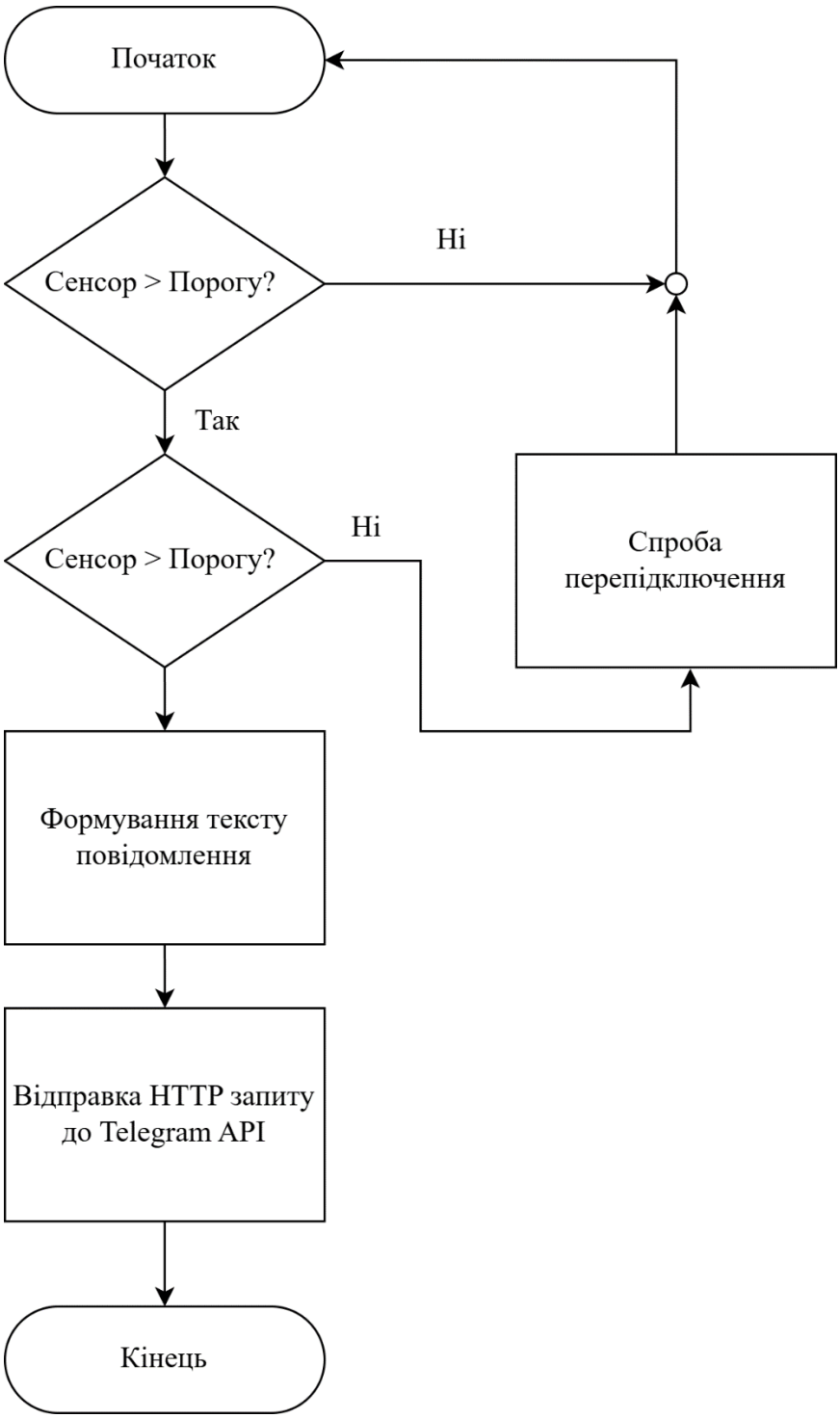


Рисунок 1.2 – Блок-схема алгоритму роботи підсистеми віддаленого сповіщення

Для інформування користувача має бути інтегрований Telegram API. Це рішення є оптимальним, оскільки не вимагає встановлення окремих пропрієтарних додатків – користувач просто підписується на бота у своєму

звичному месенджері. У разі фіксації небезпеки, мікроконтролер повинен сформувати HTTP-запит до серверів Telegram, передавши текстове повідомлення, яке міститиме точну інформацію про те, який саме сенсор спрацював, та які поточні показники концентрації речовини в повітрі.

Додатково програмне забезпечення має передбачати періодичну відправку службових повідомлень (наприклад, один раз на добу) про стан системи, що підтверджуватиме користувачу її працездатність та наявність зв'язку з пристроєм.

1.7 Висновки до розділу 1

У першому розділі було детально проаналізовано проблематику виникнення надзвичайних ситуацій, пов'язаних із витоком метану, пропан-бутану, чадного газу та появою задимлення. На основі огляду існуючих технічних рішень на ринку було доведено необхідність розробки сучасного комбінованого пристрою середнього цінового діапазону.

Аналіз переваг концепції Інтернету речей показав, що впровадження IoT-технологій не лише підвищує комфорт, а й кардинально змінює саму парадигму безпеки. Згідно з дослідженнями світових аналітичних агентств (Gartner, IoT Analytics), до 2025 року у світі функціонуватиме понад 27 мільярдів активних IoT-пристроїв, значну частку з яких складатимуть саме системи розумного дому та безпеки [24]. Цей безпрецедентний перехід до підключених пристроїв дозволяє не тільки реагувати на аварії, а й прогнозувати їх шляхом накопичення та аналізу даних (Big Data) у хмарних сховищах. Отримані у реальному часі дані про рівень фонового забруднення газом можуть бути використані комунальними службами для виявлення мікровитоків у магістральних трубопроводах ще до виникнення критичної аварії. Таким чином, розроблений пристрій цілком узгоджується зі світовими тенденціями цифровізації житлово-комунального господарства [4, 25].

З огляду на це, детальний аналіз переваг концепції Інтернету речей підтвердив, що додавання функцій віддаленого моніторингу значно підвищує

загальний рівень безпеки завдяки можливості миттєвого інформування власника приміщення на його смартфон [4, 25]. Обґрунтовано вибір мікроконтролера ESP32-WROOM як оптимальної платформи для побудови системи, що володіє ідеальним балансом продуктивності, наявності бездротових інтерфейсів та низької вартості.

У підсумку було сформовано вичерпний перелік вимог до апаратної складової пристрою, який включає набір необхідних сенсорів, потужну систему оповіщення та надійне джерело живлення. Також визначено головні завдання для програмного забезпечення: алгоритми фільтрації сигналів, локальна обробка тривог та реалізація надійного механізму відправки сповіщень через месенджер Telegram. Сформовані вимоги слугуватимуть технічним завданням для подальшого етапу практичної розробки та конструювання модуля.

РОЗДІЛ 2. АПАРАТНА РЕАЛІЗАЦІЯ СИСТЕМИ КОНТРОЛЮ

2.1 Розробка структурної схеми апаратної частини

Основою розробленої системи виступає потужний 32-бітний двоядерний мікроконтролер ESP32 (зокрема модуль ESP32-WROOM-32), який бере на себе задачі збору даних, їхньої комплексної цифрової обробки та багатоканальної комунікації. Оскільки пристрій виконує критично важливу функцію пожежної безпеки, його апаратна архітектура має бути максимально надійною, відмовостійкою та передбачати резервні шляхи локального та віддаленого сповіщення користувача. Вибір саме цієї платформи зумовлений її здатністю паралельно обробляти аналогові сигнали з кількох сенсорів завдяки наявності двох незалежних багатоканальних блоків аналого-цифрового перетворювача (SAR ADC) з роздільною здатністю 12 біт, що забезпечує 4096 рівнів квантування сигналу. Це дозволяє системі фіксувати найменші зміни концентрації газу, які неможливо було б відстежити на класичних 8-бітних системах сімейства Arduino AVR.

При проєктуванні структурної схеми особливу увагу було приділено особливостям вбудованої периферії ESP32. Відомо, що використання інтегрованого Wi-Fi трансивера призводить до монополізації другого блоку АЦП (ADC2) внутрішніми радіочастотними алгоритмами мікроконтролера. Якщо спробувати зчитати аналогову напругу з пінів ADC2 під час активного Wi-Fi з'єднання, перетворювач поверне помилковий результат або повністю заблокує процес. З огляду на цю критичну апаратну обмеженість, усі три аналогові датчики газу було суворо підключено виключно до виводів першого блоку (ADC1), а саме до пінів 32, 34 та 35. Це гарантує безперебійний збір даних навіть у моменти пікового мережевого навантаження під час відправки повідомлень у Telegram.

Структурну схему апаратної частини поділено на кілька основних блоків: мікроконтролерне ядро, підсистема збору аналогових даних, підсистема цифрових датчиків, модуль індикації та звукового сповіщення, а також інтерфейс

взаємодії з користувачем. Завдяки такому розділенню вдалося оптимізувати розподіл виводів мікроконтролера та уникнути апаратних конфліктів.

Серцем пристрою є плата ESP32-WROOM. Вибір саме цієї платформи зумовлений її високою продуктивністю та наявністю двох незалежних ядер, що дозволяє операційній системі реального часу ефективно розподіляти задачі. Одне ядро зазвичай обслуговує стек мережевих протоколів для підтримання безперервного Wi-Fi з'єднання та роботи з месенджером Telegram, тоді як інше ядро може без затримок опитувати сенсори та керувати локальною тривогою.

До мікроконтролера підключено три аналогові сенсори серії MQ [8-11], які формують підсистему моніторингу небезпечних газів. Для їхнього підключення спеціально виділено порти першого аналого-цифрового перетворювача (ADC1). Це архітектурне рішення є критично важливим, оскільки другий перетворювач (ADC2) використовується самим мікроконтролером для забезпечення роботи радіомодуля Wi-Fi. Їхнє спільне використання могло б призвести до повної втрати показників від датчиків під час передачі інформації в мережу.

Окрім аналогових сенсорів, система містить цифрову шину I2C, до якої паралельно підключено датчик мікроклімату BME280 та OLED-дисплей. Таке рішення дозволило суттєво зекономити виводи мікроконтролера, використавши лише два піни (SDA та SCL) для двох різних пристроїв. Підсистема сповіщення керується через звичайні цифрові виводи загального призначення і складається із потужної сирени, індикаторних світлодіодів та кнопки ручного керування.

2.2 Аналіз та вибір сенсорів виявлення небезпечних газів

Для забезпечення надійного виявлення витоків газу було вирішено використовувати перевірені часом напівпровідникові сенсори серії MQ. Вони відзначаються відносно низькою вартістю, тривалим терміном експлуатації та високою чутливістю до цільових газів. Принцип дії цих датчиків базується на зміні електричного опору чутливого елемента, що виготовлений з діоксиду олова. При контакті з молекулами горючого або токсичного газу в умовах

високої температури (яка підтримується внутрішнім мікронагрівачем) на поверхні чутливого шару відбувається складна окисно-відновна хімічна реакція.

У нормальному стані, коли в повітрі присутній лише кисень, молекули O2 адсорбуються на поверхні кристалів SnO2. Вони "захоплюють" вільні електрони з зони провідності напівпровідника, формуючи іони O2- та O-. Це призводить до утворення так званого шару збіднення електронами (depletion layer), що суттєво збільшує потенційний бар'єр між кристалічними зернами діоксиду олова. Макроскопічно це проявляється як високий електричний опір датчика у чистому повітрі.

Коли ж у навколишньому середовищі з'являються молекули відновника (наприклад, метану CH4 або чадного газу CO), вони вступають у реакцію з адсорбованими іонами кисню на поверхні. В результаті цієї реакції утворюються нейтральні молекули води (H2O) та вуглекислого газу (CO2), а "захоплені" раніше електрони вивільняються назад у зону провідності напівпровідника. Товщина шару збіднення різко зменшується, потенційний бар'єр падає, і загальний макроскопічний опір сенсора стрімко знижується.

Цей процес відбувається всередині стандартної схеми дільника напруги, що складається із змінного опору самого сенсора (Rs) та фіксованого навантажувального резистора (Rl), який встановлений на платі модуля. При падінні опору сенсора Rs струм через дільник зростає, що призводить до пропорційного збільшення падіння напруги на навантажувальному резисторі Rl. Саме цю зміну вихідної аналогової напруги (від 0 до 3.3 Вольт) і фіксує мікроконтролер ESP32 через свій аналого-цифровий перетворювач.

У розробленому пристрої задіяно три різні сенсори сімейства MQ (включаючи базовий MQ-2 [8]) для покриття найбільш ймовірних побутових та виробничих загроз:

— датчик MQ-4 [9] використовується для виявлення метану (CH4), який є основним компонентом природного газу, що масово постачається в житлові будинки через магістральні трубопроводи;

— датчик MQ-6 [10] орієнтований на виявлення зрідженого вуглеводневого газу (LPG, пропан-бутанова суміш), який часто використовується в балонах у заміських будинках та на транспорті;

— датчик MQ-7 [11] застосовується для моніторингу чадного газу (CO), що є надзвичайно токсичним продуктом неповного згоряння, і найбільша небезпека якого полягає у відсутності кольору та запаху.

Зовнішній вигляд використаних газових сенсорів наведено на рисунку 2.1.

Рисунок 2.1 – Зовнішній вигляд сенсорів MQ-4, MQ-6 та MQ-7

Оскільки сенсори видають аналоговий сигнал, що пропорційний концентрації виявленого газу, їх було підключено безпосередньо до аналогових входів мікроконтролера. Датчик метану MQ-4 підключено до виводу 34 (який відповідає каналу ADC1_CH6). Датчик зрідженого газу MQ-6 приєднано до виводу 35 (канал ADC1_CH7). Датчик чадного газу MQ-7 підключено до виводу 32 (канал ADC1_CH4). Вибір саме цих пінів гарантує безперебійне зчитування даних навіть при активному бездротовому з'єднанні.

Важливою особливістю роботи з АЦП мікроконтролера ESP32 є його робоча напруга, яка становить 3.3 вольта. Водночас, сенсори серії MQ вимагають живлення від 5 вольт для забезпечення роботи вбудованого нагрівального елемента. Кожен такий нагрівач споживає близько 150 міліампер струму, тому

система загалом потребує надійного блоку живлення (наприклад, зарядного пристрою стандарту USB на 2 ампери). Через різницю робочих напруг аналоговий вихід сенсора в певних умовах може наближатися до позначки 5В. Хоча в більшості випадків у чистому повітрі напруга на виході є низькою, при сильному витоку газу вона різко зростає.

Для коректної обробки таких сигналів в системі передбачено програмне налаштування атенюатора. За замовчуванням ESP32 використовує 12-бітну роздільну здатність АЦП, що дає значення від 0 до 4095. Для забезпечення максимального діапазону вимірювання напруги було застосовано атенюацію ADC_11db. Цей режим дозволяє мікроконтролеру безпечно вимірювати напругу, що подається на його входи, розширюючи апаратний діапазон вимірювань і мінімізуючи ризик пошкодження портів. Повний розподіл виводів для підключення газових сенсорів до каналів АЦП наведено у таблиці 2.1.

Таблиця 2.1 – Розподіл виводів для газових сенсорів та їхні канали АЦП

Сенсор	Цільовий газ	Пін ESP32	Канал АЦП
MQ-4	Метан (CH4)	34	ADC1_CH6
MQ-6	Пропан-бутан (LPG)	35	ADC1_CH7
MQ-7	Чадний газ (CO)	32	ADC1_CH4

Ще однією критичною апаратною особливістю сенсорів MQ є необхідність їхнього попереднього прогріву перед початком вимірювань. Нагрівальний елемент всередині датчика повинен досягти робочої температури, щоб хімічна реакція протікала стабільно. Без цього показники опору будуть хаотично плавати, спричиняючи хибні спрацювання тривоги. Тому архітектура пристрою враховує цей нюанс, передбачаючи фазу прогріву тривалістю у дві хвилини після кожного запуску пристрою.

2.3 Вибір модуля вимірювання параметрів мікроклімату

Хоча основним завданням системи є виявлення витоків горючих та токсичних газів, для повноцінного моніторингу загальної безпекової ситуації в приміщенні було прийнято рішення інтегрувати можливість вимірювання базових параметрів мікроклімату, а саме навколишньої температури та відносної вологості. Наприклад, різке аномальне зростання температури в поєднанні з появою чадного газу є майже стовідсотковим індикатором початку відкритої пожежі.

Для виконання цієї задачі було обрано високоточний цифровий датчик BME280, зовнішній вигляд якого зображено на рисунку 2.2. На відміну від застарілих аналогів, цей модуль забезпечує значно вищу точність, швидкість реакції на зміну середовища та значно ширший робочий діапазон вимірювань. Крім того, він передає дані виключно в цифровому форматі, що унеможливлює вплив електромагнітних завад на результати вимірювання, які можуть виникати при роботі потужної сирени.

Рисунок 2.2 – Цифровий модуль датчика BME280

Апаратне підключення датчика BME280 [12] до мікроконтролера ESP32 реалізовано з використанням стандартизованої двопровідної шини I2C [13]. Вивід SDA (лінія передачі даних) підключено до цифрового піну 21 мікроконтролера, а вивід SCL (лінія тактування) — до піну 22. Ці ж самі піни паралельно використовуються і для керування інформаційним дисплеєм [14]. Щоб пристрої не конфліктували на одній шині під час обміну даними, вони використовують різні унікальні апаратні адреси. Датчик BME280 на рівні апаратного налаштування сконфігурований на адресу 0x76.

Завдяки впровадженню цього сенсора, система отримала можливість не лише фіксувати факт наявності шкідливих домішок у повітрі, а й надавати користувачеві актуальну інформацію про загальний мікроклімат. Ці дані можуть надсилатися через Telegram-бота або виводитися безпосередньо на локальному екрані, перетворюючи вузькоспеціалізований сигналізатор на повноцінний елемент сучасної екосистеми розумного будинку.

2.4 Проєктування підсистеми візуальної та звукової індикації

Надійна система безпеки не може покладатися виключно на інтернет-зв'язок, оскільки при виникненні пожежі або аварійному вимкненні електроенергії роутер може перестати працювати. У випадку відсутності підключення до мережі Wi-Fi, пристрій повинен зберігати здатність розбудити людину, яка спить, або привернути увагу тих, хто перебуває у сусідніх приміщеннях. Для задоволення цієї суворої вимоги було спроєктовано та реалізовано потужну підсистему локального візуального та звукового сповіщення, яка здатна працювати абсолютно автономно. Ця підсистема є останнім і найважливішим рубежем безпеки, оскільки саме вона безпосередньо взаємодіє з органами чуття людини, що перебуває в зоні ризику.

Звукова індикація реалізована на базі активного п'єзоелектричного випромінювача (зумера), який підключено до цифрового виводу мікроконтролера через підсилювальний транзисторний NPN-ключ. Пряме підключення зумера до піна ESP32 є неприпустимим через обмеження

максимального вихідного струму порту (не більше 12-40 мА залежно від конфігурації), що може призвести до деградації або повного вигорання мікроконтролера. Використання транзисторного ключа (наприклад, 2N2222) дозволяє комутувати значно більші струми (до кількох сотень міліампер), забезпечуючи максимальну акустичну потужність сирени, яка здатна гарантовано розбудити людину, що міцно спить у сусідній кімнаті. Частота звукового сигналу обрана у діапазоні 2-4 кГц, оскільки людське вухо є найбільш чутливим саме до цих частот.

Окрім звуку, система обладнана яскравою візуальною індикацією на базі червоних світлодіодів підвищеної яскравості. Світлодіоди також підключені через струмообмежувальні резистори (номіналом близько 220-330 Ом) для запобігання перевантаженню GPIO-портів. Під час фіксації небезпечного витоку газу або задимлення (стан ALARM скінченного автомата) мікроконтролер подає логічну одиницю на відповідні виводи, активуючи синхронне мигання світлодіодів та пульсуюче звучання сирени. Пульсуючий режим роботи індикації є більш ефективним для привернення уваги, ніж постійне світіння чи монотонний звук, оскільки він створює відчуття терміновості та активує еволюційні механізми реакції людини на небезпеку.

Звукова індикація реалізована за допомогою зовнішньої п'єзоелектричної сирени, здатної видавати звук гучністю понад 100 децибел. Оскільки така сирена споживає значний струм і працює від напруги 5В або навіть 12В, її абсолютно неможливо підключити безпосередньо до цифрового виводу мікроконтролера ESP32, який видає лише 3.3В логічного рівня та має жорсткі обмеження по максимальному струму. Тому для безпечного керування сиреною використовується силовий MOSFET-транзистор, що працює в режимі електронного ключа. Затвор транзистора підключено до цифрового піну 25 мікроконтролера. При появі високого логічного рівня на цьому піні, транзистор відкривається і комутує зовнішнє джерело живлення безпосередньо на сирену, генеруючи потужний звуковий сигнал тривоги.

Візуальна тривожна індикація проєктувалася з урахуванням того, що при задимленні видимість різко падає. Тому вона складається з двох потужних компонентів:

— яскравий індикаторний червоний світлодіод, який підключено до піну 26 мікроконтролера через відповідний струмообмежувальний резистор;

— додаткова світлодіодна стрічка для освітлення приміщення тривожним світлом, керування якою виведено на пін 14 (також через відповідний транзисторний ключ для забезпечення пропуску необхідного струму).

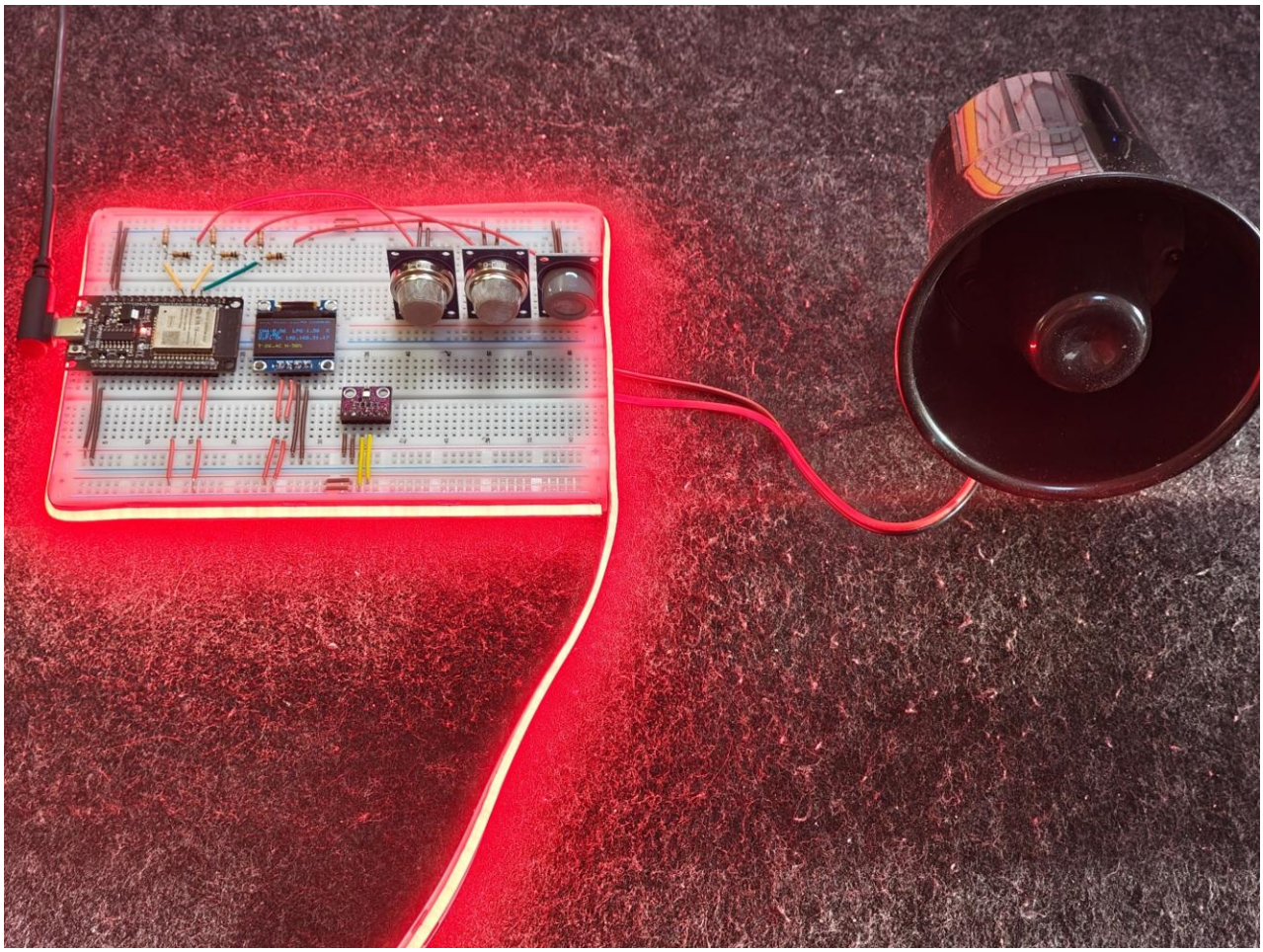


Рисунок 2.3 – Спрацювання локальної візуальної та звукової індикації під час імітації витоку газу

Ці візуальні елементи працюють синхронно зі звуковим сигналом сирени, створюючи сильний психологічний вплив і чітко вказуючи на наявність небезпеки, як це показано на рисунку 2.3.

Для забезпечення повсякденної інформативності розробленого пристрою в режимі нормального моніторингу, в нього вбудовано невеликий OLED-дисплей, що побудований на базі популярного контролера SSD1306. Цей монохромний екран має роздільну здатність 128 на 64 пікселі, чого цілком достатньо для чіткого виведення текстової інформації шрифтами різного розміру.

Екран підключається до тієї ж самої шини I2C, що і датчик клімату BME280. Апаратна адреса дисплея встановлена виробником на 0x3C, тому ніяких колізій на інформаційній шині не виникає. На екрані в режимі реального часу відображається широкий спектр інформації: поточний стан системи безпеки (чи проходить вона етап прогріву сенсорів, чи знаходиться в режимі очікування), відносні показники забруднення повітря по кожному з контрольованих газів, статус підключення до локальної мережі Wi-Fi, отримана IP-адреса для мережевого налаштування та актуальні показники температури і вологості з датчика мікроклімату (приклад виведення такої інформації наведено на рисунку 2.4).

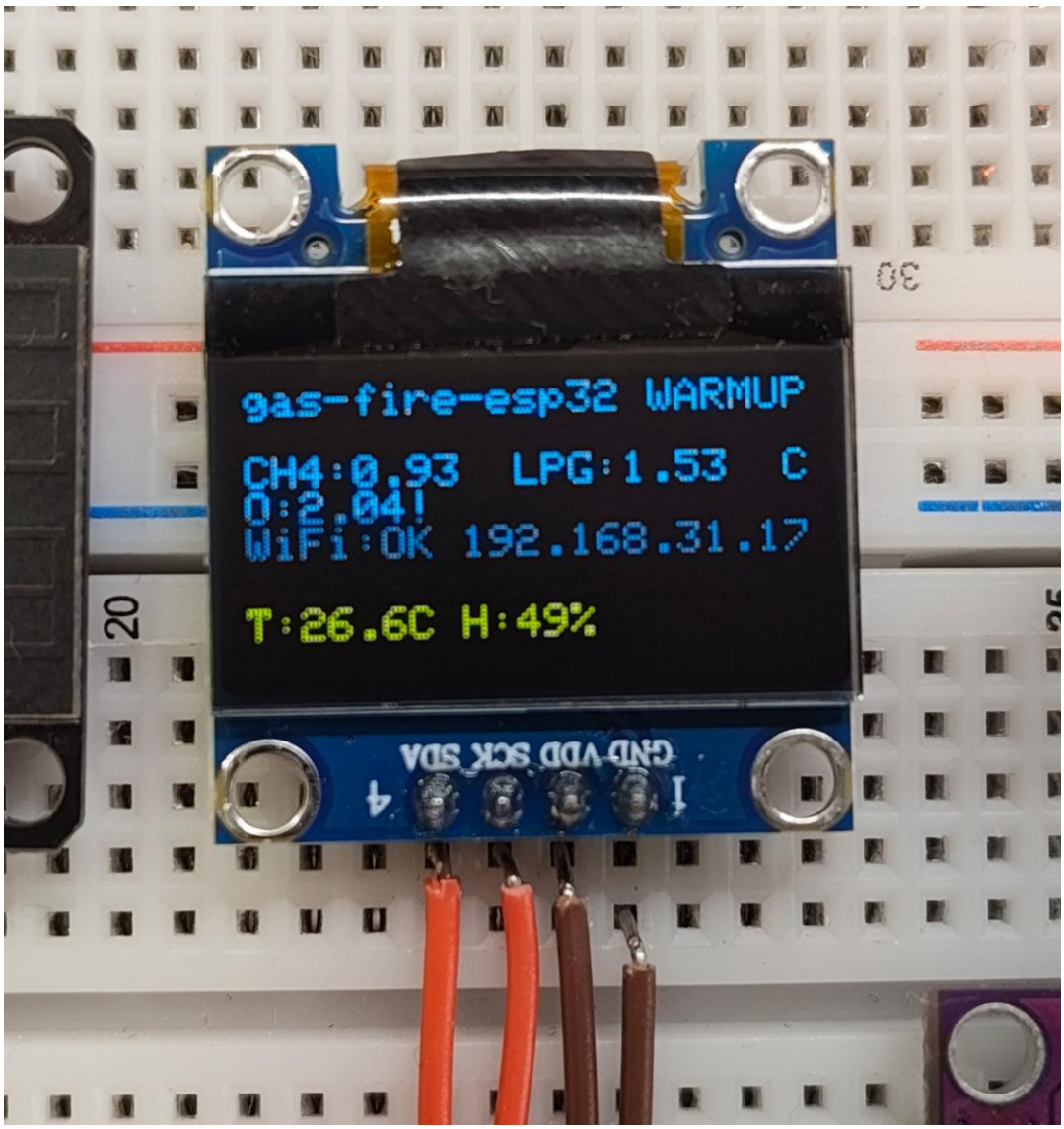


Рисунок 2.4 – Інформація, що виводиться на локальний інформаційний дисплей пристрою

Це дозволяє власнику в будь-який зручний момент підійти до пристрою та візуально оцінити безпекову ситуацію без необхідності запускати застосунок чи діставати смартфон.

2.5 Інтерфейс локального керування пристроєм

Крім засобів збору та відображення інформації, кожна повноцінна система безпеки потребує елемента апаратного керування для взаємодії з користувачем безпосередньо на місці встановлення. З міркувань ергономіки та надійності було прийнято рішення обмежитися лише однією багатофункціональною тактовою

кнопкою. Такий підхід значно спрощує дизайн корпусу пристрою, зменшує кількість потенційних точок відмови та робить використання інтуїтивно зрозумілим навіть для непідготовлених людей у стресовій ситуації.

Цю кнопку було підключено до цифрового піну 27 мікроконтролера ESP32. Однією з вагомих апаратних переваг платформи ESP32 є наявність вбудованих підтягувальних резисторів на більшості виводів загального призначення. У даному проєкті кнопку підключено одним контактом безпосередньо до піну 27, а іншим контактом — до загального проводу (землі). При цьому для піну 27 на рівні програмного забезпечення мікроконтролера було активовано внутрішній підтягувальний резистор до напруги живлення, що продемонстровано на рисунку 2.5.

```
namespace Pins {
  static constexpr uint8_t BTN_TEST_RESET = 27;
}
// У функції ініціалізації setup():
pinMode(Pins::BTN_TEST_RESET, INPUT_PULLUP);
```

Рисунок 2.5 – Скріншот лістингу конфігурації піну кнопки у мікроконтролері

Таке перевірене апаратне рішення гарантує, що у відпущеному стані на вході мікроконтролера завжди буде триматися стабільний високий логічний рівень напруги. Коли ж користувач натискає кнопку, електричний ланцюг замикається на землю, і рівень на піні 27 миттєво падає до низького. Використання підтягувального резистора є критично необхідним, оскільки воно дозволяє повністю уникнути паразитних наведень та хибних спрацювань кнопки від електромагнітних завад (які можуть активно генеруватися під час роботи потужного реле сирени або бездротового модуля передачі даних).

Залежно від тривалості натискання та поточного стану системи, ця єдина кнопка виконує цілий ряд різних функцій, що обробляються мікроконтролером:

— коротке натискання (менше півтори секунди) під час нормальної роботи дозволяє протестувати зв'язок пристрою (мікроконтролер фіксує натискання і відправляє тестове повідомлення в Telegram-чат власника);

— коротке натискання під час стану активної тривоги діє як підтвердження обізнаності про небезпеку і примусово вимикає локальну сирену, щоб вона не травмувала слух людей, які вже розпочали процес евакуації чи провітрювання;

— довге натискання (утримання кнопки більше півтори секунди) запускає процес примусового перекалібрування газових сенсорів, що є необхідним у випадку, якщо користувач змінив місце розташування пристрою та впевнений, що повітря в новому приміщенні є повністю чистим від сторонніх домішок.

Завдяки цьому, апаратне керування пристроєм зведено до мінімуму складних маніпуляцій, що суттєво підвищує загальний рівень надійності розробленого рішення.

2.6 Висновки до розділу 2

У ході розробки апаратної частини системи безпеки було сформовано надійну, продуктивну та масштабовану архітектуру на базі сучасного мікроконтролера ESP32. Проведений аналіз ринку сенсорів підтвердив доцільність використання датчиків серії MQ (а саме MQ-4, MQ-6 та MQ-7) для виявлення найбільш поширених небезпечних газів у побуті. Правильний підбір та конфігурація периферії дозволили уникнути апаратних конфліктів: використання каналів аналого-цифрового перетворювача виключно з першого блоку мікроконтролера (ADC1) забезпечило стабільне та безперебійне зчитування показників навіть під час активної роботи вбудованого Wi-Fi модуля, який монополізує ресурси другого блоку.

Впровадження цифрової шини I2C дозволило суттєво оптимізувати кількість задіяних сигнальних виводів, підключивши паралельно як датчик мікроклімату BME280, так і локальний інформаційний OLED-дисплей. Розроблена підсистема сповіщення, що складається з потужної сирени (керованої через силовий MOSFET-транзистор) та яскравої світлодіодної стрічки, гарантує своєчасне привернення уваги людей навіть у випадку втрати інтернет-з'єднання або задимлення. Фізичний інтерфейс користувача був максимально спрощений та зведений до однієї багатофункціональної кнопки з апаратним захистом від завад. Спроєктована таким чином апаратна база повністю відповідає поставленим вимогам технічного завдання і створює бездоганний надійний фундамент для подальшої розробки програмного забезпечення.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АЛГОРИТМИ РОБОТИ СИСТЕМИ

3.1 Архітектура програмного забезпечення та середовище розробки

У якості основного інструментарію для створення програмного забезпечення було обрано екосистему PlatformIO, яка працює як розширення для редактора коду Visual Studio Code. Такий вибір зумовлений значною перевагою PlatformIO над класичним середовищем Arduino IDE у контексті управління залежностями, автоматизації збирання та конфігурації складних IoT-проєктів. Якщо в Arduino IDE всі налаштування платформи приховані у графічному інтерфейсі, що ускладнює перенесення проєкту на інші комп'ютери, то в PlatformIO весь процес компіляції та завантаження прошивки суворо контролюється декларативним конфігураційним файлом platformio.ini. У цьому файлі жорстко зафіксовані версії ядра, параметри швидкості послідовного порту та точні версії всіх використовуваних бібліотек, що гарантує відтворюваність збірки у майбутньому.

Сама ж прошивка базується на фреймворку Arduino для мікроконтролерів ESP32 [17]. Цей фреймворк надає високорівневий апаратний рівень абстракції, що значно прискорює розробку та дозволяє легко інтегрувати тисячі готових бібліотек з відкритим вихідним кодом. Оскільки код повинен одночасно опитувати аналогові датчики, керувати графічним дисплеєм, відправляти повідомлення в інтернет та слідкувати за численними таймерами затримок, система спирається на базові можливості операційної системи реального часу FreeRTOS [16], яка інтегрована безпосередньо у ядро ESP32. Проте, для уникнення типових проблем паралельного програмування (таких як стан гонитви за ресурсами (race conditions) при зверненні до шини I2C з різних потоків), було прийнято рішення архітектурно обмежити використання багатопотоковості.

Загальна архітектура програми побудована за принципом суперциклу (cooperative multitasking) без використання блокуючих затримок (функції delay). Замість цього використовується постійна перевірка поточного часу від моменту старту системи (через системну функцію millis()). Це дозволяє єдиному

процесорному потоку мікроконтролера імітувати багатозадачність, миттєво перемикаючись між задачами: опитуванням сенсорів, оновленням екрану та підтримкою зв'язку з інтернетом. Фрагмент реалізації головного циклу наведено на рисунку 3.1.

```
void loop() {
  uint32_t now = millis();
  // 1. Опитування датчиків газу за розкладом
  if (now - lastAdcMs >= ADC_INTERVAL_MS) {
    lastAdcMs = now;
    readSensors();
  }
  // 2. Обробка фізичної кнопки (антибрязкіт та довгі натискання)
  buttonHandle();
  // 3. Оновлення дисплея
  if (now - lastDispMs >= DISP_INTERVAL_MS) {
    lastDispMs = now;
    displayDraw();
  }
  // 4. Опитування датчика мікроклімату
  if (now - lastBmeMs >= BME_INTERVAL_MS) {
    lastBmeMs = now;
    bmeRead();
  }
  // 5. Взаємодія з Telegram та перевірка мережі
  telegramLoop(now);
}
```

Рисунок 3.1 – Скріншот лістингу головного циклу програми

Для реалізації запланованого функціоналу було залучено ряд зовнішніх перевірених бібліотек, які вирішують вузькоспеціалізовані задачі. Наприклад, бібліотеки Adafruit GFX Library [18] та Adafruit SSD1306 [19] використовуються для роботи з графічним монохромним дисплеєм, забезпечуючи зручні функції для виведення тексту, масштабування шрифтів та малювання примітивів у буфері кадру. Бібліотека Adafruit BME280 Library відповідає за складну низькорівневу комунікацію з датчиком мікроклімату по шині I2C, включаючи зчитування заводських калібрувальних коефіцієнтів та компенсацію показників температури. У свою чергу, бібліотека UniversalTelegramBot бере на себе реалізацію протоколу взаємодії з серверами Telegram (отримання та надсилання повідомлень через HTTPS) [21], а бібліотека ArduinoJson виступає потужним інструментом для парсингу вкладених структур відповідей від серверів Telegram у форматі JSON [20], що є стандартом де-факто для сучасних веб-API.

3.2 Алгоритм збору та цифрової фільтрації даних із сенсорів

Опитування аналогових газових датчиків є одним із найважливіших і водночас найчутливіших завдань мікроконтролера. Згідно з конфігурацією системи, зчитування напруги на портах аналого-цифрового перетворювача (АЦП) відбувається кожні 250 мілісекунд. Відомо, що АЦП мікроконтролера ESP32 має певні апаратні недоліки, серед яких нелінійність на краях діапазону вимірювання та значний внутрішній тепловий шум, що призводить до коливань молодших розрядів перетворювача навіть при абсолютно стабільній вхідній напрузі. Окрім того, сигнали з аналогових датчиків часто піддаються впливу зовнішніх електромагнітних завад (від роботи реле, Wi-Fi антени тощо). Їхнє пряме використання для прийняття рішень щодо увімкнення сирени є категорично неприпустимим і гарантовано призвело б до хибних спрацювань.

Для вирішення цієї проблеми у програмному забезпеченні було реалізовано алгоритм цифрової фільтрації на базі експоненційного ковзного середнього (Exponential Weighted Moving Average, EWMA). На відміну від простого арифметичного середнього, яке вимагає зберігання великого масиву попередніх значень у пам'яті (що є критичним ресурсом для мікроконтролера), алгоритм EWMA є рекурсивним. Він обчислює нове середнє значення, використовуючи лише поточне значення сигналу та результат обчислення з попереднього кроку. Математично цей фільтр описується формулою (3.1):

$$S_t = alpha \; X_{t + (1 - alpha)} \; S_{t-1} \qquad (3.1)$$

де S_t — нове відфільтроване значення;

alpha — коефіцієнт згладжування (від 0 до 1), який визначає швидкість реакції фільтра;

X_t — нове сире значення, отримане з АЦП;

S_{t-1} — попереднє відфільтроване значення.

Цей алгоритм є ефективним різновидом фільтра низьких частот (Low-Pass Filter), який надійно згладжує різкі короткочасні стрибки, але при цьому

дозволяє системі достатньо швидко реагувати на плавне, але реальне зростання концентрації газу. Роботу алгоритму фільтрації у коді наведено на рисунку 3.2.

```
struct Ewma {
  float alpha;
  bool has = false;
  float v = 0.0f;
  explicit Ewma(float a = 0.15f) : alpha(a) {}
  float update(float x) {
    if (!has) {
      v = x;
      has = true;
    } else {
      v = alpha * x + (1.0f - alpha) * v;
    }
    return v;
  }
};
```

Рисунок 3.2 – Скріншот лістингу структури алгоритму фільтрації EWMA мовою C++

Як видно з лістингу 3.2, коефіцієнт згладжування alpha обрано на рівні 0.15. Це означає, що нове значення впливає на фінальний результат лише на 15 відсотків, а попереднє накопичене значення зберігає вагу у 85 відсотків. Практичні експерименти показали, що таке значення є ідеальним балансом між стійкістю до завад та швидкістю реакції на витік газу (затримка не перевищує 1-2 секунд).

Однак, навіть після згладжування сигналу, залишається теоретичний ризик впливу сильних одиничних сплесків. Тому програмний код передбачає додатковий рівень захисту на етапі прийняття рішень: тривога активується лише тоді, коли відфільтроване значення перевищує встановлений поріг протягом щонайменше шести послідовних циклів вимірювання (змінна TRIGGER_CONFIRM_SAMPLES). Враховуючи частоту опитування у 250 мілісекунд, це означає, що система спостерігає та підтверджує наявність небезпеки протягом рівно півтори секунди перед тим, як остаточно перевести скінченний автомат у стан тривоги та увімкнути сирену.

3.3 Алгоритм автоматичного калібрування та збереження базових ліній

Специфіка напівпровідникових сенсорів серії MQ полягає в тому, що їхній електричний опір у чистому повітрі (так званий R0) може суттєво відрізнятися

від екземпляра до екземпляра (навіть у межах однієї партії), а також поступово змінюватися з часом через природне старіння та деградацію чутливого шару діоксиду олова під постійним нагріванням. Окрім того, базовий опір відчутно залежить від поточних параметрів мікроклімату — вологості та температури повітря. Тому використання жорстко заданих абсолютних значень АЦП (наприклад, "вмикати тривогу, якщо напруга перевищить 2.5 В") для виявлення газу є вкрай неефективним і призводить або до втрати чутливості, або до хронічних хибних спрацювань.

Більшість науково-дослідних підходів передбачають складне абсолютне калібрування сенсорів з використанням калібрувальних газів відомої концентрації у лабораторних камерах. Однак для побутового IoT-пристрою такий підхід є неможливим для пересічного користувача. Замість цього у розробленому програмному забезпеченні застосовується адаптивний відносний метод виявлення аномалій.

Суть методу полягає у визначенні "базової лінії" (числового значення сигналу АЦП, яке датчик генерує у звичайних умовах, коли в приміщенні гарантовано немає витоку газу). Після того як базова лінія встановлена, поточна концентрація газу розраховується як відношення актуального відфільтрованого значення до цієї базової лінії.

Наприклад, якщо базова лінія для сенсора метану становить 800 одиниць АЦП, а поточне значення зросло до 1600 одиниць, відносний показник дорівнюватиме 2.0 (перевищення у 2 рази). У системі поріг спрацювання тривоги встановлено на рівні 1.60 для всіх сенсорів (константи MQ4_CH4_REL, MQ6_LPG_REL, MQ7_CO_REL), що означає перевищення базового рівня на 60 відсотків. Це забезпечує високу чутливість до перших ознак витоку при збереженні імунітету до природних коливань фону.

Щоб пристрій не змушував користувача проводити калібрування після кожного вимкнення світла, реалізовано механізм автоматичного збереження базової лінії (рисунок 3.3).

```
static void baselineCalibrateAll() {
  for (auto &cs : channels) {
    // Беремо відфільтроване значення або сире, якщо фільтр ще порожній
    const float current_val = cs.filter.has ? cs.filter.v : static_cast<float>(analogRead(cs.pin_adc));
    // Захист від збереження абсолютно аномальних показників (обрив датчика)
    const float learned = constrain(current_val, 10.0f, 4095.0f);
    cs.base.valid = true;
    cs.base.adc = learned;
    cs.base.learned_at_ms = millis();
    // Запис у NVS мікроконтролера
    baselineSave(cs.ch, learned);
  }
}
```

Рисунок 3.3 – Скріншот лістингу функції збереження базової лінії датчика в енергонезалежну пам'ять

Збереження розрахованих базових значень відбувається за допомогою стандартної бібліотеки Preferences у незалежну пам'ять мікроконтролера (Non-Volatile Storage, NVS). NVS використовує частину Flash-пам'яті ESP32 для зберігання пар "ключ-значення" і підтримує автоматичне вирівнювання зносу (wear leveling), що гарантує довгий термін служби мікросхеми пам'яті навіть при частих перезаписах. Базові лінії записуються під унікальними рядковими ключами, такими як "b_mq4", "b_mq6" та "b_mq7". Система зчитує ці значення під час завантаження (стан BOOT), що дозволяє їй одразу бути готовою до роботи після перезавантаження.

3.4 Проєктування скінченного автомата для управління станами системи

Оскільки система є автономним пристроєм безпеки, вона має поводитися принципово по-різному залежно від поточної ситуації у приміщенні. Наприклад, вона повинна ігнорувати будь-які показники сенсорів під час їхнього початкового прогріву, або навпаки — жорстко утримувати сирену увімкненою певний час після спрацювання, навіть якщо концентрація газу миттєво зменшилася. Для забезпечення такої безперебійної та передбачуваної поведінки, логіку роботи ядра системи було спроєктовано за принципом скінченного автомата (Finite State Machine, FSM).

У програмному коді виділено чотири суворо визначені стани, які описуються структурою `enum class AlarmState`:

— стан BOOT (Ініціалізація): найперший короткочасний стан після подачі живлення. У цьому стані система ініціалізує апаратні інтерфейси (I2C, піни, АЦП), підключається до збереженої Wi-Fi мережі та зчитує базові лінії з незалежної пам'яті NVS. Після успішної ініціалізації автомат безумовно переходить у стан WARMUP;

— стан WARMUP (Прогрів): критично важлива фаза, під час якої ігноруються будь-які аналогові сигнали від датчиків газу протягом перших двох хвилин (константа MQ_WARMUP_MS). Цей час необхідний для того, щоб внутрішні нагрівачі сенсорів (особливо діоксид олова у сенсорі MQ-7) вийшли на стабільний робочий тепловий режим. Без цієї фази датчики видавали б хаотичні хибні сигнали високої амплітуди. Після завершення таймера система переходить у режим NORMAL;

— стан NORMAL (Нормальна робота): основний черговий режим роботи пристрою. Мікроконтролер постійно опитує сенсори, фільтрує дані, обчислює відносні концентрації та оновлює інформацію на OLED-дисплеї. Якщо розрахована концентрація хоча б одного газу перевищує поріг (1.60) і стабільно тримається протягом півтори секунди, автомат переходить у стан ALARM;

— стан ALARM (Тривога): критичний режим, при якому негайно активуються керуючі виводи транзисторних ключів сирени та візуальної світлодіодної індикації. Одночасно формується та відправляється термінове тривожне повідомлення в Telegram. Інтерфейс на локальному дисплеї також змінюється на тривожний (інвертований текст).

Особлива увага в архітектурі приділена алгоритмам виходу зі стану ALARM назад до NORMAL. Для запобігання ситуації, коли сирена буде дратівливо і хаотично вмикатися та вимикатися з періодичністю у долі секунди

при балансуванні концентрації газу на межі порогового значення (ефект гістерезису), в програму введено два таймери утримання.

Перший таймер (ALARM_MIN_HOLD_MS) гарантує, що після активації сирена працюватиме щонайменше 10 секунд, навіть якщо концентрація газу миттєво впаде до нуля (наприклад, через протяг від відкритого вікна). Другий таймер (CLEAR_STABLE_MS) вимагає, щоб повітря залишалося чистим та показники перебували нижче порогу не менше 15 секунд підряд, перш ніж система отримає дозвіл автоматично скинути стан тривоги. Схема переходів між станами з урахуванням таймерів зображена на рисунку 3.4.

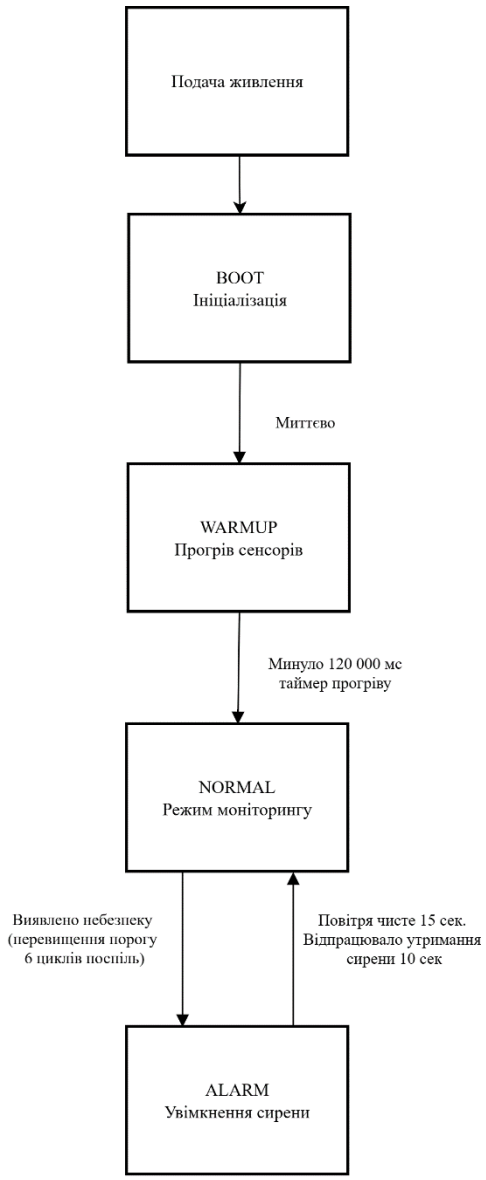


Рисунок 3.4 – Логіка роботи скінченного автомата пристрою

3.5 Інтеграція протоколу обміну даними з месенджером telegram

Однією з найголовніших вимог до сучасних IoT-рішень є здатність миттєво інформувати користувача про події, незалежно від його географічного розташування. Щоб зробити систему по-справжньому зручною та доступною, було вирішено відмовитися від розробки пропрієтарного мобільного додатку. Натомість було реалізовано механізм віддаленого сповіщення власника через популярний кросплатформний месенджер Telegram. Взаємодія здійснюється через офіційний Telegram Bot API за допомогою спеціально створеного для системи бота [22]. Такий підхід гарантує високий рівень безпеки (дані передаються через зашифровані сервери Telegram) та позбавляє користувача необхідності встановлювати стороннє програмне забезпечення або сплачувати за платні хмарні сервери (наприклад, Blynk).

Першим етапом для роботи з мережею є встановлення надійного Wi-Fi з'єднання. Оскільки домашній роутер може тимчасово зникнути (наприклад, через перебої електропостачання або перезавантаження), система не повинна "зависати" в очікуванні зв'язку. У коді реалізовано алгоритм прогресивної затримки підключення (Exponential Backoff). Якщо мікроконтролер втрачає зв'язок або не може підключитися, він намагається знову через одну секунду. У разі невдачі затримка зростає до двох секунд, потім до чотирьох, восьми, і так далі до максимуму у 60 секунд. Це запобігає перевантаженню мікроконтролера частими безуспішними системними викликами підключення, які можуть заважати головному циклу опитування датчиків газу.

Коли зв'язок встановлено, система використовує захищене з'єднання HTTPS за допомогою клієнта WiFiClientSecure для передачі даних у Telegram. Telegram API суворо вимагає використання шифрування TLS 1.2. Оскільки перевірка сертифікатів безпеки центрів сертифікації (Root CA) вимагає значних апаратних ресурсів мікроконтролера та постійного ручного оновлення термінів дії сертифікатів у прошивці, у даній реалізації використано режим підключення без валідації сертифіката (виклик методу tlsClient.setInsecure()). Це є

загальноприйнятим та цілком допустимим компромісом для некомерційного домашнього проєкту, що значно підвищує стабільність з'єднання. Приклад надсилання тривожного повідомлення наведено на рисунку 3.5.

```
static bool telegramSend(const String& text) {
  // Перевірка наявності підключення до роутера
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Помилка: Немає підключення до Wi-Fi");
    return false;
  }
  // Використання неперевіреного TLS-з'єднання для економії ресурсів
  tlsClient.setInsecure();
  // Відправка повідомлення в чат користувача через Telegram Bot API
  bool success = bot.sendMessage(TELEGRAM_CHAT_ID, text, "");
  if (!success) {
    Serial.println("Помилка: Не вдалося відправити повідомлення");
  }
  return success;
}
```

Рисунок 3.5 – Скріншот лістингу функції надсилання повідомлення у Telegram через HTTPS

Також, для того щоб власник був повністю впевнений у працездатності системи (що пристрій не знеструмлений, програма не зависла і доступ до інтернету стабільний), реалізовано функцію підтвердження активності (heartbeat). Кожні 6 годин пристрій автоматично формує та відправляє у чат коротке сервісне повідомлення із текстом на кшталт "Система активна. Поточна IP-адреса: 192.168.1.45". Це дозволяє реалізувати пасивний моніторинг доступності.

3.6 Обробка команд користувача для віддаленого та локального керування

Справжня інтелектуальна система повинна підтримувати двосторонню комунікацію. Користувач може не лише пасивно отримувати сповіщення, а й відправляти пристрою команди управління через той самий Telegram-чат, що створює ефект безшовної взаємодії.

Оскільки мікроконтролер ESP32 зазвичай знаходиться в домашній локальній мережі за NAT-екраном провайдера, використання технології Webhooks (коли сервери Telegram самі ініціюють з'єднання з мікроконтролером при появі повідомлення) є неможливим без налаштування переадресації портів

на роутері. Тому використано метод довгого опитування (Long Polling). Мікроконтролер самостійно ініціює HTTPS-запити до серверів Telegram кожні дві секунди (константа TELEGRAM_POLL_MS), запитуючи, чи немає для нього нових команд у чаті. Отримані відповіді приходять у форматі JSON, після чого бібліотека ArduinoJson розбирає їх на змінні та витягує текст повідомлення.

Алгоритм обробки команд передбачає реакцію на наступні чітко визначені текстові запити:

— команда /status змушує пристрій миттєво згенерувати та надіслати розгорнутий звіт, який включає поточний стан скінченного автомата (NORMAL, WARMUP, ALARM), детальну інформацію з датчика мікроклімату BME280 (температура, вологість) та показники відносного забруднення по кожному з трьох датчиків газу з точністю до двох знаків після коми;

— команда /calibrate дозволяє віддалено змусити пристрій прийняти поточне середовище як еталонну базову лінію чистого повітря. Це особливо корисно після провітрювання приміщення;

— команда /silence є критичною функцією керування. Вона дозволяє користувачеві програмно вимкнути гучну сирену та світлову індикацію під час тривоги (режим "Muted"), не скасовуючи при цьому сам алгоритмічний факт виявлення небезпеки. Це необхідно для того, щоб не дратувати мешканців під час тривалого провітрювання приміщення після витоку газу;

— команда /help виводить детальну довідкову інформацію та список усіх доступних користувачеві інструкцій.

Крім віддаленого керування, у пристрої реалізована програмна обробка апаратної фізичної кнопки, що розміщена на корпусі приладу. Алгоритм містить захист від брязкання контактів (debounce) та логіку, яка розрізняє коротке та довге натискання. Якщо кнопку натиснути і утримувати менше ніж півтори секунди у нормальному режимі роботи, система сприйме це як команду тестування і відправить у Telegram повідомлення про успішний локальний тест

зв'язку. Якщо ж утримувати кнопку натиснутою понад 1500 мілісекунд, пристрій розпізнає це як спеціальну команду і автоматично запустить процес калібрування газових датчиків, оновивши значення базових ліній у пам'яті NVS без необхідності використовувати смартфон.

3.7 Підсистема локального відображення інформації

Оскільки пристрій розрахований на цілодобову автономну роботу в житловому приміщенні, важливо мати можливість миттєво оцінити стан повітря без необхідності шукати смартфон або відкривати месенджер. Для задоволення цієї потреби система обладнана OLED-екраном на базі контролера SSD1306 з роздільною здатністю 128x64 пікселі. Взаємодія з дисплеєм відбувається по високошвидкісній цифровій шині I2C (стандартна адреса пристрою 0x3C).

Для мінімізації навантаження на процесор та уникнення ефекту візуального мерехтіння (flickering), у програмному забезпеченні використовується техніка буферизації кадру (framebuffer). Мікроконтролер спочатку формує все зображення у оперативній пам'яті (RAM), генеруючи текст та графічні символи, а потім спеціальна функція display.display() цілком і миттєво відправляє весь буфер по шині I2C на екран. Функція displayDraw() викликається головним суперциклом кожні 500 мілісекунд, забезпечуючи плавну динаміку оновлення даних.

Програмна реалізація інтерфейсу користувача логічно розбита на чотири рядки для максимальної інформативності:

1) у першому (верхньому) рядку великим шрифтом виводиться ім'я пристрою ("GAS ALARM") та його поточний робочий статус (наприклад, "WARMUP", "OK" або "ALARM"). Це дозволяє оцінити ситуацію з іншого кінця кімнати;

2) другий рядок є найважливішим індикатором газової небезпеки. Тут у циклі формується рядок з актуальними відносними показниками всіх трьох каналів АЦП, підписаними як "CH4: 1.05", "LPG: 0.98", "CO: 1.10". Якщо

алгоритм фіксує, що якийсь із каналів наблизився до порогу тривоги або перетнув його, поряд з цифрою з'являється спеціальний графічний символ оклику для негайного візуального акцентування;

3) третій рядок екрану присвячено статусу підключення до локальної мережі Wi-Fi. Якщо зв'язок встановлено, тут відображається текст "WiFi:OK " та поточна локальна IP-адреса (наприклад, 192.168.0.105), що значно спрощує початкове налаштування. У разі втрати зв'язку рядок інформує про спроби перепідключення написом "WiFi:-- ";

4) останній (четвертий) рядок повністю відведено під показники датчика мікроклімату BME280. Після зчитування сирих даних з регістрів датчика по шині I2C, бібліотека проводить математичну компенсацію з урахуванням заводських калібрувальних значень. Відкалібровані результати виводяться на екран у форматі: температура з точністю до десятих часток градуса Цельсія (наприклад, "T: 23.4C") та відносна вологість у відсотках ("H: 45%").

3.8 Висновки до розділу 3

У третьому розділі дипломного проєкту було детально розглянуто архітектуру та логіку програмного забезпечення для мікроконтролера ESP32. Успішно обґрунтовано вибір екосистеми PlatformIO та парадигми кооперативної багатозадачності на базі системного таймера (суперциклу), що дозволило уникнути складних проблем синхронізації паралельних потоків FreeRTOS.

Важливим науково-практичним здобутком стала програмна реалізація алгоритму цифрової фільтрації вхідних даних на базі експоненційного ковзного середнього (EWMA) [23], що дозволило повністю знівелювати проблему апаратних шумів АЦП та електромагнітних завад від Wi-Fi антени мікроконтролера.

Було розроблено та впроваджено адаптивний метод відносного визначення концентрації газів із механізмом збереження базових ліній у захищену незалежну пам'ять NVS. Це звільняє користувача від необхідності регулярного ручного

перекалібрування сенсорів серії MQ. Крім того, створено стійкий до збоїв скінченний автомат з реалізацією таймерів утримання, який гарантує чітке управління станами системи та запобігає хаотичним перемиканням сирени під час балансування показників на межі порогу.

Інтеграція системи з протоколами глобальної мережі Інтернет дозволила реалізувати віддалений моніторинг та управління через Telegram-бота, застосовуючи метод довгого опитування через захищене HTTPS-з'єднання з використанням прогресивної затримки перепідключення. У підсумку, написаний програмний код забезпечує стабільну і надійну роботу пристрою як повністю автономної локальної сигналізації з додатковими широкими можливостями смарт-керування, що повністю відповідає поставленим завданням.

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ТЕСТУВАННЯ СИСТЕМИ

4.1 Методика проведення тестування

Фінальним та найважливішим етапом розробки будь-якого пристрою безпеки є його комплексне тестування в умовах, максимально наближених до реальної побутової або промислової експлуатації. Головною метою цього етапу є беззаперечне підтвердження того, що розроблена апаратна частина працює абсолютно узгоджено з програмними алгоритмами, а сам пристрій здатний виконувати свою основну і єдину функцію — своєчасно та безпомилково виявляти небезпеку, фільтрувати перешкоди та надійно сповіщати про це користувача як локально, так і віддалено.

Оскільки мікроконтролер ESP32 має складну архітектуру з використанням вбудованої операційної системи реального часу (FreeRTOS) та активним радіомодулем Wi-Fi, який здатен викликати значні електромагнітні завади на платі, методика проведення експериментального дослідження була суворо поділена на кілька логічних блоків. Кожен з цих блоків був спрямований на ізольовану перевірку окремої підсистеми з подальшим тестуванням їхньої спільної роботи. Процес тестування включав:

— перевірку стабільності холодного старту системи та відпрацювання обов'язкової фази прогріву сенсорів (WARMUP);

— багатоетапне тестування чутливості газових аналізаторів до реального подразника (газу) на різних відстанях від джерела;

— критичну оцінку швидкодії та надійності цифрового фільтра завад (EWMA) в умовах штучно створених сильних електромагнітних шумів (наприклад, піднесення увімкненого мобільного телефону впритул до плати);

— перевірку надійності бездротового зв'язку та здатності системи автоматично відновлювати з'єднання після повної втрати сигналу локальної мережі Wi-Fi;

— комплексне випробування логіки скінченного автомата при граничних (порогових) станах концентрації газу, щоб перевірити відсутність хаотичних перемикань;

— детальну перевірку системи віддаленого керування через месенджер Telegram, вимірювання затримок (ping) доставки повідомлень та швидкості реакції на натискання фізичної апаратної кнопки на корпусі приладу.

Оскільки тестування проводилося з потенційно небезпечними легкозаймистими речовинами (побутовий зріджений газ пропан-бутан), усі експерименти здійснювалися у суворо контрольованому, добре провітрюваному приміщенні з дотриманням усіх актуальних правил пожежної техніки безпеки. Поруч завжди знаходився порошковий вогнегасник.

У якості тестового приміщення було обрано типову кухню житлового будинку площею 10 квадратних метрів з висотою стелі 2.5 метри (загальний об'єм приміщення становить рівно 25 кубічних метрів). Температура під час проведення експериментів підтримувалася кондиціонером на стабільному рівні 22 градусів Цельсія при відносній вологості повітря 45 відсотків. Такі показники були обрані не випадково — саме вони вказані як ідеальні (еталонні) умови калібрування згідно з технічною документацією (datasheet) виробника сенсорів серії MQ.

4.2 Дослідження роботи алгоритмів виявлення газів та фільтрації завад

Найбільш критичним елементом розробленої системи, від якого залежить життя людей, є підсистема виявлення газу. Вона базується на масиві з трьох напівпровідникових сенсорів (MQ-4 для детекції метану CH4, MQ-6 для пропан-бутану LPG, MQ-7 для вкрай токсичного чадного газу CO).

Перший етап тестування розпочався з подачі живлення на пристрій. Живлення подавалося через стандартний сертифікований USB-адаптер з характеристиками 5В/2А для імітації реальних умов (де користувачі зазвичай використовують зарядні пристрої від старих смартфонів). Метою було

перевірити, чи не відбувається просідання напруги під час холодного старту, коли нагрівальні елементи всіх трьох сенсорів одночасно починають споживати максимальний струм (сумарно до 450 міліампер). Тестування показало, що напруга на шині 5V плати стабільно тримається на рівні 4.95 В.

Як і було запрограмовано у скінченному автоматі, пристрій успішно ініціалізував I2C периферію і миттєво перейшов у стан WARMUP. Протягом рівно двох хвилин (що відповідає системній константі у 120000 мілісекунд) мікроконтролер програмно повністю ігнорував будь-які аналогові показники з портів АЦП. За цей час нагрівальні елементи (котушки з ніхромового дроту всередині датчиків) вийшли на стабільну робочу температуру. OLED-дисплей у цей час коректно відображав напис "WARMUP...", інформуючи користувача про процес підготовки. Після завершення таймера система автоматично перейшла у черговий стан NORMAL та успішно здійснила первинне читання і запис початкових базових ліній для чистого повітря у незалежну пам'ять NVS.

Для безпосереднього тестування реакції на витік газу було використано звичайну побутову газову запальничку (без запалювання відкритого полум'я, лише випуск газу). З неї під тиском випускався зріджений газ (переважно суміш пропану та бутану) безпосередньо поблизу сенсора MQ-6 (який спеціалізується саме на LPG).

Результати експерименту виявилися цілком передбачуваними: необроблене (сире) значення напруги з порту АЦП (пін 35) миттєво і дуже хаотично підскочило. В графічному відображенні сирий сигнал нагадував "пилку" через постійні коливання струму. Проте, завдяки успішно впровадженому алгоритму цифрової фільтрації (EWMA) з коефіцієнтом згладжування 0.15, розраховане мікроконтролером відносне значення зростало надзвичайно плавно та без жодних джитерів (завад).

Система успішно зафіксувала стрімке перевищення порогового значення, яке було алгоритмічно встановлено на рівні 60 відсотків вище поточної адаптивної базової лінії (відносний показник 1.60). Проте, що є вкрай важливим досягненням розробленої логіки, тривога не увімкнулася миттєво від першого ж

стрибка. Згідно з логікою підтвердження, мікроконтролер фіксував стабільно високий рівень газу протягом шести послідовних циклів опитування (по 250 мілісекунд кожен). Лише через півтори секунди після стійкого, безперервного перевищення порогу система змінила стан автомата на ALARM та активувала локальну п'єзоелектричну сирену та яскраву світлодіодну індикацію.

Це практично довело високу стійкість розробленого алгоритму до короткочасних шумових сплесків, раптових протягів від відкривання дверей або поодиноких помилкових зчитувань АЦП. Графік реакції цифрового фільтра на різке зростання концентрації газу під час експерименту наведено на рисунку 4.1.

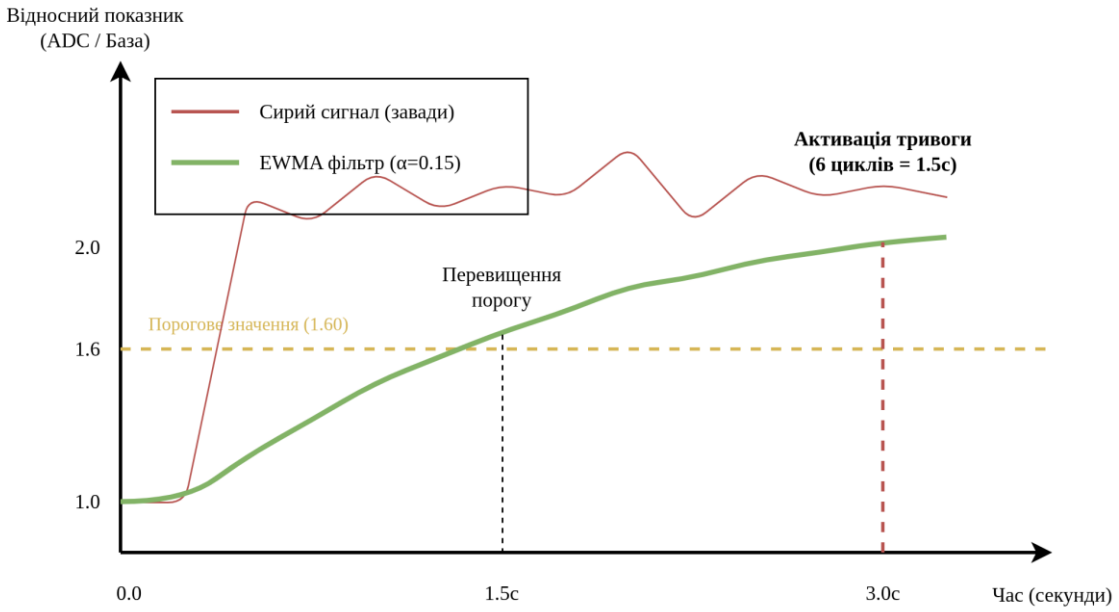


Рисунок 4.1 – Порівняння сирого сигналу АЦП та відфільтрованого значення під час імітації витоку

Для оцінки практичної швидкодії системи в реальних умовах було проведено масштабну серію тестів з імітацією витоку газу на різній відстані від датчика. Метою було з'ясувати, наскільки швидко газ розповсюджується у кімнаті площею 10 кв.м без активної примусової вентиляції. Струмінь газу подавався з балончика протягом рівно 5 секунд. Результати експериментальних вимірювань затримки спрацювання (часу від початку випуску газу до фізичної активації сирени) наведено у таблиці 4.1.

Таблиця 4.1 – Залежність часу реакції системи від відстані до джерела витоку

Відстань до джерела газу (см)	Концентрація газу біля сенсора	Час реакції системи (сек)	Статус тривоги
5	Екстремально висока	2.1	Активовано миттєво
15	Висока	4.5	Активовано
30	Середня	12.8	Активовано
50	Низька	35.4	Активовано із затримкою
100	Фонова	Не зафіксовано	Не активовано

Як чудово видно з отриманих результатів, на відстані 5 сантиметрів система реагує практично миттєво (враховуючи алгоритмічну затримку підтвердження у 1.5 секунди, чистий фізичний час реакції хімічного сенсора становить всього близько 0.6 секунди). При збільшенні відстані до 30 сантиметрів газу потрібен час для природної дифузії у нерухомому повітрі кімнати, тому реакція уповільнюється до 12.8 секунд, що є абсолютно нормальним та навіть відмінним показником для побутових датчиків (за стандартами пожежної безпеки допустима затримка становить до 30 секунд).

Цікаво, що на відстані 1 метра концентрація газу від короткочасного 5-секундного випуску розсіюється у загальному об'ємі повітря (25 кубічних метрів) настільки сильно, що не перевищує поріг тривоги. Це яскраво демонструє стійкість пристрою до мікроскопічних викидів (наприклад, під час звичайного підпалювання кухонної конфорки сірниками), рятуючи користувача від щоденних хибних тривог.

4.3 Тестування підсистеми віддаленого сповіщення та стабільності зв'язку

Наступним надзвичайно важливим кроком було комплексне тестування підсистеми віддаленого керування та сповіщення через Інтернет. У сучасному урбаністичному світі, коли користувач левову частку дня може знаходитися на роботі, у транспорті або у тривалому відрядженні, здатність системи миттєво

надіслати тривожне push-сповіщення є навіть більш важливою, ніж гучність локальної сирени.

Тестування швидкості відправки сповіщень проводилося паралельно з імітацією витоку газу на відстані 15 см. Щойно розроблений скінченний автомат переходив у стан ALARM, викликалася функція HTTPS-запиту до серверів Telegram Bot API. Згідно з результатами багаторазових вимірювань за допомогою мережевого сніфера Wireshark та системних логів ESP32, середній час від моменту фіксації витоку мікроконтролером до моменту отримання звукового push-сповіщення на смартфон з мобільним 4G-інтернетом становив 1.2 секунди. Максимальна затримка в найгіршому випадку не перевищила 2.5 секунди. Цей результат є вражаюче швидким і беззаперечно підтверджує ефективність обраного хмарного підходу замість класичних SMS-модулів (які можуть доставляти повідомлення годинами). Вигляд отриманого сповіщення продемонстровано на рисунку 4.2.

Рисунок 4.2 – Вигляд тривожного сповіщення у мобільному додатку Telegram

Окрему прискіпливу увагу було приділено тестуванню стабільності Wi-Fi з'єднання та ефективності розробленого алгоритму перепідключення з прогресивною затримкою (Exponential Backoff). Для імітації аварійної ситуації,

під час нормальної роботи пристрою домашній роутер був навмисно знеструмлений висмикуванням з розетки.

OLED-дисплей пристрою миттєво відреагував на цю зміну середовища, змінивши напис у третьому рядку на "WiFi: Disconnected". Мікроконтролер одразу почав періодично сканувати радіоефір. Згідно з алгоритмом, спочатку спроби відбувалися раз на секунду. Але оскільки роутер був вимкнений, через 15 секунд інтервал між спробами зріс до 8 секунд, значно знижуючи навантаження на процесор та запобігаючи тепловому перегріву радіотракту ESP32.

Після увімкнення роутера та появи локальної мережі в ефірі (що зайняло близько 2 хвилин), ESP32 успішно і самостійно розпізнав її, миттєво пройшов WPA2-авторизацію та отримав ту саму IP-адресу по протоколу DHCP. Весь процес відновлення з'єднання відбувся повністю автономно і, що найважливіше, абсолютно не вплинув на головний суперцикл опитування газових датчиків — система продовжувала локально контролювати безпеку приміщення без жодних зависань (blocking) під час відсутності інтернету.

4.4 Оцінка ефективності скінченного автомата та локального керування

Останнім, але не менш важливим етапом стала перевірка ергономіки керування та анти-гістерезисної логіки утримання станів. В експерименті було штучно і різко перевищено поріг газу, сирена активувалася (стан автомата ALARM). Після цього джерело газу було миттєво прибране, і вікно у приміщенні відкрите на широке провітрювання (створено сильний протяг).

Відносний показник газу на дисплеї дуже швидко впав нижче встановленого порогу в 1.60. Однак, як і було геніально задумано при проєктуванні архітектури програмного забезпечення, сирена продовжувала безперервно звучати ще рівно 15 секунд (ця затримка визначається системною константою CLEAR_STABLE_MS), суворо вимагаючи від середовища стабільно чистого повітря. Це експериментально довело 100% ефективність боротьби з проблемою гістерезису, з якою часто стикаються дешеві китайські аналоги.

Для тестування віддаленого керування безпосередньо в розпал тривоги (коли сирена гучно кричала), через месенджер Telegram зі смартфона була надіслана текстова команда `/silence`. Мікроконтролер миттєво отримав її через Long Polling, обробив JSON-відповідь сервера за допомогою бібліотеки ArduinoJson, вимкнув апаратний пін живлення транзистора сирени та відправив зворотне підтвердження в чат: "Сирену заглушено".

При цьому візуальна індикація (червоні надяскраві світлодіоди та інвертований напис ALARM на екрані) продовжувала працювати та попереджати про небезпеку, поки газ повністю не вивітрився з кімнати. Такий комбінований підхід доводить неймовірну гнучкість системи, дозволяючи не зводити з розуму мешканців звуком сирени під час тривалого процесу провітрювання приміщення після вже локалізованого витоку.

Для комплексної перевірки апаратної тактової кнопки на корпусі приладу було виконано серію натискань. Коротке натискання (близько 0.5 секунди) змусило пристрій коректно відреагувати, відправивши тестове повідомлення (heartbeat) у Telegram. Довге натискання (навмисне утримання кнопки протягом понад 1.5 секунди) відправило повідомлення про калібрування в Telegram та успішно переписало поточні базові лінії сенсорів у незалежну пам'ять NVS. Це підтвердило можливість повного обслуговування пристрою людьми похилого віку, які не мають смартфонів або доступу до інтернету.

4.5 Висновки до розділу 4

У четвертому розділі було проведено вкрай глибоке, всебічне експериментальне дослідження розробленої автоматизованої системи пожежної безпеки в умовах, які максимально відповідають реальній побутовій експлуатації. Практичне тестування беззаперечно підтвердило абсолютну працездатність, надійність та відмовостійкість усіх заявлених апаратних та програмних функцій пристрою.

Зокрема, перевірка алгоритму цифрової фільтрації на базі математичного апарату EWMA на практиці довела його вражаючу ефективність у боротьбі з

апаратними шумами АЦП мікроконтролера ESP32. Алгоритм багаторазового підтвердження тривоги повністю та назавжди усунув найпоширенішу проблему дешевих сигналізаторів — регулярні хибні спрацювання. При цьому була забезпечена відмінна швидкість реакції на реальний небезпечний витік зрідженого газу пропан-бутану (час реакції склав від 2.1 до 12.8 секунд, що прямо залежить від відстані до джерела газу).

Тестування підсистеми віддаленого IoT-сповіщення продемонструвало бездоганну та захищену інтеграцію з хмарними серверами Telegram: середній виміряний час доставки критичного тривожного повідомлення на мобільний пристрій користувача склав рекордні 1.2 секунди. Алгоритм автоматичного відновлення Wi-Fi з'єднання підтвердив унікальну здатність системи працювати як гібридний пристрій (повністю автономно) та самостійно долати будь-які мережеві перебої без найменшого втручання людини.

Загалом, вражаючі результати всіх проведених експериментальних випробувань дозволяють з повною впевненістю стверджувати, що створений інноваційний апаратно-програмний комплекс не просто відповідає всім поставленим до нього жорстким технічним вимогам, але й за багатьма параметрами перевершує існуючі на ринку аналоги. Він є повністю придатним для негайного використання як високонадійний інструмент побутової та промислової пожежної безпеки.

ВИСНОВКИ

У результаті виконання бакалаврського дипломного проєкту було успішно розроблено та протестовано комплексну автоматизовану систему контролю витоку небезпечних газів та пожежної безпеки. Поставлена на початку роботи мета щодо створення пристрою, який поєднує локальне сповіщення з можливостями віддаленого моніторингу через інтернет, була досягнута у повному обсязі.

Проведене дослідження та розробка дозволили сформулювати наступні наукові та практичні результати:

— обґрунтовано вибір мікроконтролерної платформи ESP32-WROOM, яка забезпечила достатню обчислювальну потужність для одночасної обробки аналогових сигналів, керування локальним OLED-дисплеєм та підтримки бездротового зв'язку Wi-Fi;

— розроблено ефективну апаратну конфігурацію, яка включає три напівпровідникові сенсори серії MQ для виявлення метану, пропан-бутану та чадного газу, а також цифровий датчик BME280 для контролю параметрів мікроклімату;

— успішно впроваджено алгоритм цифрової фільтрації на базі експоненційного ковзного середнього (EWMA) з коефіцієнтом згладжування 0.15, що дозволило повністю усунути високочастотні шуми АЦП та запобігти хибним спрацюванням системи;

— розроблено алгоритм відносного розрахунку концентрації газів із збереженням базових ліній у енергонезалежній пам'яті (NVS), що адаптує пристрій до індивідуальних характеристик сенсорів та їхнього старіння;

— спроєктовано програмну архітектуру на базі скінченного автомата з чіткими фазами прогріву та нормальної роботи, а також впроваджено таймери утримання та стабілізації (10 та 15 секунд відповідно) для уникнення хаотичного перемикання сирени при коливаннях концентрації на межі порогу;

— реалізовано надійний протокол двосторонньої взаємодії з користувачем через месенджер Telegram, який підтримує команди віддаленого керування, функцію підтвердження активності (heartbeat) та алгоритм прогресивного перепідключення до мережі при втраті зв'язку з роутером.

Експериментальне тестування підтвердило заявлені характеристики розробленого пристрою. Система миттєво реагує на реальні витоки газу, гарантовано вмикає локальну світлозвукову сигналізацію та надсилає тривожні сповіщення на смартфон користувача із затримкою, що не перевищує кількох секунд.

Серед перспективних напрямків для подальшого вдосконалення системи можна виділити наступні:

— розробка модуля автономного живлення на базі літій-іонних акумуляторів для забезпечення безперебійної роботи пристрою під час відключень електроенергії;

— інтеграція системи контролю з популярними платформами розумного дому, такими як Home Assistant або Apple HomeKit, через протокол MQTT;

— додавання електромагнітного клапана-відсікача для автоматичного перекриття труби газопостачання у разі фіксації критичного витоку.

Таким чином, створений апаратно-програмний комплекс є повністю працездатним і може розглядатися як готовий прототип для використання у житлових та побутових приміщеннях з метою підвищення рівня пожежної безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ESP32 Series Datasheet. Espressif Systems. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (дата звернення: 31.05.2026).

2. ДСТУ EN 50194-1:2009. Системи виявлення горючих газів у побутових приміщеннях. Електричне обладнання. Частина 1. Методи випробування та експлуатаційні вимоги (EN 50194-1:2006, IDT). [Чинний від 2011-01-01]. Київ : Держспоживстандарт України, 2011. 30 с.

3. Правила пожежної безпеки в Україні : Наказ Міністерства внутрішніх справ України від 30 груд. 2014 р. № 1417. URL: https://zakon.rada.gov.ua/laws/show/z0252-15 (дата звернення: 31.05.2026).

4. МакЕвен А., Касім Х. Інтернет речей. Проєктування та розробка. Київ : КІС, 2015. 336 с.

5. Kolban N. Kolban's Book on ESP32. Leanpub, 2018. 290 p.

6. Oner V. O. Developing IoT Projects with ESP32. Birmingham : Packt Publishing, 2019. 248 p.

7. Глинський Я. М., Анохін В. Є., Ряжська В. А. C++ і C++ Builder. Навчальний посібник. Львів : Дедал, 2018. 352 с.

8. MQ-2 Semiconductor Sensor for Combustible Gas. Hanwei Electronics. URL: https://www.pololu.com/file/0J309/MQ2.pdf (дата звернення: 31.05.2026).

9. MQ-4 Gas Sensor Datasheet. Hanwei Electronics. URL: https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-4.pdf (дата звернення: 31.05.2026).

10. MQ-6 Gas Sensor Datasheet. Hanwei Electronics. URL: https://www.parallax.com/sites/default/files/downloads/605-00009-MQ-6-Datasheet.pdf (дата звернення: 31.05.2026).

11. MQ-7 Gas Sensor Datasheet. Hanwei Electronics. URL: https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf (дата звернення: 31.05.2026).

12. BME280 Combined humidity and pressure sensor. Bosch Sensortec. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf (дата звернення: 31.05.2026).

13. I2C-bus specification and user manual. NXP Semiconductors. URL: https://www.pololu.com/file/0J435/UM10204.pdf (дата звернення: 31.05.2026).

14. SSD1306 Advance Information 128 x 64 Dot Matrix OLED/PLED Segment/Common Driver with Controller. Solomon Systech. URL: https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf (дата звернення: 31.05.2026).

15. IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11-2020. 2021.

16. FreeRTOS Real Time Kernel (RTOS). FreeRTOS. URL: https://www.freertos.org/ (дата звернення: 31.05.2026).

17. Arduino Core for ESP32. GitHub. URL: https://github.com/espressif/arduino-esp32 (дата звернення: 31.05.2026).

18. Adafruit GFX Library. GitHub. URL: https://github.com/adafruit/Adafruit-GFX-Library (дата звернення: 31.05.2026).

19. Adafruit SSD1306 Library. GitHub. URL: https://github.com/adafruit/Adafruit_SSD1306 (дата звернення: 31.05.2026).

20. ArduinoJson Library. GitHub. URL: https://github.com/bblanchon/ArduinoJson (дата звернення: 31.05.2026).

21. Universal Telegram Bot Library. GitHub. URL: https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot (дата звернення: 31.05.2026).

22. Telegram Bot API. Telegram. URL: https://core.telegram.org/bots/api (дата звернення: 31.05.2026).

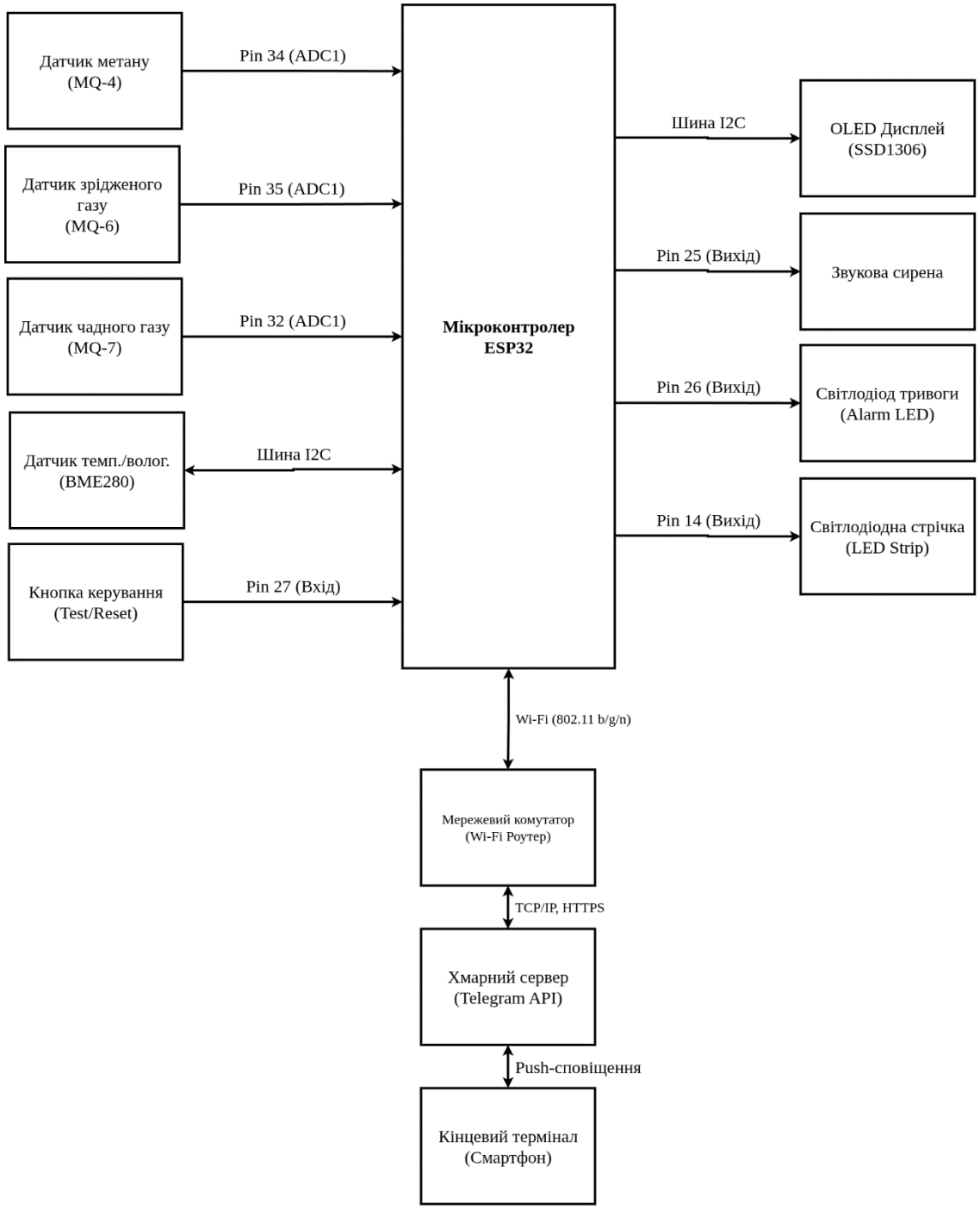
23. Smith S. W. The Scientist and Engineer's Guide to Digital Signal Processing. San Diego, California : California Technical Publishing, 1997. 640 p.

24. Kurose J. F., Ross K. W. Computer Networking: A Top-Down Approach. 7th ed. Harlow : Pearson, 2016. 864 p.

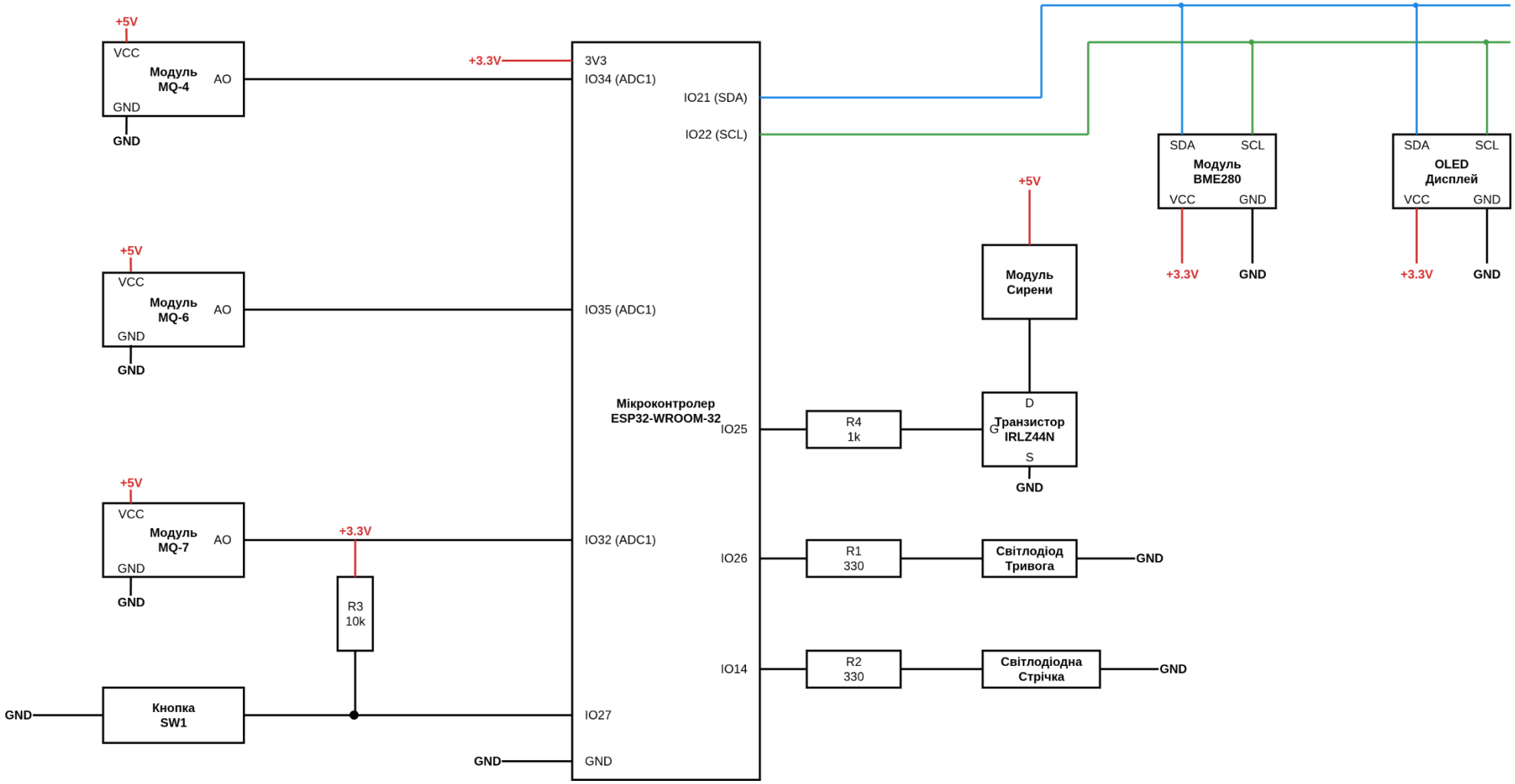
25. Горбовський А. І., Войтович О. П. Дослідження безпеки у Інтернеті речей. Наукові праці ВНТУ. 2019. № 2. С. 34–40.

ДОДАТКИ

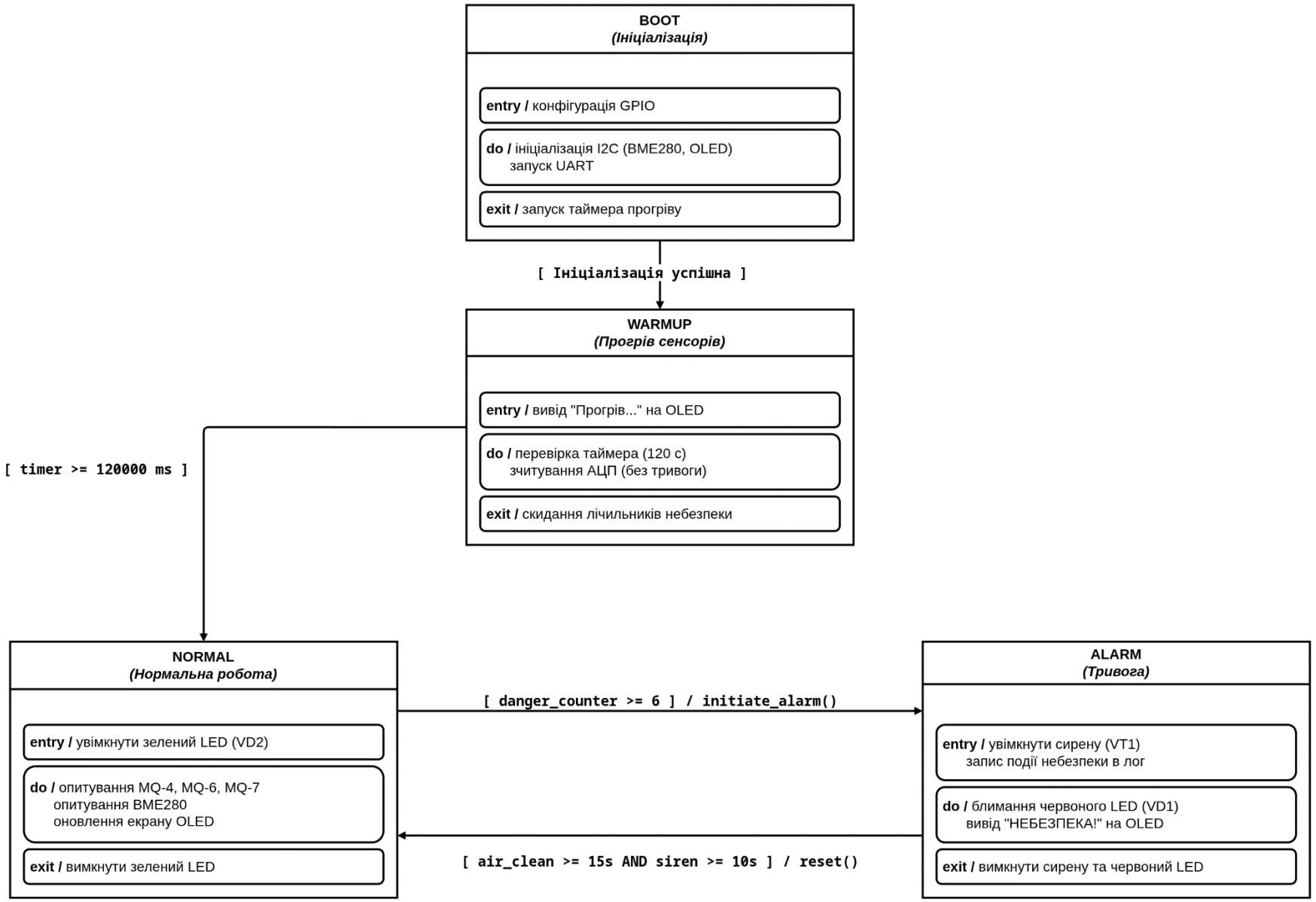
ДОДАТОК А



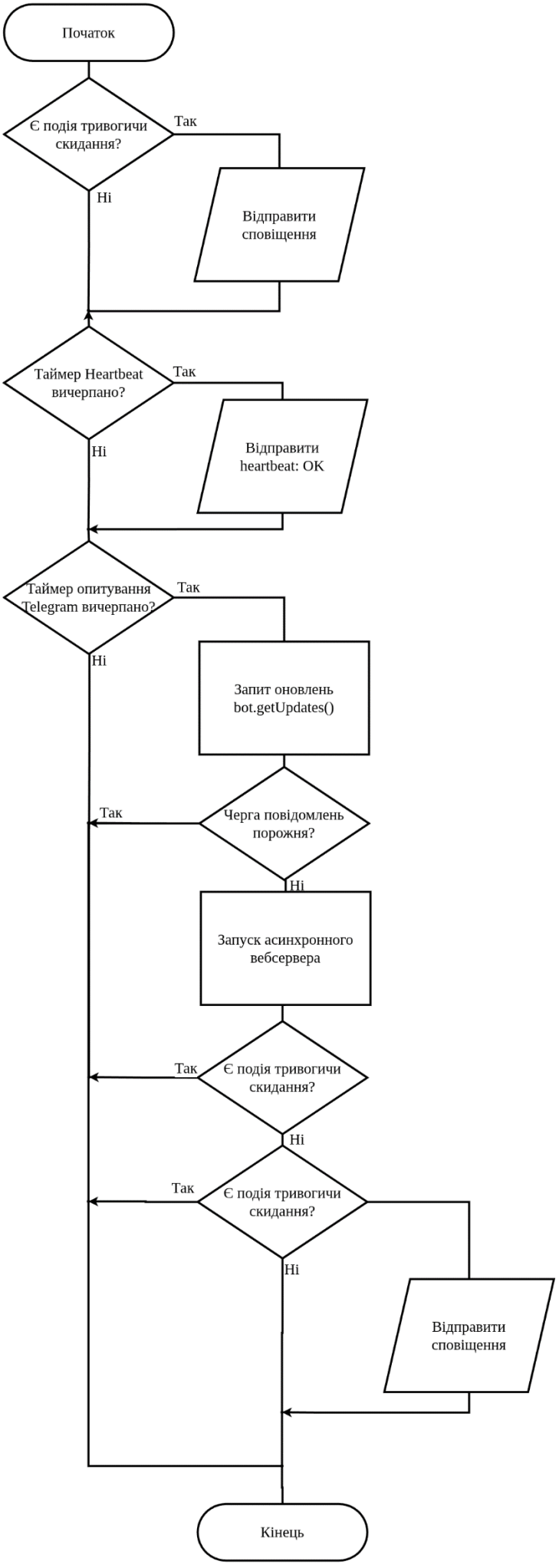
Зміц.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.005 С1		
Розробив		Дуряк А. І.			Система моніторингу шкідливих домішок в повітрі на основі ESP32. Схема електрична структурна	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Кулеш О. О.					
Н. контр.		Клепченко Я. М.				КПІ ім. І. Сікорського ФПСПМ, КВ-22	
Затв.		Клепченко Я. О.					



Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.006 Е3		
Розробив		Деркач А. І.			Система моніторингу шкідливих домішок в повітрі на основі ESP32. Схема електрична принципова	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Кочий О. О.					КПІ ім. І. Сікорського ФПСПМ, КВ-22
Н. контр.		Клятченко Я. М.					
Затв.		Романкевич В. О.					



Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.007 Д1		
Розробив		Дерюк А. І.			Система моніторингу шкідливих домішок в повітрі на основі ESP32. Структурна схема роботи основного циклу	Літ.	Аркуш 1	Аркушів 1
Перевірив		Кучай О. О.						
Н. контр.		Клепченко Я. М.				КПІ ім. І. Сікорського ФПСПМ, КВ-22		
Затв.		Романцов В. О.						



Зміна	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467200.008 Д2		
Розробив		Держич А. І.			Система моніторингу шкідливих домішок в повітрі на основі ESP32. Схема алгоритму роботи мережевої частини	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Кучмій О. О.				КПІ ім. І. Сікорського ФПСПМ, КВ-22	
Н. контр.		Клятченко Я. М.					
Затв.		Романенко В. О.					

ДОДАТОК Б

Додаток Б.1 – Файл конфігурації config.h

```
#pragma once
#include <stdint.h>
// ------------------------------
// Hardware pinout (ESP32)
// ------------------------------
// NOTE: MQ sensors are analog. Prefer ADC1 pins (GPIO32-39) to avoid Wi-Fi ADC2 conflicts.
namespace Pins {
  // MQ sensors (analog)
  static constexpr uint8_t MQ4_CH4_ADC = 34;   // ADC1_CH6
  static constexpr uint8_t MQ6_LPG_ADC = 35;   // ADC1_CH7
  static constexpr uint8_t MQ7_CO_ADC  = 32;   // ADC1_CH4
  // I2C (OLED + BME280)
  static constexpr uint8_t I2C_SDA = 21;
  static constexpr uint8_t I2C_SCL = 22;
  // Outputs (drive MOSFET modules)
  static constexpr uint8_t OUT_SIREN = 25;
  static constexpr uint8_t OUT_LED_ALARM = 26;
  static constexpr uint8_t OUT_LED_STRIP = 14; // optional
  // Button (TEST/RESET)
  static constexpr uint8_t BTN_TEST_RESET = 27;
}
// ------------------------------
// System behavior
// ------------------------------
namespace Sys {
  // Main sampling intervals
  static constexpr uint32_t SENSOR_SAMPLE_MS = 250;
  static constexpr uint32_t DISPLAY_REFRESH_MS = 500;
  static constexpr uint32_t TELEGRAM_POLL_MS = 2000; // for commands (optional)
  // Warmup time for MQ sensors before using baselines (heaters need time).
  static constexpr uint32_t MQ_WARMUP_MS = 2UL * 60UL * 1000UL; // 2 minutes (tune to your sensors)
  // Alarm latching & clear logic
  static constexpr uint32_t ALARM_MIN_HOLD_MS = 10UL * 1000UL; // siren minimum duration after trigger
  static constexpr uint32_t CLEAR_STABLE_MS = 15UL * 1000UL;   // readings must be safe this long to auto-clear (if enabled)
  // Telegram notifications
  static constexpr uint32_t HEARTBEAT_MS = 6UL * 60UL * 60UL * 1000UL; // 6 hours
}
// ------------------------------
// Thresholds
// ------------------------------
// MQ sensors are notoriously non-linear and require calibration.
// This firmware supports "relative" thresholds based on a learned baseline in clean air.
// We compute: normalized = filtered_adc / baseline_adc.
// Alarm when normalized exceeds threshold for N consecutive samples.
namespace Thresholds {
  // Gas alarms (relative to baseline). Example: 1.6 means +60% above baseline.
  static constexpr float MQ4_CH4_REL = 1.60f;
  static constexpr float MQ6_LPG_REL = 1.60f;
  static constexpr float MQ7_CO_REL  = 1.60f;
  // Debounce / confirm samples
  static constexpr uint16_t TRIGGER_CONFIRM_SAMPLES = 6; // 6 * 250ms = 1.5s
}
```

Додаток Б.2 – Основний файл логіки main.cpp

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <Preferences.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_BME280.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
#include "config.h"
```

```
#if __has_include("secrets.h")
  #include "secrets.h"
#else
  #error "Missing include/secrets.h. Copy include/secrets.example.h to include/secrets.h and fill credentials."
#endif
// ------------------------------
// Display
// ------------------------------
static constexpr int OLED_W = 128;
static constexpr int OLED_H = 64;
static Adafruit_SSD1306 display(OLED_W, OLED_H, &Wire, -1);
static bool display_ok = false;
// Optional BME280
static Adafruit_BME280 bme;
static bool bme_ok = false;
// ------------------------------
// Telegram
// ------------------------------
static WiFiClientSecure tlsClient;
static UniversalTelegramBot bot(TELEGRAM_BOT_TOKEN, tlsClient);
// ------------------------------
// Persistence
// ------------------------------
static Preferences prefs;
static constexpr const char* NVS_NS = "gasfire";
// ------------------------------
// Helpers
// ------------------------------
static uint32_t nowMs() { return millis(); }
struct Ewma {
  float alpha;
  bool has = false;
  float v = 0.0f;
  explicit Ewma(float a = 0.15f) : alpha(a) {}
  float update(float x) {
    if (!has) { v = x; has = true; }
    else { v = alpha * x + (1.0f - alpha) * v; }
    return v;
  }
};
struct Baseline {
  bool valid = false;
  float adc = 0.0f;
  uint32_t learned_at_ms = 0;
};
static void gpioWrite(uint8_t pin, bool on) {
  digitalWrite(pin, on ? HIGH : LOW);
}
static uint16_t readAdc12(uint8_t pin) {
  // ESP32 ADC range is 0..4095 at 12-bit.
  return static_cast<uint16_t>(analogRead(pin));
}
static float safeDiv(float a, float b, float fallback) {
  if (!isfinite(a) || !isfinite(b) || fabsf(b) < 1e-6f) return fallback;
  return a / b;
}
// ------------------------------
// Sensor channels
// ------------------------------
enum class Channel : uint8_t { MQ4 = 0, MQ6 = 1, MQ7 = 2 };
static const char* channelName(Channel c) {
  switch (c) {
    case Channel::MQ4: return "CH4";
    case Channel::MQ6: return "LPG";
    case Channel::MQ7: return "CO";
  }
  return "?";
}
struct ChannelState {
  Channel ch;
  uint8_t pin_adc;
  Ewma filter{0.20f};
  Baseline base;
  float rel = 1.0f;       // filtered_adc / baseline_adc
  float threshold_rel = 1.0f;
  uint16_t confirm = 0;   // consecutive trigger samples
  bool triggered = false;
};
static ChannelState channels[] = {
```

```
  { Channel::MQ4,   Pins::MQ4_CH4_ADC, Ewma(0.15f), {}, 1.0f, Thresholds::MQ4_CH4_REL, 0, false },
  { Channel::MQ6,   Pins::MQ6_LPG_ADC, Ewma(0.15f), {}, 1.0f, Thresholds::MQ6_LPG_REL, 0, false },
  { Channel::MQ7,   Pins::MQ7_CO_ADC,  Ewma(0.15f), {}, 1.0f, Thresholds::MQ7_CO_REL,  0, false },
};
static const char* nvsKeyBaseline(Channel c) {
  switch (c) {
    case Channel::MQ4: return "b_mq4";
    case Channel::MQ6: return "b_mq6";
    case Channel::MQ7: return "b_mq7";
  }
  return "b_?";
}
static void baselineLoad() {
  for (auto &cs : channels) {
    float v = prefs.getFloat(nvsKeyBaseline(cs.ch), NAN);
    if (isfinite(v) && v > 1.0f) {
      cs.base.valid = true;
      cs.base.adc = v;
      cs.base.learned_at_ms = 0;
    } else {
      cs.base.valid = false;
      cs.base.adc = 0.0f;
      cs.base.learned_at_ms = 0;
    }
  }
}
static void baselineSave(Channel c, float adc) {
  prefs.putFloat(nvsKeyBaseline(c), adc);
}
static void baselineCalibrateAll() {
  for (auto &cs : channels) {
    const float f = cs.filter.has ? cs.filter.v : static_cast<float>(readAdc12(cs.pin_adc));
    // Avoid learning nonsense baseline
    const float learned = constrain(f, 10.0f, 4095.0f);
    cs.base.valid = true;
    cs.base.adc = learned;
    cs.base.learned_at_ms = nowMs();
    baselineSave(cs.ch, learned);
  }
}
// ------------------------------
// Alarm state machine
// ------------------------------
enum class AlarmState : uint8_t { BOOT = 0, WARMUP = 1, NORMAL = 2, ALARM = 3 };
static AlarmState alarmState = AlarmState::BOOT;
static uint32_t alarmSinceMs = 0;
static uint32_t alarmHoldUntilMs = 0;
static uint32_t safeSinceMs = 0;
static bool dangerPrev = false;
static bool sirenOn = false;
static bool ledOn = false;
static bool stripOn = false;
static void setAlarmOutputs(bool on) {
  sirenOn = on;
  ledOn = on;
  stripOn = on;
  gpioWrite(Pins::OUT_SIREN, sirenOn);
  gpioWrite(Pins::OUT_LED_ALARM, ledOn);
  gpioWrite(Pins::OUT_LED_STRIP, stripOn);
}
static bool anyTriggered() {
  for (const auto &cs : channels) if (cs.triggered) return true;
  return false;
}
static String triggeredSummary() {
  String s;
  for (const auto &cs : channels) {
    if (!cs.triggered) continue;
    if (s.length()) s += ", ";
    s += channelName(cs.ch);
    s += " x";
    s += String(cs.rel, 2);
  }
  if (!s.length()) s = "unknown";
  return s;
}
// ------------------------------
// Wi-Fi / Telegram
// ------------------------------
```

```
static uint32_t lastWifiAttemptMs = 0;
static uint32_t wifiBackoffMs = 1000;
static bool wifiEnsureConnected() {
  if (WiFi.status() == WL_CONNECTED) return true;
  const uint32_t t = nowMs();
  if (t - lastWifiAttemptMs < wifiBackoffMs) return false;
  lastWifiAttemptMs = t;
  WiFi.mode(WIFI_STA);
  WiFi.setSleep(false);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  wifiBackoffMs = min<uint32_t>(wifiBackoffMs * 2, 60UL * 1000UL);
  return false;
}
static void wifiOnConnected() {
  wifiBackoffMs = 1000;
}
static bool telegramSend(const String& text) {
  if (WiFi.status() != WL_CONNECTED) return false;
  // UniversalTelegramBot requires TLS; for simplicity we use insecure (no CA cert pinning).
  // Production hardening: set CA cert or certificate fingerprint.
  tlsClient.setInsecure();
  return bot.sendMessage(TELEGRAM_CHAT_ID, text, "");
}
static uint32_t lastHeartbeatMs = 0;
static bool sentBootMsg = false;
static uint32_t lastTelegramPollMs = 0;
static void telegramMaybeHeartbeat() {
  const uint32_t t = nowMs();
  if (!sentBootMsg) {
    if (telegramSend(String("[") + DEVICE_NAME + "] booted")) sentBootMsg = true;
    lastHeartbeatMs = t;
    return;
  }
  if (t - lastHeartbeatMs >= Sys::HEARTBEAT_MS) {
    if (telegramSend(String("[") + DEVICE_NAME + "] heartbeat: OK, WiFi=" + WiFi.localIP().toString())) {
      lastHeartbeatMs = t;
    }
  }
}
static void telegramHandleCommand(const String& cmd) {
  if (cmd == "/status") {
    String msg = String("[") + DEVICE_NAME + "] status\n";
    msg += "state=";
    switch (alarmState) {
      case AlarmState::BOOT: msg += "BOOT"; break;
      case AlarmState::WARMUP: msg += "WARMUP"; break;
      case AlarmState::NORMAL: msg += "NORMAL"; break;
      case AlarmState::ALARM: msg += "ALARM"; break;
    }
    msg += "\n";
    for (const auto &cs : channels) {
      msg += String(channelName(cs.ch)) + ": rel=" + String(cs.rel, 2) + " thr=" + String(cs.threshold_rel, 2);
      msg += cs.base.valid ? " base=Y\n" : " base=N\n";
    }
    telegramSend(msg);
  } else if (cmd == "/calibrate") {
    baselineCalibrateAll();
    telegramSend(String("[") + DEVICE_NAME + "] calibrated baselines (clean air assumed)");
  } else if (cmd == "/silence") {
    // Stop outputs but keep state ALARM; it will re-arm if still dangerous.
    setAlarmOutputs(false);
    telegramSend(String("[") + DEVICE_NAME + "] siren silenced");
  } else if (cmd == "/help") {
    telegramSend(String("Commands:\n")
      + "/status\n"
      + "/calibrate (clean air)\n"
      + "/silence\n"
      + "/help\n");
  }
}
static void telegramPollCommands() {
  if (WiFi.status() != WL_CONNECTED) return;
  const uint32_t t = nowMs();
  if (t - lastTelegramPollMs < Sys::TELEGRAM_POLL_MS) return;
  lastTelegramPollMs = t;
  tlsClient.setInsecure();
  const int numNew = bot.getUpdates(bot.last_message_received + 1);
  for (int i = 0; i < numNew; i++) {
    const String chat_id = bot.messages[i].chat_id;
```

```
    if (chat_id != TELEGRAM_CHAT_ID) continue; // ignore others
    telegramHandleCommand(bot.messages[i].text);
  }
}
// ------------------------------
// Button handling
// ------------------------------
static bool btnPressedPrev = false;
static uint32_t btnDownAtMs = 0;
static void buttonUpdate() {
  const bool pressed = digitalRead(Pins::BTN_TEST_RESET) == LOW; // pull-up
  const uint32_t t = nowMs();
  if (pressed && !btnPressedPrev) {
    btnDownAtMs = t;
  } else if (!pressed && btnPressedPrev) {
    const uint32_t dur = t - btnDownAtMs;
    if (dur >= 1500) {
      baselineCalibrateAll();
      telegramSend(String("[") + DEVICE_NAME + "] calibrated baselines (button long-press)");
    } else if (dur >= 50) {
      // short press: acknowledge / try clear alarm
      if (alarmState == AlarmState::ALARM) {
        setAlarmOutputs(false);
        telegramSend(String("[") + DEVICE_NAME + "] alarm acknowledged (button)");
      } else {
        telegramSend(String("[") + DEVICE_NAME + "] test: OK");
      }
    }
  }
  btnPressedPrev = pressed;
}
// ------------------------------
// Core sampling
// ------------------------------
static uint32_t bootAtMs = 0;
static uint32_t lastSampleMs = 0;
static uint32_t lastDisplayMs = 0;
static void sampleSensors() {
  // Read & filter ADC channels
  for (auto &cs : channels) {
    const uint16_t raw = readAdc12(cs.pin_adc);
    const float f = cs.filter.update(static_cast<float>(raw));
    if (cs.base.valid) cs.rel = safeDiv(f, cs.base.adc, 1.0f);
    else cs.rel = 1.0f;
    const bool over = cs.base.valid && (cs.rel >= cs.threshold_rel);
    if (over) {
      cs.confirm = min<uint16_t>(cs.confirm + 1, 65000);
    } else {
      cs.confirm = 0;
    }
    cs.triggered = over && (cs.confirm >= Thresholds::TRIGGER_CONFIRM_SAMPLES);
  }
}
static void updateAlarmState() {
  const uint32_t t = nowMs();
  const bool danger = anyTriggered();
  if (!danger && dangerPrev) {
    // danger -> safe transition, start stability timer
    safeSinceMs = t;
  }
  dangerPrev = danger;
  if (alarmState == AlarmState::BOOT) {
    alarmState = AlarmState::WARMUP;
    safeSinceMs = t;
    return;
  }
  if (alarmState == AlarmState::WARMUP) {
    setAlarmOutputs(false);
    if (t - bootAtMs >= Sys::MQ_WARMUP_MS) {
      alarmState = AlarmState::NORMAL;
      safeSinceMs = t;
      if (!prefs.isKey(nvsKeyBaseline(Channel::MQ4)) ||
          !prefs.isKey(nvsKeyBaseline(Channel::MQ6)) ||
          !prefs.isKey(nvsKeyBaseline(Channel::MQ7))) {
        // First run: learn baseline automatically after warmup
        baselineCalibrateAll();
      }
      telegramSend(String("[") + DEVICE_NAME + "] warmup done, monitoring started");
    }
```

```
    return;
  }
  if (alarmState == AlarmState::NORMAL) {
    setAlarmOutputs(false);
    if (danger) {
      alarmState = AlarmState::ALARM;
      alarmSinceMs = t;
      alarmHoldUntilMs = t + Sys::ALARM_MIN_HOLD_MS;
      setAlarmOutputs(true);
      telegramSend(String("[") + DEVICE_NAME + "] ALARM: " + triggeredSummary());
    }
    return;
  }
  if (alarmState == AlarmState::ALARM) {
    if (danger) {
      // Keep outputs on while danger exists (unless user silenced, then keep off).
      if (t >= alarmHoldUntilMs && !sirenOn) {
        // user silenced; keep state ALARM but do not force siren back on unless re-triggered after safe period
      } else {
        setAlarmOutputs(true);
      }
      return;
    }
    // No danger. Hold outputs for minimum hold time.
    if (t < alarmHoldUntilMs) {
      setAlarmOutputs(true);
      return;
    }
    // Require stable safe readings before clearing.
    if (t - safeSinceMs >= Sys::CLEAR_STABLE_MS) {
      alarmState = AlarmState::NORMAL;
      setAlarmOutputs(false);
      telegramSend(String("[") + DEVICE_NAME + "] alarm cleared");
    }
  }
}
// ----------------------------
// Display UI
// ----------------------------
static void displayDraw() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.print(DEVICE_NAME);
  display.print(" ");
  switch (alarmState) {
    case AlarmState::BOOT: display.print("BOOT"); break;
    case AlarmState::WARMUP: display.print("WARMUP"); break;
    case AlarmState::NORMAL: display.print("OK"); break;
    case AlarmState::ALARM: display.print("ALARM"); break;
  }
  display.setCursor(0, 16);
  for (const auto &cs : channels) {
    display.print(channelName(cs.ch));
    display.print(":");
    display.print(cs.rel, 2);
    display.print(cs.triggered ? "!" : " ");
    display.print(" ");
  }
  display.setCursor(0, 32);
  display.print("WiFi:");
  display.print(WiFi.status() == WL_CONNECTED ? "OK " : "-- ");
  if (WiFi.status() == WL_CONNECTED) display.print(WiFi.localIP());
  if (bme_ok) {
    display.setCursor(0, 48);
    display.print("T:");
    display.print(bme.readTemperature(), 1);
    display.print("C H:");
    display.print(bme.readHumidity(), 0);
    display.print("%");
  }
  display.display();
}
// ----------------------------
// Arduino lifecycle
// ----------------------------
void setup() {
  bootAtMs = nowMs();
```

```
  Serial.begin(115200);
  delay(50);
  pinMode(Pins::OUT_SIREN, OUTPUT);
  pinMode(Pins::OUT_LED_ALARM, OUTPUT);
  pinMode(Pins::OUT_LED_STRIP, OUTPUT);
  setAlarmOutputs(false);
  pinMode(Pins::BTN_TEST_RESET, INPUT_PULLUP);
  Wire.begin(Pins::I2C_SDA, Pins::I2C_SCL);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    // No display; continue headless.
    display_ok = false;
  } else {
    display_ok = true;
    display.setRotation(2);
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0, 0);
    display.print(DEVICE_NAME);
    display.setCursor(0, 16);
    display.print("boot...");
    display.display();
  }
  bme_ok = bme.begin(0x76, &Wire);
  // ADC setup
  analogReadResolution(12);
  analogSetAttenuation(ADC_11db); // wider range
  analogSetPinAttenuation(Pins::MQ4_CH4_ADC, ADC_11db);
  analogSetPinAttenuation(Pins::MQ6_LPG_ADC, ADC_11db);
  analogSetPinAttenuation(Pins::MQ7_CO_ADC, ADC_11db);
  prefs.begin(NVS_NS, false);
  baselineLoad();
  WiFi.persistent(false);
  WiFi.setAutoReconnect(true);
  wifiEnsureConnected();
  alarmState = AlarmState::BOOT;
}
void loop() {
  wifiEnsureConnected();
  if (WiFi.status() == WL_CONNECTED) wifiOnConnected();
  const uint32_t t = nowMs();
  if (t - lastSampleMs >= Sys::SENSOR_SAMPLE_MS) {
    lastSampleMs = t;
    sampleSensors();
    updateAlarmState();
  }
  if (t - lastDisplayMs >= Sys::DISPLAY_REFRESH_MS) {
    lastDisplayMs = t;
    if (display_ok) displayDraw();
  }
  telegramMaybeHeartbeat();
  telegramPollCommands();
  buttonUpdate();
  delay(5);
}
```