

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему

**“Програмне забезпечення системи керування моніторинговою
інформаційною панеллю апаратно-програмного комплексу з
журналюванням змін робочих параметрів”**

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Горбачов Є.І.
«___» _____ 2026 р.

Керівник проекту
д.т.н., проф.
_____ Коваленко О.В.
«___» _____ 2026 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Освітній ступінь бакалавр
Галузь знань ._12 _____ “Інформаційні технології”
Спеціальність 123 “Комп’ютерна інженерія”
Освітньо-професійна (освітньо-наукова) програма “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 15 » січня 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Горбачова Євгенія Ігоровича

(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи керування моніторинговою інформаційною панеллю апаратно-програмного комплексу з журналюванням змін робочих параметрів.*

2. Керівник роботи *Коваленко Олександр Володимирович, д.т.н., проф.*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 136-02 від 27.02.2026 року

3. Строк подання студентом роботи до захисту 23.05.2026 р.

4. Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи керування моніторинговою інформаційною панеллю апаратно-програмного комплексу з журналюванням змін робочих параметрів.*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2. Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи. Реалізація роботи.

5. Підготовка та впровадження системи в промислову експлуатацію.

6. Основні висновки.

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи *1 аркуш*

Функціональна схема системи *1 аркуш*

Діаграма процесів *1 аркуш*

Блок-схема алгоритму роботи додатку *2 аркуша*

7. Дата видачі завдання « 27 » лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	23.05.2026 р.	

Дата видачі завдання
« 27 » лютого 2026 р.

Підпис керівника

Коваленко О.В.

(прізвище та ініціали)

Завдання прийнято до виконання
« 27 » лютого 2026 р.

Підпис здобувача

Горбачов Є.І.

(прізвище та ініціали)

АНОТАЦІЯ

Горбачов Є.І. Програмне забезпечення системи керування моніторинговою інформаційною панеллю апаратно-програмного комплексу з журналюванням змін робочих параметрів. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

У роботі розглянуто локальний моніторинговий комплекс на базі ESP32 і серверної вебпанелі. Об'єкт — збір, відображення та накопичення даних про стан IoT-пристрою з LED-індикацією в LAN. Предмет — програмне забезпечення панелі, HTTP-взаємодія з ESP32 і журналювання змін параметрів з розмежуванням `auto` та `user`.

Мета — розробити таке ПЗ, яке забезпечує моніторинг стану, керування LED-ефектами та збереження історії змін у SQLite; виконано огляд аналогів, обґрунтування засобів, опис архітектури, схем, алгоритмів, впровадження та перевірки.

Методи: аналіз джерел і порівняльний огляд IoT-рішень, ООП, прошивка з HTTP-сервісом, клієнт-серверна взаємодія, натурні випробування на стенді.

Результати: макет ESP32 + WS2812B, прошивка (Captive Portal, моніторинг, перевірка Internet), панель ASP.NET Core / Blazor Server з фоновим опитуванням і REST API, журнал у SQLite за порівнянням знімків стану; наведено схеми та блок-схеми. Новизна — поєднання локальної панелі, наочної індикації та журналу без обов'язкових хмарних платформ.

Практичне значення: навчальні та лабораторні стенди, демонстрація ланцюга «мікроконтролер — мережа — веб — БД»; робота ПЗ — під Windows з .NET SDK.

Ключові слова: моніторингова панель, ESP32, IoT, HTTP, Blazor Server, ASP.NET Core, SQLite, журналювання змін, LED-індикація.

ABSTRACT

Horbachov Y.I. Software of the monitoring information panel management system of the hardware and software complex with logging of changes in operating parameters. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

The thesis covers a local ESP32-based stack with a server-side web panel. Object: LAN telemetry acquisition, display, and persistence for an IoT device with LED status output. Subject: panel software, HTTP access to the device, and change logging with `auto` vs. `user` sources.

Goal: software for monitoring, LED control, and SQLite-backed change history; work done includes literature and comparative review, tool selection, architecture and algorithms, diagrams, deployment, and validation.

Methods: document analysis, IoT platform comparison, OOP, embedded HTTP firmware, client-server HTTP, bench tests.

Results: ESP32 + WS2812B mock-up, firmware (Captive Portal, monitoring, Internet check), ASP.NET Core / Blazor Server app with polling and REST API, snapshot-based logging; diagrams and flowcharts included. Novelty: integrated local panel, LED feedback, and change log without mandatory cloud IoT services.

Practical value: teaching and lab demos of the embedded-to-web-to-database path; runs on Windows with the .NET SDK.

Keywords: monitoring dashboard, ESP32, IoT, HTTP, Blazor Server,

A
S
P
.
N
E
T

C
o
r

К6П3_2026

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	3
ВСТУП.....	6
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	11
1.1 Призначення системи	11
1.2 Область застосування	15
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	17
2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.....	17
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування	24
2.3 Розгорнута постановка завдання	32
3 ОПИС І ОБґРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	35
3.1 Опис функціонування системи.....	35
3.2 Розробка структурної схеми.....	40
3.3 Розробка функціональної схеми	42
3.4 Розробка діаграми процесів	45
3.5 Розробка принципової схеми пристрою	46
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ	50
4.1 Алгоритмічна реалізація роботи системи та побудова блок-схем.....	49
4.2 Надійність, обробка помилок та захист програмного забезпечення	56
5 ПІДГОТОВКА ТА ВПРОВАДЖЕННЯ СИСТЕМИ В ЕКСПЛУАТАЦІЮ	59
5.1 Передумови та підготовка робочого місця.....	60
5.2 Збирання, прошивка та діагностика ESP32	62

					ВКРБ-123.26.0038.00.00.ПЗ				
Вим	Арк.	№ докум.	Підп.	Дата	Програмне забезпечення системи керування моніторинговою інформаційною панеллю апаратно-програмного комплексу з журналюванням змін робочих параметрів	Лім.	Аркуш	Аркушів	
Розроб.	Горбачов Є. І.					Б		1	79
Перев.	Коваленко О. В.								
Н.контр.	Коваленко А. С.								
Затв.	Смірнов О. А.					ЦНТУ КІ-22-2			

5.3 Налаштування Wi-Fi, Captive Portal та інтеграція з панеллю.....	63
5.4 Запуск панелі, перевірка інтерфейсу та сценаріїв моніторинг.....	64
5.5 Перевірка журналювання, відтворюваність результатів та типові збої.....	67
5.6 Авторське право, склад об'єкта права інтелектуальної власності та правова інформація у застосунку.....	68
6 ОСНОВНІ ВИСНОВКИ	73
СПИСОК ЛІТЕРАТУРИ.....	77

К6ПЗ_2026

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
		№ докум.	Підпис	Дата		2

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

У роботі використовуються такі умовні позначення, скорочення та терміни:

API	- Application Programming Interface, програмний інтерфейс прикладної взаємодії, за допомогою якого програмні компоненти обмінюються даними та командами локальна обчислювальна мережа центр обробки даних.
ASP.NET Core	- Платформа Microsoft для створення вебзастосунків, серверних служб та REST API.
Blazor Server	- Технологія ASP.NET Core для створення інтерактивних вебінтерфейсів із виконанням основної логіки на сервері.
Captive Portal	- Режим первинного налаштування мережі, за якого пристрій створює власну точку доступу та відкриває сторінку конфігурації Wi-Fi.
Entity Framework Core	- ORM-фреймворк для роботи з базами даних у застосунках .NET.
ESP32	- Мікроконтролер із підтримкою Wi-Fi та Bluetooth, який у роботі використовується як апаратна основа моніторингового комплексу.
FastLED	- Бібліотека для керування адресними світлодіодними стрічками у мікроконтролерів.
GPIO	- General Purpose Input/Output, виводи загального призначення мікроконтролера, які використовуються для підключення периферійних пристроїв.
HTTP	- HyperText Transfer Protocol, протокол прикладного рівня, який використовується для обміну даними між ESP32 та

ресурсу, що використовується для адресації вебсторінок і HTTP-маршрутів.

- Wi-Fi

- Технологія бездротового передавання даних, яка використовується для підключення ESP32 до локальної мережі.
- WS2812B

- Адресний RGB-світлодіод із вбудованим контролером; у роботі використовується у складі LED-стрічки для візуальної індикації стану.

ВСТУП

Загальна характеристика теми. В умовах стрімкого розвитку IoT-технологій, вбудованих систем та локальних засобів автоматизації зростає потреба у створенні апаратно-програмних комплексів, які здатні не лише виконувати окремі технічні функції, але й надавати користувачу зрозумілі засоби спостереження, керування та аналізу. Особливого значення набувають системи, що поєднують мікроконтролерні пристрої, мережеву взаємодію, вебінтерфейс та механізми збереження історії подій. Такі рішення дозволяють забезпечити прозору роботу пристрою, швидку діагностику його стану та зручне керування параметрами без безпосереднього втручання у прошивку.

Тема випускної кваліфікаційної роботи пов'язана з розробкою програмного забезпечення системи керування моніторинговою інформаційною панеллю апаратно-програмного комплексу з журналюванням змін робочих параметрів. У межах роботи розглядається комплекс на базі мікроконтролера ESP32, адресної LED-стрічки WS2812B та вебзастосунку, що виконує функції інформаційної панелі. Апаратна частина забезпечує підключення до Wi-Fi, перевірку доступності Internet-з'єднання та світлову індикацію стану, а програмна частина виконує періодичне опитування пристрою, відображення параметрів, керування LED-ефектами та збереження змін до локальної бази даних.

Характеристика бази розробки. Практична реалізація роботи виконувалася на базі власного апаратно-програмного стенду, що складається з плати ESP32, адресної LED-стрічки WS2812B, джерела живлення, кнопки скидання та програмної інформаційної панелі. Такий стенд дозволяє відтворити типовий сценарій роботи IoT-пристрою у локальній мережі: первинне налаштування Wi-Fi, підключення до маршрутизатора, передавання стану через

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
						6
		№ докум.	Підпис	Дата		

НТТР, керування режимами індикації та накопичення журналу змін робочих параметрів. Використання фізичного макета є важливим, оскільки воно дає змогу перевірити не лише програмну логіку, але й взаємодію з реальними апаратними компонентами. На відміну від суто програмної симуляції, апаратно-програмний стенд враховує особливості роботи Wi-Fi-з'єднання, живлення LED-стрічки, реакцію мікроконтролера на перезавантаження та поведінку системи у випадку втрати доступу до Internet. Це створює більш наближені до практичних умов передумови для розробки та тестування програмного забезпечення.

Актуальність теми. Актуальність розробки зумовлена тим, що у невеликих IoT-комплексах часто відсутні зручні засоби моніторингу та журналювання. Пристрій може виконувати свої основні функції, але користувач не завжди має можливість швидко визначити причину зміни стану, момент втрати зв'язку або джерело виконаної команди. У таких умовах інформаційна панель із журналом змін стає важливим інструментом діагностики, супроводу та аналізу роботи комплексу.

Актуальність проєкту визначається такими аспектами:

- поєднання апаратної та програмної частин: система демонструє повний цикл взаємодії між ESP32, LED-стрічкою, вебзастосунком та базою даних;
- наочність моніторингу: LED-індикація дозволяє швидко оцінити стан Internet-з'єднання без відкриття браузера;
- зручність керування: вебпанель надає користувачу можливість змінювати LED-ефекти та повертати пристрій до режиму автоматичного моніторингу;
- журналювання змін: система фіксує час події, назву параметра, попереднє значення, нове значення та джерело зміни;

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						7
		№ докум.	Підпис	Дата		

- локальність рішення: комплекс може працювати у межах локальної мережі без обов'язкового використання хмарних сервісів або зовнішніх IoT-платформ.

Особливістю розроблюваної системи є те, що моніторинг розглядається не лише як відображення поточного стану, а як сукупність процесів отримання даних, порівняння значень, фіксації змін та подання результатів користувачу. Поточний стан пристрою є корисним у момент перегляду, однак для аналізу стабільності роботи важливо мати історію подій. Саме журнал змін дозволяє визначити, коли пристрій втрачав Internet-з'єднання, які ефекти обирав користувач, чи змінювалася IP-адреса та які параметри оновлювалися автоматично.

Мета й завдання дослідження. Метою роботи є розробка програмного забезпечення системи керування моніторинговою інформаційною панеллю апаратно-програмного комплексу на базі ESP32, що забезпечує відображення поточного стану пристрою, керування LED-ефектами та журналювання змін робочих параметрів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- визначити призначення та область застосування системи керування моніторинговою інформаційною панеллю;
- виконати огляд існуючих IoT-платформ, систем моніторингу, програмних панелей та засобів журналювання подій;
- обґрунтувати вибір мікроконтролера ESP32, адресної LED-стрічки WS2812B, джерела живлення та допоміжних апаратних засобів;
- обґрунтувати вибір мов програмування, середовища розробки, ASP.NET Core, Blazor Server, Entity Framework Core та SQLite;
- описати архітектуру взаємодії між ESP32, серверною частиною, інформаційною панеллю та базою даних;

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						8
		№ докум.	Підпис	Дата		

- розробити логіку отримання стану пристрою, порівняння попередніх і нових значень та визначення змін робочих параметрів;
- описати механізм керування LED-ефектами, повернення до режиму автоматичного моніторингу та скидання параметрів пристрою;
- описати механізм журналювання змін із розмежуванням автоматичних подій та дій користувача;
- розробити структурну, функціональну та принципову схеми апаратно-програмного комплексу.

Об'єктом розробки є апаратно-програмний комплекс моніторингу стану мережевого IoT-пристрою з візуальною індикацією. Предметом розробки є програмне забезпечення моніторингової інформаційної панелі, механізм взаємодії з ESP32 та засоби журналювання змін робочих параметрів.

Практична цінність отриманих результатів. Результати роботи мають практичне значення, оскільки можуть бути використані як основа для навчального, демонстраційного або прикладного IoT-комплексу. Розроблена система дозволяє спостерігати за станом ESP32 у локальній мережі, керувати LED-індикацією через вебінтерфейс та накопичувати історію змін параметрів у базі даних SQLite. Це дає змогу використовувати комплекс для діагностики мережевого стану, демонстрації принципів роботи вбудованих систем та вивчення взаємодії між апаратною і програмною частинами.

Практична цінність також полягає у можливості подальшого масштабування архітектури. Якщо змінити набір параметрів, які передає ESP32, система може бути адаптована для моніторингу температури, вологості, освітленості, напруги живлення або стану промислового обладнання. При цьому загальні принципи залишаються незмінними: пристрій формує поточний стан, серверна частина виконує опитування, інформаційна панель відображає дані, а журнал фіксує зміни. Крім того, виконання такої роботи дозволяє поєднати знання з різних напрямів комп'ютерної інженерії: програмування

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						9
		№ докум.	Підпис	Дата		

мікроконтролерів, мережевої взаємодії, веброзробки, баз даних, проектування інтерфейсів та аналізу подій. Це підтверджує доцільність теми для випускної кваліфікаційної роботи, оскільки розробка охоплює як апаратний, так і програмний рівні системи.

Таким чином, програмне забезпечення системи керування моніторинговою інформаційною панеллю апаратно-програмного комплексу з журналюванням змін робочих параметрів є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ_2026

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						10
		№ докум.	Підпис	Дата		

1. ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Розроблювана система призначена для керування моніторинговою інформаційною панеллю апаратно-програмного комплексу, побудованого на основі мікроконтролера ESP32 та адресної LED-стрічки WS2812B. Основне призначення системи полягає у забезпеченні зручного контролю поточного стану пристрою, керуванні режимами світлової індикації та збереженні історії змін робочих параметрів. Апаратна частина комплексу виконує функції підключення до бездротової мережі, перевірки доступності Internet-з'єднання та відображення стану за допомогою LED-стрічки. Програмна частина виконує роль інформаційної панелі, яка через HTTP-запити отримує дані від ESP32, відображає їх у зручному вигляді та записує зміни до бази даних.

Призначення системи полягає не лише у передаванні команд на пристрій, а й у формуванні єдиного середовища спостереження за його роботою. Користувач повинен мати можливість швидко оцінити стан комплексу без аналізу службових повідомлень у послідовному порту або без безпосереднього підключення до мікроконтролера. Для цього інформаційна панель відображає параметри у вигляді окремих карток стану та таблиці подій.

Важливою функцією системи є розмежування автоматичних змін і дій користувача. Автоматичні зміни виникають у результаті періодичного опитування ESP32, наприклад у разі втрати Internet-з'єднання або зміни IP-адреси. Користувацькі зміни виникають тоді, коли оператор самостійно обирає LED-ефект, задає колір або виконує скидання пристрою. Такий поділ підвищує інформативність журналу та спрощує подальший аналіз.

З позиції користувача система повинна бути простою та зрозумілою. Для роботи з нею не потрібно знати внутрішню будову прошивки або вручну

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						11
		№ докум.	Підпис	Дата		

надсилати HTTP-запити на пристрій. Уся необхідна інформація виводиться на інформаційну панель, а керування здійснюється за допомогою кнопок і елементів інтерфейсу. Це зменшує ймовірність помилкових дій та робить комплекс придатним для демонстраційного використання.

З позиції розробника важливим є те, що система має чіткий розподіл відповідальності між компонентами. ESP32 відповідає за апаратну взаємодію та первинне визначення стану, серверна частина відповідає за обробку даних і збереження журналу, а вебінтерфейс відповідає за представлення інформації користувачу. Такий підхід полегшує налагодження, оскільки кожен компонент можна перевіряти окремо.

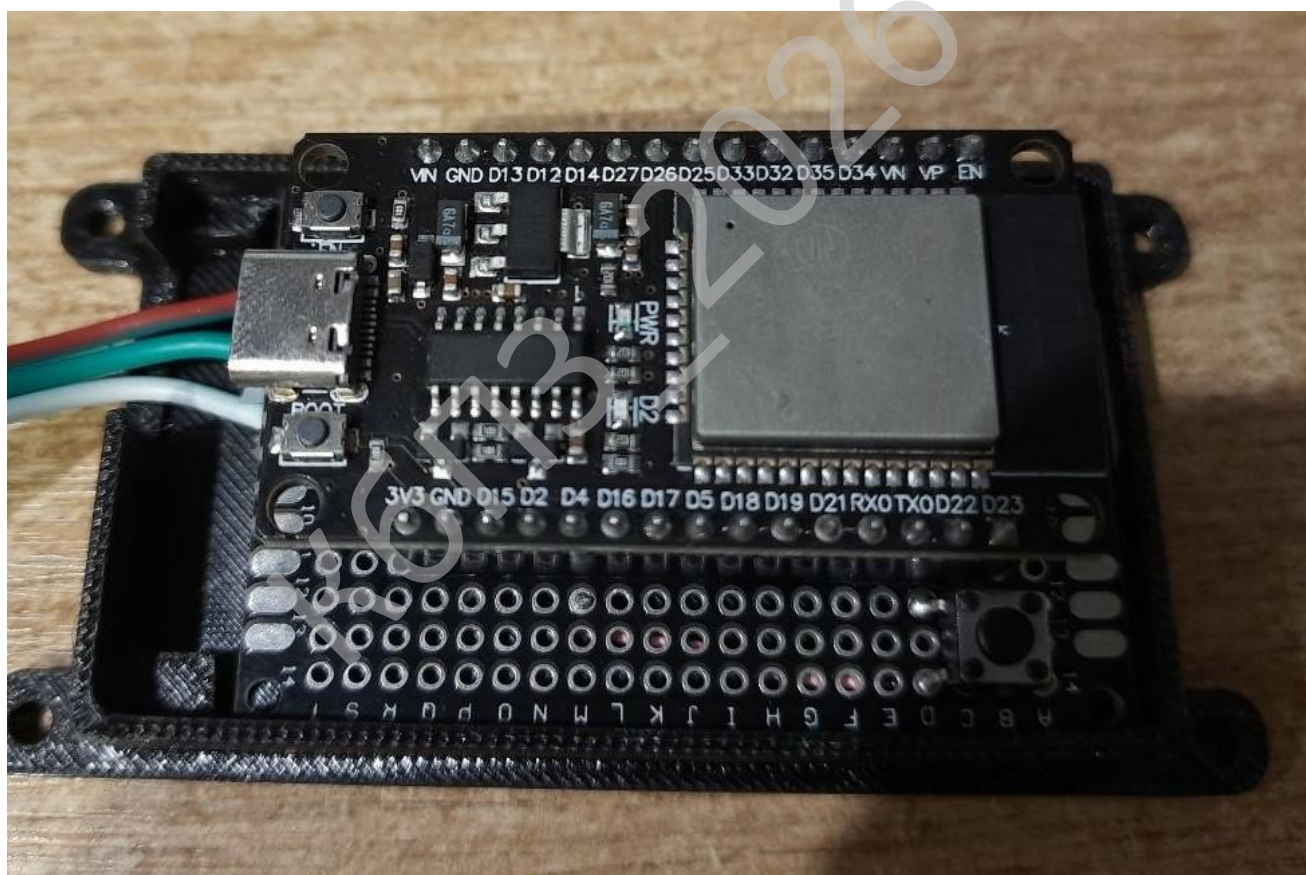


Рисунок 1.1 - Мікроконтролер ESP32, використаний у складі апаратно-програмного комплексу

ESP32 у розроблюваній системі працює у двох основних режимах.

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
		№ докум.	Підпис	Дата		12

Перший режим є режимом первинного налаштування. У цьому режимі пристрій створює точку доступу Wi-Fi з назвою 'ESP32-WiFi-Monitor', запускає Captive Portal і надає користувачу сторінку для вибору бездротової мережі та введення пароля. Після успішного підключення параметри мережі зберігаються в енергонезалежній пам'яті пристрою. Другий режим є режимом моніторингу. У цьому режимі ESP32 підключається до заданої Wi-Fi-мережі, періодично перевіряє доступність Internet та надає вебінтерфейс для отримання стану й керування LED-ефектами. Інформаційна панель, реалізована як вебзастосунок, звертається до пристрою через локальну мережу та отримує поточні значення параметрів.

Перехід між режимами має важливе практичне значення. Режим первинного налаштування дозволяє підготувати пристрій до роботи без попереднього програмування параметрів мережі в коді прошивки. Режим моніторингу використовується у штатній експлуатації, коли пристрій уже підключений до локальної мережі та може взаємодіяти з інформаційною панеллю. Це робить систему зручнішою для повторного використання в різних мережевих умовах.

Система повинна забезпечувати відображення таких параметрів:

- доступність пристрою в локальній мережі;
- стан підключення до Wi-Fi;
- назва підключеної мережі;
- IP-адреса пристрою;
- доступність Internet-з'єднання;
- активний режим LED-індикації;
- історія зміни робочих параметрів.

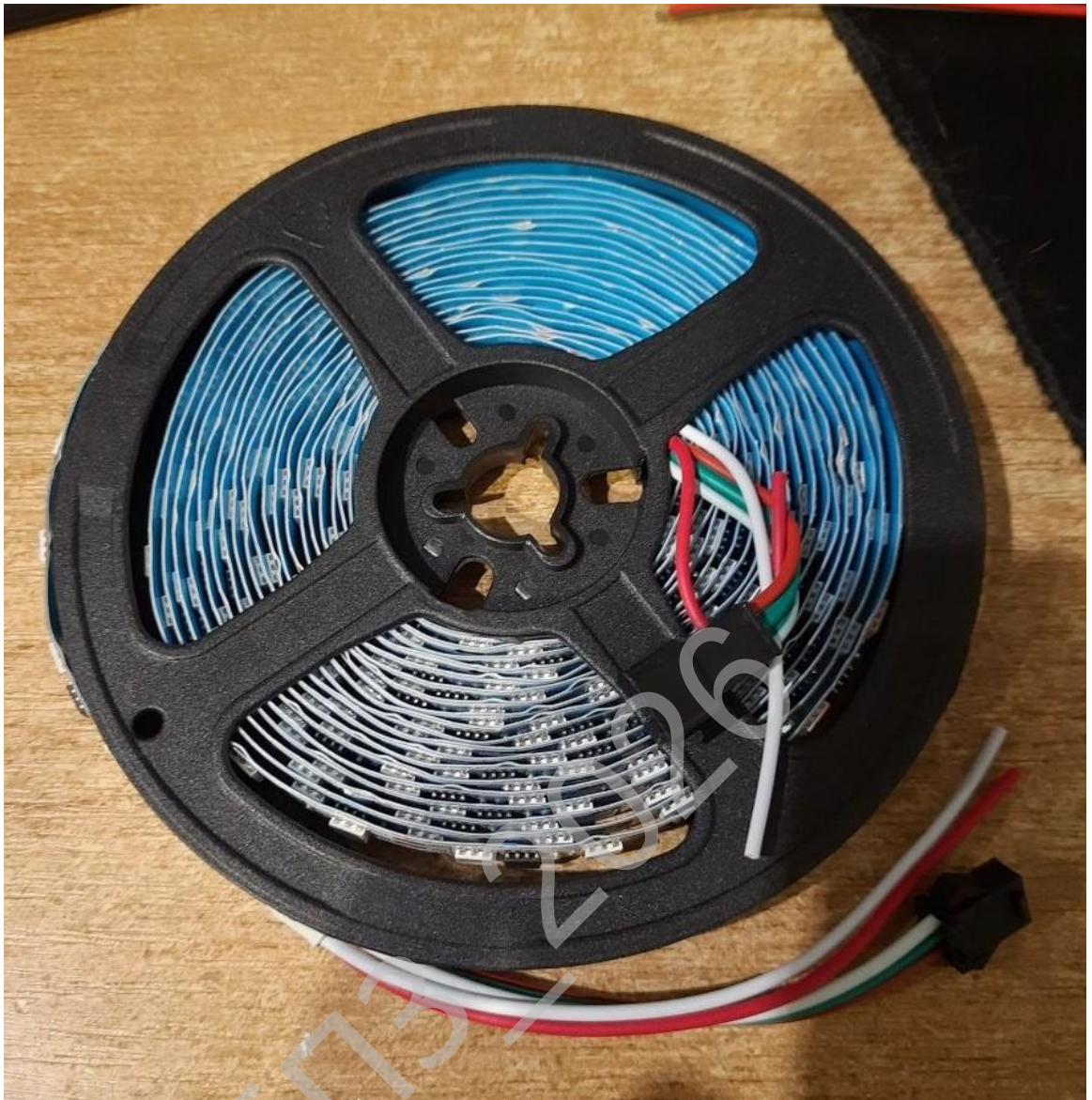


Рисунок 1.2 - Адресна LED-стрічка WS2812B як засіб візуальної індикації стану

LED-стрічка WS2812B використовується для наочного відображення стану. У разі доступності Internet-з'єднання застосовується зелений пульсуючий ефект. У разі втрати Internet-з'єднання використовується червоне миготіння. Додатково передбачено декоративні ефекти, зокрема 'rainbow', 'fill_rainbow', 'static', 'snake' та 'waiting', які можуть бути обрані користувачем через інформаційну панель. Використання світлової індикації є доцільним, оскільки вона дозволяє отримати інформацію про стан системи без відкриття браузера. У

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
		№ докум.	Підпис	Дата		14

реальних умовах це зручно для швидкої діагностики: зелений ефект сигналізує про нормальний стан, червоний - про проблему з доступністю Internet. Декоративні режими потрібні для перевірки працездатності LED-стрічки та демонстрації можливостей керування окремими ефектами.

Світлова індикація також підвищує наочність роботи комплексу. На відміну від звичайного текстового повідомлення або числового параметра, LED-стрічка дозволяє швидко помітити зміну стану навіть на відстані. Це особливо корисно у навчальних аудиторіях або лабораторіях, де пристрій може використовуватися як демонстраційний стенд для пояснення принципів моніторингу та керування.

1.2 Область застосування

Областю застосування розроблюваної системи є локальні IoT-комплекси, навчальні лабораторії, демонстраційні стенди та невеликі прикладні системи моніторингу. Комплекс може використовуватися для контролю доступності мережевого підключення, перевірки стабільності Wi-Fi, демонстрації принципів роботи вбудованих систем та вивчення механізмів взаємодії між мікроконтролером і вебзастосунком.

У навчальному процесі система може бути використана для демонстрації повного циклу розробки IoT-рішення. Такий цикл включає підключення апаратних компонентів, написання прошивки для ESP32, налаштування Wi-Fi, створення REST-подібного HTTP-інтерфейсу, розробку вебпанелі, роботу з базою даних та реалізацію журналювання подій.

У лабораторних умовах система може застосовуватися як індикатор стану мережевого підключення. Наприклад, у разі нестабільної роботи маршрутизатора або провайдера LED-стрічка швидко відображатиме зміну стану, а програмна панель зберігатиме час та характер зміни. Це дозволяє не лише побачити поточний стан, а й проаналізувати історію роботи пристрою.

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						15
		№ докум.	Підпис	Дата		

У прикладних умовах розроблену архітектуру можна адаптувати для інших типів моніторингу. Замість перевірки Internet-з'єднання ESP32 може отримувати дані з датчиків температури, вологості, освітленості, напруги або стану промислового обладнання. При цьому загальна логіка системи залишається незмінною: пристрій надає поточний стан, інформаційна панель його відображає, а зміни параметрів зберігаються в журналі.

Окремою сферою застосування є тестування і демонстрація принципів взаємодії між апаратним та програмним рівнями. У межах одного комплексу можна показати роботу GPIO, Wi-Fi, HTTP-запитів, JSON-структур, фонових сервісів, REST API, вебінтерфейсу та бази даних. Завдяки цьому система є корисною не тільки як готове рішення, але й як навчальний приклад побудови повного апаратно-програмного циклу.

Система також може застосовуватися для прототипування майбутніх рішень. Якщо виникає потреба створити пристрій з іншими датчиками або індикаторами, наявна архітектура дає змогу зберегти програмну частину майже без змін, модифікувавши лише набір параметрів, які передає ESP32. Така гнучкість є важливою перевагою при розробці малих IoT-комплексів.

Ще одним напрямом застосування є використання системи як основи для дослідження надійності мережевого з'єднання. Оскільки журнал зберігає моменти зміни стану, можна оцінювати частоту втрати доступу до Internet, тривалість проблемних періодів та поведінку пристрою після відновлення зв'язку. Це робить систему корисною не тільки для керування LED-індикацією, але й для спостереження за якістю мережевого середовища.

2. ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Системи моніторингу IoT-пристроїв можуть мати різну архітектуру залежно від масштабу задачі, кількості пристроїв, вимог до зберігання даних і способу взаємодії з користувачем. У загальному випадку такі системи складаються з апаратного пристрою, каналу передавання даних, програмного сервера, сховища даних та інформаційної панелі. Для обміну даними між пристроєм і сервером можуть використовуватися HTTP, MQTT, WebSocket, CoAP або інші протоколи. HTTP є одним із найпростіших варіантів для локальних систем, оскільки він добре підтримується як на ESP32, так і у вебзастосунках. MQTT доцільний для систем із великою кількістю пристроїв і брокером повідомлень, однак для одного локального пристрою він може бути надмірним.

Журналювання змін параметрів є важливою складовою систем моніторингу. Воно дає змогу зберігати не лише поточне значення, а й історію переходів між станами. Типовий запис журналу містить час події, назву параметра, старе значення, нове значення та джерело зміни. Саме такий підхід використовується у розроблюваній системі. При аналізі аналогічних систем доцільно враховувати не лише наявність інформаційної панелі, а й спосіб організації обміну даними, складність розгортання, вимоги до апаратних ресурсів і можливість локальної роботи. Для навчального або лабораторного комплексу важливими є простота налаштування, прозорість архітектури та можливість пояснити роботу кожного компонента. Надмірно складні платформи

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						17
		№ докум.	Підпис	Дата		

можуть мати більше функцій, але ускладнюють розуміння внутрішньої логіки системи.

Ще одним критерієм порівняння є можливість керування пристроєм, а не лише перегляду телеметрії. Багато систем моніторингу орієнтовані на збір і візуалізацію даних, але не завжди зручно підтримують зворотні команди до конкретного пристрою. У розроблюваному комплексі керування є обов'язковою функцією, оскільки користувач повинен мати можливість змінювати LED-ефекти та повертати пристрій до режиму автоматичного моніторингу.

При виборі архітектури також необхідно враховувати вимоги до прозорості роботи системи. У навчальному проєкті важливо, щоб було зрозуміло, звідки надходять дані, як вони обробляються, де зберігаються та яким чином відображаються. Тому перевага надається такому рішенню, у якому всі етапи проходження інформації можна описати послідовно: від мікроконтролера до вебпанелі й бази даних.

Окрему увагу слід приділяти питанню автономності. Хмарні платформи зручні для масштабних систем, але вони залежать від зовнішніх сервісів, облікових записів та доступу до Internet. У даній роботі розглядається локальний комплекс, який повинен залишатися працездатним у межах локальної мережі. Саме тому використання локального вебзастосунку та локальної бази даних є обґрунтованим.

Готові IoT-платформи та системи моніторингу

До поширених готових платформ моніторингу IoT-пристроїв належать ThingsBoard, Home Assistant, Node-RED, Grafana у поєднанні з базами даних часових рядів, а також хмарні сервіси на зразок AWS IoT Core, Azure IoT Hub та Google Cloud IoT-подібних сервісів. Такі рішення забезпечують широкі можливості інтеграції, підтримують велику кількість пристроїв, дозволяють створювати інформаційні панелі, будувати графіки, налаштовувати правила обробки подій і надсилати сповіщення.

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
						18
		№ докум.	Підпис	Дата		

ThingsBoard є платформою для збору, обробки та візуалізації даних IoT-пристроїв. Вона підтримує MQTT, HTTP та інші протоколи, має засоби побудови панелей і може використовуватися для промислового моніторингу. Однак її розгортання потребує окремого серверного середовища, налаштування бази даних, правил обробки телеметрії та прав доступу.

Home Assistant орієнтований переважно на автоматизацію розумного будинку. Він підтримує багато інтеграцій, зокрема пристрої ESP32 через ESPHome або MQTT. Перевагою є велика екосистема, але для невеликого спеціалізованого комплексу така платформа може бути надмірною, оскільки містить значно більше функцій, ніж потрібно для задачі моніторингу одного пристрою.

Node-RED є інструментом візуального програмування потоків даних. Він дозволяє швидко створювати логіку обробки повідомлень, підключати пристрої, API та бази даних. Проте використання Node-RED означає залежність від окремого середовища виконання та потребує додаткового налаштування вузлів, потоків і сховищ.

Grafana часто використовується для побудови інформаційних панелей на основі даних з Prometheus, InfluxDB або інших джерел. Це потужний інструмент візуалізації, але він не виконує самостійно роль сервера керування ESP32 і потребує окремого джерела даних. Для невеликої локальної системи з журналом змін використання Grafana може ускладнити архітектуру.

Таким чином, готові IoT-платформи мають значні функціональні можливості, але для розроблюваного комплексу вони є надлишковими. Основною задачею роботи є створення власного програмного забезпечення, яке точно відповідає вимогам теми: опитування ESP32, керування LED-ефектами, відображення стану та журналювання змін робочих параметрів.

Для даної роботи важливо, щоб система залишалася зрозумілою на рівні програмної реалізації. Використання власного ASP.NET Core застосунку дозволяє явно описати всі ключові елементи: HTTP-клієнт для ESP32,

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						19
		№ докум.	Підпис	Дата		

контролери REST API, фонову службу опитування, модель журналу та сторінки Blazor. У готових платформах значна частина цієї логіки прихована всередині фреймворку або налаштовується через графічний інтерфейс, що менш зручно для пояснення у випускній роботі.

Крім того, готові платформи часто передбачають універсальність, а не вузьку спеціалізацію. Вони підтримують різні типи пристроїв, протоколів і сценаріїв, але саме через це потребують додаткового налаштування. У розроблюваній системі функціональність обмежена конкретною задачею, тому архітектура є простішою: один пристрій ESP32, один вебзастосунок, одна локальна база даних і чітко визначений набір параметрів.

Під час порівняння аналогів слід враховувати, що надлишкова функціональність не завжди є перевагою. Якщо система має багато можливостей, але для конкретної задачі використовується лише мала їх частина, це збільшує складність супроводу. У розроблюваному проєкті важливо зберегти баланс між достатньою функціональністю та простотою реалізації. Саме тому власна реалізація інформаційної панелі є доцільною. Вона дозволяє уникнути непотрібних залежностей, зберегти контроль над структурою даних і точно описати механізм журналювання. Крім того, власна реалізація дає можливість продемонструвати практичні навички розробки серверної частини, вебінтерфейсу та роботи з базою даних.

Програмні панелі, симулятори та засоби локального моніторингу мережевих пристроїв

Для локального моніторингу мережевих пристроїв часто використовуються вебінтерфейси, REST API, консольні утиліти, журнали операційної системи або спеціалізовані програми. Наприклад, стан мережевого пристрою можна перевіряти за допомогою ICMP-запитів, HTTP-запитів або періодичного опитування сервісу. Однак такі засоби зазвичай не забезпечують одночасно керування пристроєм, наочну LED-індикацію та структуроване журналювання змін.

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
						20
		№ докум.	Підпис	Дата		

Програмні симулятори та емулятори IoT-пристроїв дозволяють перевіряти логіку системи без фізичного обладнання. У програмній частині розроблюваного комплексу передбачено режим симуляції, який може імітувати зміну параметрів без підключення реального ESP32. Це корисно під час розробки й тестування вебінтерфейсу, оскільки дозволяє перевірити роботу журналу, фільтрів і відображення стану.

Водночас симулятор не може повністю замінити фізичний пристрій, оскільки не враховує реальні особливості Wi-Fi-з'єднання, живлення LED-стрічки, затримки HTTP-запитів та поведінку ESP32 у разі втрати мережі. Тому для остаточної перевірки системи необхідне тестування з реальним апаратним комплексом.

Локальна вебпанель, створена спеціально для конкретного пристрою, має низку переваг. Вона може відображати саме ті параметри, які надає ESP32, підтримувати потрібні команди керування та зберігати журнал у зручному форматі. При цьому не потрібно розгортати складну сторонню платформу або підключати хмарні сервіси. У розроблюваній системі локальна панель виконує також роль проміжного рівня між користувачем і ESP32. Користувач не звертається до маршрутів мікроконтролера напряду, а працює через зручний інтерфейс. Це дозволяє приховати технічні деталі HTTP-запитів, форматів даних і обробки помилок. Якщо ESP32 тимчасово недоступний, панель може відобразити відповідний стан, не перериваючи роботу вебзастосунку.

Режим симуляції, передбачений у програмній частині, є корисним для перевірки інтерфейсу без фізичного пристрою. Він дозволяє створювати тестові зміни параметрів, перевіряти появу записів у журналі та відпрацьовувати логіку фільтрації. Проте остаточно перевірка повинна виконуватися з реальним ESP32, оскільки саме фізичний пристрій демонструє роботу Wi-Fi, LED-стрічки, кнопки скидання та реального мережевого обміну.

Наявність симуляційного режиму також корисна з погляду розробки та демонстрації. Якщо апаратна частина тимчасово недоступна або перебуває в

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						21
		№ докум.	Підпис	Дата		

процесі налагодження, програмна частина все одно може бути перевірена. Це дозволяє розділити процес тестування на два етапи: спочатку перевірити логіку інтерфейсу та журналювання, а потім перевірити взаємодію з реальним пристроєм.

Такий підхід зменшує ризики під час розробки, оскільки помилки в інтерфейсі або базі даних можна виявити ще до підключення ESP32. У результаті процес налагодження стає більш послідовним, а система легше доводиться до стабільного стану.

Порівняльна характеристика та переваги розроблюваної системи

У порівнянні з готовими IoT-платформами розроблювана система має менший масштаб, але краще відповідає поставленій задачі. Вона не потребує брокера повідомлень, хмарної інфраструктури або складного середовища розгортання. Для роботи достатньо ESP32, локальної мережі, вебзастосунку ASP.NET Core та бази даних SQLite.

Основними перевагами розроблюваної системи є:

- простота архітектури;
- локальна робота без обов'язкового підключення до хмарних сервісів;
- прямий HTTP-обмін між вебзастосунком та ESP32;
- наявність візуальної LED-індикації стану;
- можливість керування ефектами через вебінтерфейс;
- автоматичне журналювання змін параметрів;
- окреме фіксування дій користувача;
- використання SQLite без необхідності встановлення окремого сервера баз даних.

Недоліком такого підходу є менша масштабованість порівняно з промисловими IoT-платформами. Однак для задачі моніторингу одного апаратно-програмного комплексу це не є критичним. Навпаки, обмеження масштабу дозволяє зробити систему простішою, зрозумілішою та зручнішою для навчального використання.

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
						22
		№ докум.	Підпис	Дата		

Порівняно з універсальними платформами, розроблювана система має перевагу у контрольованості програмного коду. Усі основні процеси реалізовані явно: отримання стану, перевірка доступності пристрою, порівняння параметрів, запис журналу та обробка команд користувача. Це дає змогу легко простежити шлях даних від ESP32 до інформаційної панелі та бази даних.

Ще однією перевагою є можливість подальшого розширення без зміни загальної архітектури. Наприклад, до JSON-відповіді ESP32 можна додати новий параметр, після чого розширити модель стану в ASP.NET Core та додати його відображення на панелі. Аналогічно можна додати новий LED-ефект або нову команду керування. Така модульність робить систему придатною для модернізації.

Узагальнюючи результати порівняння, можна зробити висновок, що розроблювана система займає проміжне місце між простим вбудованим вебінтерфейсом ESP32 та повноцінною промисловою IoT-платформою. Вона не є надто складною для розгортання, але водночас має важливі функції, які відсутні у найпростіших рішеннях: централізовану панель, журнал подій, фільтрацію записів і розмежування джерел змін.

Саме така архітектура відповідає темі роботи, оскільки головний акцент зроблено на програмному забезпеченні керування інформаційною панеллю та журналюванні параметрів. Апаратна частина залишається достатньо простою, але забезпечує реальне джерело даних і дозволяє перевірити роботу програмної частини в практичних умовах.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

У цьому підрозділі обґрунтовується вибір апаратних і програмних засобів, що використовуються для побудови системи керування моніторинговою

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
						23
		№ докум.	Підпис	Дата		

інформаційною панеллю. Вибір виконано з урахуванням практичних вимог до локального IoT-комплексу: стабільності роботи, доступності компонентів, простоти розгортання та можливості подальшої модернізації.

Під час добору засобів враховувалися як технічні, так і експлуатаційні критерії. До технічних критеріїв належать підтримка Wi-Fi, можливість керування адресною LED-стрічкою, наявність зручного HTTP-інтерфейсу та сумісність із сучасними фреймворками веброзробки. До експлуатаційних критеріїв належать простота налаштування, невисока вартість, зручність тестування й можливість використання системи у навчальному середовищі.

Загальна концепція вибору полягає у використанні достатніх, але не надлишкових засобів. Для апаратної частини обрано ESP32 та WS2812B як поєднання функціональності й доступності. Для програмної частини обрано стек ASP.NET Core, Blazor Server, EF Core та SQLite, який забезпечує цілісну реалізацію вебінтерфейсу, серверної логіки, фонових процесів і журналювання в межах одного застосунку.

Мікроконтролер ESP32

Для апаратної частини системи обрано мікроконтролер ESP32. Цей вибір зумовлений наявністю вбудованого Wi-Fi-модуля, достатньою продуктивністю, підтримкою GPIO, доступністю засобів розробки та широкою бібліотечною базою. ESP32 дозволяє одночасно виконувати мережеві функції, обробляти HTTP-запити та керувати LED-стрічкою.

ESP32 є доцільним вибором для IoT-систем, оскільки підтримує бездротове підключення без використання додаткових модулів. У розроблюваному комплексі це дозволяє підключити пристрій до локальної мережі та зробити його доступним для вебзастосунку. Після підключення до Wi-Fi ESP32 отримує IP-адресу від маршрутизатора та може відповідати на HTTP-запити.

У прошивці використовується Arduino Framework для ESP32. Такий підхід спрощує розробку, оскільки надає готові бібліотеки для роботи з Wi-Fi,

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						24
		№ докум.	Підпис	Дата		

HTTP-клієнтом, вебсервером, DNS-сервером і енергонезалежним сховищем. Для збереження параметрів Wi-Fi використовується бібліотека 'Preferences', що дозволяє зберігати SSID і пароль після перезавантаження пристрою.

ESP32 також є зручним для створення автономних вебінтерфейсів. У прошивці використовується вбудований вебсервер, який може обробляти HTTP-запити без залучення зовнішнього сервера. Це дає змогу реалізувати маршрути '/status', '/effect', '/reset' та '/monitoring' безпосередньо на пристрої. Завдяки цьому ESP32 виступає не лише як виконавчий модуль, але і як самостійний мережевий вузол.

Важливою перевагою ESP32 є наявність достатньої кількості GPIO для підключення периферійних пристроїв. У даному комплексі використовується GPIO 4 для передавання даних на LED-стрічку та GPIO 14 для кнопки скидання. Якщо система буде розширюватися, до ESP32 можна підключити додаткові датчики або індикатори, не змінюючи основну логіку вебпанелі.

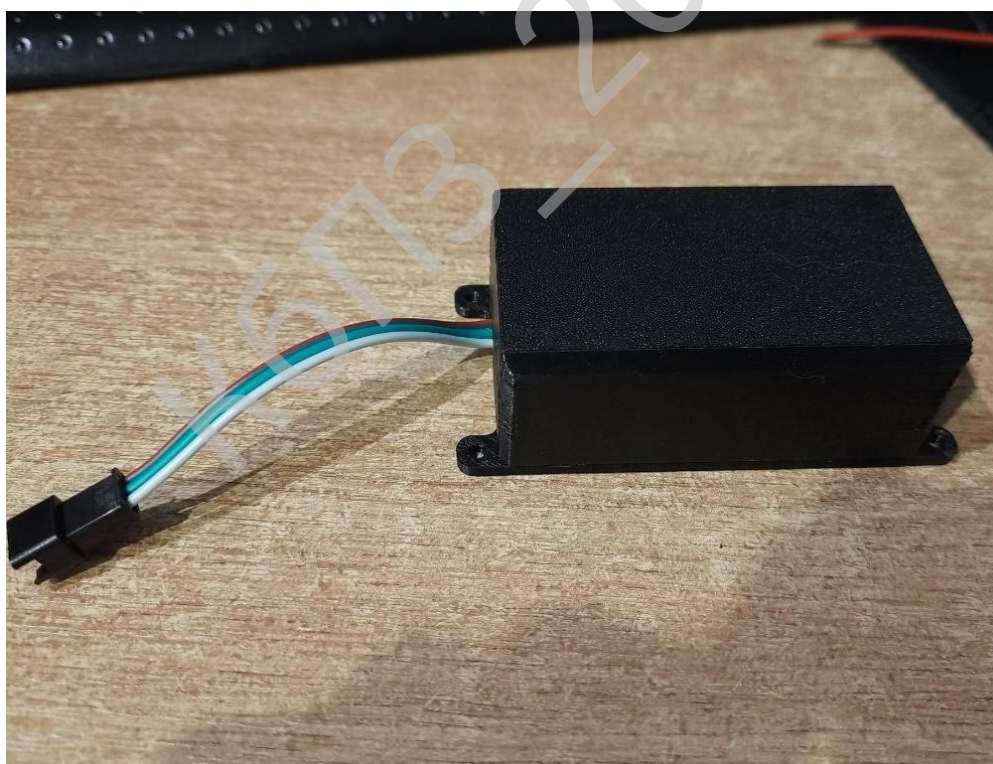


Рисунок 2.1 - Розміщення ESP32 у корпусі апаратного модуля

У системі ESP32 виконує такі функції:

- запуск режиму первинного налаштування;
- створення точки доступу та Captive Portal;
- підключення до Wi-Fi-мережі;
- перевірка доступності Internet;
- формування JSON-відповіді зі станом пристрою;
- приймання команд керування LED-ефектами;
- керування адресною LED-стрічкою WS2812B.

Окремо слід зазначити, що ESP32 добре підходить для прототипування завдяки доступності плат розробки та підтримці середовища PlatformIO. Розробник може швидко змінювати код прошивки, завантажувати його на пристрій і перевіряти результат через Serial Monitor. Це зменшує час налагодження та дозволяє поступово вдосконалювати алгоритм роботи.

Для даної системи важливо, що ESP32 може поєднувати декілька ролей одночасно. Він працює як клієнт Wi-Fi-мережі, як невеликий HTTP-сервер, як контролер LED-стрічки та як пристрій, що виконує періодичну діагностику Internet-з'єднання. Така багатофункціональність дозволяє зменшити кількість апаратних компонентів і зробити макет компактним.

Також ESP32 має достатню популярність у навчальних і аматорських проєктах, тому для нього існує велика кількість прикладів, бібліотек і документації. Це важливо для супроводу системи, оскільки в разі потреби можна швидко знайти інформацію щодо підключення периферії, роботи з Wi-Fi, організації вебсервера або використання енергонезалежної пам'яті.

Адресна LED-стрічка WS2812B та допоміжні апаратні засоби

Для візуальної індикації стану обрано адресну LED-стрічку WS2812B. Її перевагою є можливість індивідуального керування кожним світлодіодом через один цифровий сигнальний провід. Це дозволяє створювати як прості службові індикатори, так і складні динамічні ефекти.

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
						26
		№ докум.	Підпис	Дата		

У розроблюваній системі LED-стрічка виконує роль зовнішнього індикатора стану. Якщо Internet-з'єднання доступне, пристрій відображає зелений пульсуючий ефект. Якщо Internet-з'єднання недоступне, пристрій відображає червоне миготіння. Такий спосіб індикації дозволяє оцінити стан без відкриття вебінтерфейсу.



Рисунок 2.2 - Джерело живлення LED-стрічки та апаратної частини комплексу

Оскільки LED-стрічка може споживати значний струм, для її роботи необхідне відповідне джерело живлення. Потужність джерела живлення має обиратися з урахуванням кількості світлодіодів та максимальної яскравості. Також важливо забезпечити спільну землю між ESP32 і джерелом живлення LED-стрічки, інакше цифровий сигнал керування може працювати нестабільно.

Додатковим апаратним елементом є кнопка скидання до заводських налаштувань. Вона підключається до GPIO 14 і дозволяє очистити збережені Wi-Fi-параметри, після чого пристрій повертається до режиму первинного налаштування. Це важливо у випадку зміни Wi-Fi-мережі або помилкового введення параметрів підключення.

Для адресної LED-стрічки важливо правильно організувати електричне підключення. Сигнальний провід має бути підключений до визначеного GPIO, а земля джерела живлення LED-стрічки повинна бути спільною із землею ESP32. Якщо спільна земля відсутня, цифровий сигнал може сприйматися некоректно, що призводить до випадкових кольорів, мерехтіння або нестабільної роботи ефектів. Потужність джерела живлення залежить від кількості світлодіодів і режиму їх роботи. При максимальній яскравості й білому кольорі споживання струму є найбільшим, тому живлення довгої стрічки безпосередньо від плати ESP32 є неприпустимим. Для стабільної експлуатації доцільно використовувати окремий блок живлення з достатнім запасом по струму.

Під час побудови апаратної частини необхідно також враховувати умови фізичного розміщення компонентів. ESP32, LED-стрічка, джерело живлення та кнопка скидання повинні бути розміщені так, щоб забезпечити зручність експлуатації та доступ до основних елементів. Якщо пристрій використовується як демонстраційний стенд, важливо, щоб LED-індикація була добре видимою, а кнопка скидання була доступною для користувача. Допоміжні апаратні засоби не є центральною частиною програмної розробки, проте вони впливають на стабільність роботи всього комплексу. Нестабільне живлення або ненадійні з'єднання можуть призводити до помилок, які зовні виглядатимуть як програмні збої. Саме тому апаратна частина повинна бути достатньо надійною, щоб результати тестування програмного забезпечення були коректними.



Рисунок 2.3 - Додатковий апаратний модуль для подальшої модернізації комплексу

Мови програмування, фреймворки та середовище розробки

Прошивка ESP32 реалізована мовою C++ з використанням Arduino Framework у середовищі PlatformIO. C++ є типовою мовою для програмування мікроконтролерів, оскільки забезпечує достатню продуктивність, прямий доступ до апаратних ресурсів і широку підтримку бібліотек. PlatformIO спрощує керування залежностями, збирання проєкту та завантаження прошивки на пристрій.

Для керування LED-стрічкою використовується бібліотека FastLED. Вона підтримує WS2812B та надає зручні засоби для роботи з кольорами, яскравістю, HSV-моделлю і динамічними ефектами. Завдяки FastLED реалізовано ефекти `rainbow`, `fill\_rainbow`, `static`, `snake`, `waiting`, `breathe\_green` і `blink\_red`.

Програмна інформаційна панель реалізована з використанням ASP.NET Core 8 та Blazor Server. ASP.NET Core забезпечує створення REST API, фонових служб і конфігурацію HTTP-клієнтів. Blazor Server дозволяє створювати

інтерактивний вебінтерфейс мовою C# без необхідності розробляти окремий JavaScript-додаток.

Для доступу до бази даних використовується Entity Framework Core. Він дає змогу працювати з таблицею журналу через C#-класи, що спрощує додавання, фільтрацію та отримання записів. Як база даних використовується SQLite, оскільки вона зберігає дані у локальному файлі та не потребує окремого серверного процесу.

ASP.NET Core обрано тому, що він дозволяє об'єднати в одному застосунку декілька важливих частин: REST API, фонову службу, конфігурацію HTTP-клієнта, доступ до бази даних і вебінтерфейс. Це зменшує кількість окремих програмних компонентів та спрощує розгортання системи. Усі частини застосунку працюють в одному середовищі виконання .NET.

Blazor Server є зручним варіантом для інформаційної панелі, оскільки дозволяє створювати сторінки інтерфейсу мовою C#. Компоненти Blazor можуть безпосередньо використовувати сервіси застосунку, отримувати дані з бази та оновлювати відображення за таймером. Для даної роботи це доцільно, оскільки основна логіка системи також реалізована мовою C#.

SQLite обрано як локальне сховище журналу через простоту використання. Для роботи з ним не потрібно встановлювати окремий сервер баз даних або виконувати складне адміністрування. Файл бази даних може зберігатися поруч із застосунком, що зручно для навчального стенду або демонстраційного комплексу.

Використання єдиного технологічного стеку на базі .NET спрощує розробку програмної частини. Моделі стану пристрою, записи журналу, контролери API, фоновий сервіс і сторінки інтерфейсу створюються в межах одного проєкту. Це зменшує кількість переходів між різними мовами програмування та робить код більш однорідним.

Окремою перевагою є можливість подальшого розширення серверної частини. У майбутньому до ASP.NET Core застосунку можна додати

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
						30
		№ докум.	Підпис	Дата		

авторизацію, експорт журналу, графіки зміни параметрів, підтримку декількох пристроїв або API для зовнішніх систем. Таким чином, обрані технології не обмежують розвиток проєкту.

2.3 Розгорнута постановка завдання

На основі аналізу предметної області та обраних засобів сформовано постановку завдання на розробку системи. Необхідно створити апаратно-програмний комплекс, який забезпечує моніторинг стану ESP32, керування LED-індикацією та журналювання змін параметрів.

Прошивка ESP32 повинна забезпечувати:

- запуск режиму первинного налаштування у разі відсутності збережених Wi-Fi-параметрів. Такий режим необхідний для того, щоб пристрій міг бути підготовлений до роботи без попереднього внесення мережових даних безпосередньо в код прошивки. Якщо у пам'яті ESP32 немає збереженого SSID або пароля, пристрій повинен автоматично перейти до стану конфігурування, у якому користувач зможе задати параметри підключення через вебінтерфейс;
- створення точки доступу 'ESP32-WiFi-Monitor', через яку користувач зможе підключитися до пристрою під час первинного налаштування. Наявність власної точки доступу робить систему незалежною від попередньої конфігурації локальної мережі та дозволяє виконати налаштування навіть тоді, коли ESP32 ще не має доступу до маршрутизатора. Це особливо важливо при першому запуску або після скидання параметрів;
- роботу Captive Portal для зручного налаштування Wi-Fi. Captive Portal повинен автоматично перенаправляти користувача на сторінку конфігурації або забезпечувати доступ до неї через адресу пристрою. Завдяки цьому користувачу не потрібно знати технічні деталі роботи ESP32, а процес підключення до бездротової мережі стає подібним до налаштування звичайного мережевого обладнання;

- збереження SSID і пароля в енергонезалежній пам'яті пристрою. Це потрібно для того, щоб після перезавантаження або тимчасового вимкнення живлення ESP32 міг автоматично відновити підключення до заданої Wi-Fi мережі. Збереження параметрів у пам'яті усуває необхідність повторного ручного налаштування після кожного запуску та підвищує зручність експлуатації комплексу;
- перехід до режиму моніторингу після успішного підключення до Wi-Fi. У цьому режимі пристрій повинен припинити роботу як точка доступу для налаштування та перейти до штатного режиму функціонування в локальній мережі. Саме в режимі моніторингу ESP32 виконує основні задачі системи: перевіряє доступність Internet, обслуговує HTTP-запити та керує LED-індикацією;
- періодичну перевірку доступності Internet-з'єднання з фіксованим інтервалом часу. У розроблюваній системі така перевірка виконується кожні 5 секунд за допомогою HTTP-запиту до зовнішнього сервісу. Результат перевірки використовується для визначення поточного стану мережі та подальшого вибору відповідного LED-ефекту. Це дозволяє пристрою реагувати на зміну доступності Internet без участі користувача;
- керування LED-ефектами відповідно до стану системи та команд користувача. Прошивка повинна підтримувати службові ефекти для відображення доступності або недоступності Internet-з'єднання, а також додаткові декоративні режими, які можуть бути обрані через інформаційну панель. Такий підхід поєднує діагностичну функцію LED-стрічки з можливістю ручного керування індикацією;
- надання HTTP-маршрутів `/status`, `/effect`, `/reset` і `/monitoring`, які забезпечують взаємодію між ESP32 та програмною інформаційною панеллю. Маршрут `/status` повинен повертати поточний стан пристрою у структурованому форматі, `/effect` - приймати команди зміни LED-ефекту, `/reset` - ініціювати скидання параметрів, а `/monitoring` - повертати пристрій до

автоматичного режиму індикації. Наявність таких маршрутів формує простий прикладний інтерфейс для керування пристроєм.

Програмна частина повинна забезпечувати:

- періодичне опитування ESP32 з інтервалом 5 секунд для отримання актуального стану пристрою. Такий інтервал є достатнім для своєчасного виявлення змін мережевого стану та водночас не створює надмірного навантаження на мікроконтролер і локальну мережу. Опитування повинно виконуватися фоновим сервісом, щоб процес моніторингу не залежав від дій користувача на сторінці вебінтерфейсу;
- відображення стану пристрою на головній сторінці інформаційної панелі. Користувач повинен бачити основні параметри комплексу в зручному для сприйняття вигляді: доступність пристрою, стан Wi-Fi, доступність Internet, SSID, IP-адресу та активний LED-ефект. Такий спосіб подання даних дозволяє швидко оцінити працездатність системи без аналізу службових повідомлень або журналів прошивки;
- керування LED-ефектами через окрему сторінку вебзастосунку. На цій сторінці користувач повинен мати можливість обрати потрібний режим індикації, задати колір для ефектів, які цього потребують, а також повернути пристрій до автоматичного режиму моніторингу. Виділення керування ефектами в окрему сторінку робить інтерфейс більш упорядкованим і відокремлює функції спостереження від функцій керування;
- обробку помилок недоступності пристрою без аварійного завершення роботи вебзастосунку. Якщо ESP32 тимчасово не відповідає на HTTP-запити, програмна частина повинна коректно зафіксувати цей стан і відобразити його користувачу. Така поведінка є важливою для практичної експлуатації, оскільки в реальних умовах пристрій може перезавантажуватися, втрачати Wi-Fi-з'єднання або бути тимчасово вимкненим;
- порівняння нового стану з попереднім для виявлення реальних змін робочих параметрів. Просте періодичне збереження всіх отриманих значень

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						33
		№ докум.	Підпис	Дата		

призвело б до накопичення великої кількості однакових записів, які ускладнювали б аналіз. Тому система повинна визначати, які саме параметри змінилися після чергового опитування, і створювати записи журналу лише для таких змін;

- запис змін до журналу з фіксацією часу події, назви параметра, попереднього значення, нового значення та джерела зміни. Такий формат журналу дозволяє відновити послідовність подій і зрозуміти, як змінювався стан пристрою під час роботи. Журналювання є ключовою функцією системи, оскільки саме воно перетворює інформаційну панель із засобу поточного перегляду на інструмент аналізу історії;

- фільтрацію журналу за параметром і джерелом зміни для зручного пошуку потрібних подій. Користувач повинен мати можливість переглядати не весь журнал одразу, а лише записи, пов'язані з конкретним параметром, наприклад `'internet'`, `'effect'` або `'wifi_connected'`. Також важливим є поділ подій за джерелом: автоматично виявлені зміни повинні відрізнятися від дій, виконаних користувачем через панель керування;

- збереження журналу в базі даних SQLite, що забезпечує постійне накопичення історії подій у локальному файлі. Використання SQLite є доцільним для даної системи, оскільки база даних не потребує окремого серверного процесу, легко розгортається разом із вебзастосунком і достатня для зберігання журналу невеликого апаратно-програмного комплексу. Збереження даних у базі також дозволяє переглядати історію після перезапуску програмної частини.

Журнал змін повинен містити час події, назву параметра, старе значення, нове значення та джерело зміни. Автоматично виявлені зміни повинні мати джерело `'auto'`, а дії користувача - джерело `'user'`.

Результатом виконання поставленого завдання має бути система, у якій усі компоненти працюють узгоджено. ESP32 повинен надавати актуальний стан у форматі JSON, серверна частина повинна коректно обробляти цю відповідь, а інформаційна панель повинна відображати дані у зрозумілому для користувача

вигляді. У разі зміни параметрів система повинна автоматично створювати записи журналу без участі користувача.

До нефункціональних вимог належать простота розгортання, зрозумілість архітектури, можливість локальної роботи та стійкість до тимчасової недоступності ESP32. Якщо пристрій не відповідає на запити, вебзастосунок не повинен завершувати роботу з помилкою, а має відобразити стан недоступності. Після відновлення зв'язку система повинна продовжити опитування.

Постановка завдання також передбачає, що система повинна бути придатною для пояснення та захисту. Це означає, що кожна її функція має бути логічно обґрунтована та пов'язана з темою роботи. Наприклад, LED-індикація демонструє апаратну реакцію на зміну стану, інформаційна панель забезпечує програмне керування, а журналювання підтверджує можливість накопичення історії параметрів.

Таким чином, розгорнута постановка завдання охоплює не лише перелік функцій, але й вимоги до взаємодії між компонентами. У результаті повинна бути створена система, яка працює як єдиний комплекс, а не як набір окремих непов'язаних модулів.

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						35
		№ докум.	Підпис	Дата		

3. ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Функціонування системи починається із запуску ESP32. Під час старту прошивка ініціалізує послідовний порт, LED-стрічку, кнопку скидання та енергонезалежне сховище параметрів. Далі пристрій перевіряє наявність збережених Wi-Fi-даних. Якщо такі дані відсутні, запускається режим первинного налаштування.

У режимі первинного налаштування ESP32 створює точку доступу та вебсторінку для введення параметрів Wi-Fi. Користувач підключається до мережі пристрою, відкриває сторінку налаштування, обирає доступну мережу та вводить пароль. Після успішного підключення дані зберігаються, а пристрій перезавантажується.

У режимі моніторингу ESP32 підключається до заданої Wi-Fi-мережі та запускає вебсервер. Програмна інформаційна панель періодично звертається до маршруту `/status` і отримує JSON-об'єкт з поточними параметрами. Якщо пристрій недоступний, вебзастосунок позначає його як offline. Якщо відповідь отримана, стан оновлюється, а зміни порівнюються з попередніми значеннями.

Керування LED-ефектами здійснюється через маршрут `/effect`. Користувач обирає ефект на сторінці інформаційної панелі, після чого серверна частина надсилає відповідну команду на ESP32. У разі успішного виконання команда записується до журналу як дія користувача.

Перевірка Internet-з'єднання виконується на ESP32 через HTTP-запит до сервісу `'http://clients3.google.com/generate_204'`. Якщо відповідь має код 204, Internet вважається доступним. Якщо відповідь відсутня або код відрізняється, Internet вважається недоступним. Залежно від результату змінюється LED-індикація.

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						36
		№ докум.	Підпис	Дата		

У режимі моніторингу ESP32 фактично виконує роль виконавчого та діагностичного вузла. Він самостійно перевіряє зовнішнє з'єднання, зберігає поточний ефект і відповідає на запити програмної панелі. При цьому складніші функції, такі як збереження історії та фільтрація журналу, винесені на серверну частину. Такий розподіл навантаження є раціональним, оскільки мікроконтролер має обмежені ресурси порівняно з комп'ютером, на якому працює вебзастосунок.

Програмна панель працює за принципом періодичного опитування. Через заданий проміжок часу фоновий сервіс надсилає запит до ESP32 і отримує новий стан. Далі цей стан порівнюється з попереднім. Якщо значення параметрів не змінилися, запис до журналу не створюється. Якщо хоча б один параметр змінився, система формує один або декілька записів журналу. Це дозволяє уникнути дублювання однакових записів і зберігати саме події зміни, а не кожен факт опитування.

Окремий сценарій пов'язаний із діями користувача. Коли користувач змінює LED-ефект, ця дія має бути зафіксована незалежно від того, чи одразу після цього фонове опитування отримає новий стан. Тому користувацькі команди записуються до журналу окремо з джерелом `user`. Це дозволяє відрізнити автоматичні зміни, спричинені станом мережі, від ручних дій оператора.

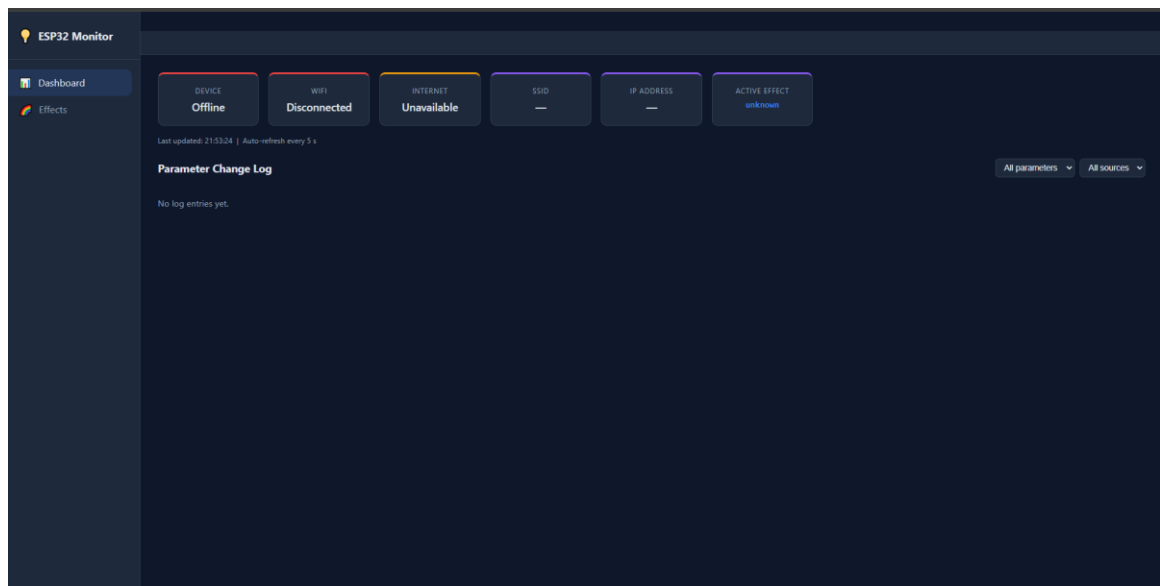


Рисунок 3.1 - Головна сторінка моніторингової інформаційної панелі

Головна сторінка інформаційної панелі відображає поточний стан пристрою, зокрема доступність ESP32, стан Wi-Fi, стан Internet, SSID, IP-адресу та активний ефект. Також на цій сторінці розміщується таблиця журналу змін робочих параметрів.

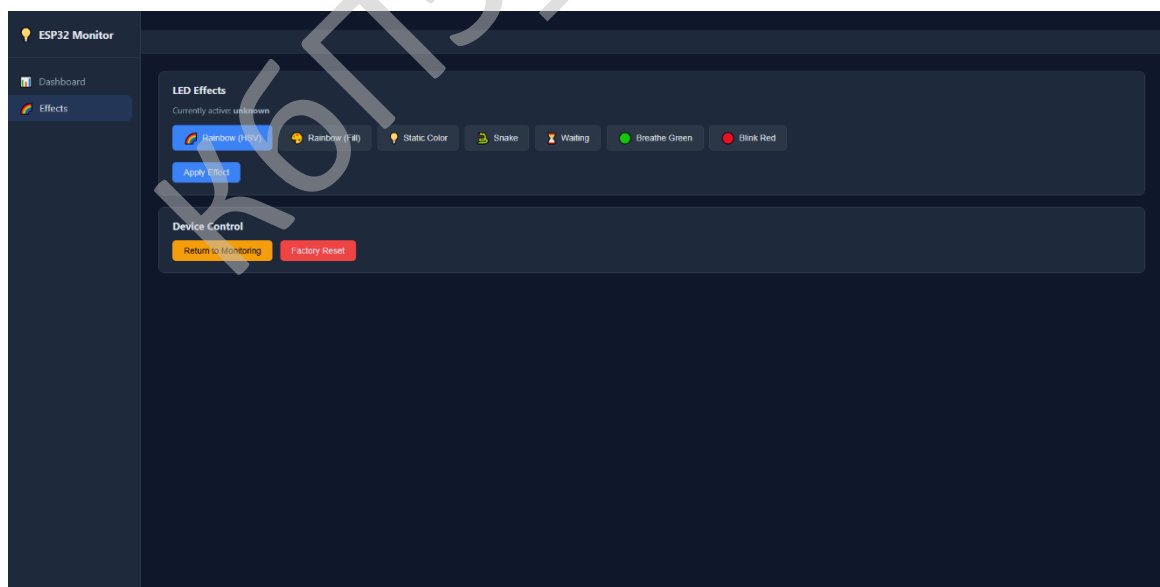


Рисунок 3.2 - Сторінка керування LED-ефектами

Сторінка керування ефектами дозволяє обрати один із доступних режимів індикації, задати колір для статичного ефекту або ефекту `snake`, повернути пристрій до режиму автоматичного моніторингу або виконати Factory Reset.

Наявність окремої сторінки керування ефектами робить систему не тільки засобом спостереження, але й засобом керування. Це важливо для теми роботи, оскільки йдеться саме про програмне забезпечення системи керування моніторинговою інформаційною панеллю. Користувач може перевірити працездатність LED-стрічки, обрати потрібний режим індикації та повернути пристрій до автоматичного режиму без перепрошивання ESP32.

Описане функціонування системи демонструє взаємодію програмних та апаратних компонентів у реальному часі. Зміна стану Internet-з'єднання впливає на LED-ефект, зміна ефекту через вебінтерфейс передається на ESP32, а журнал фіксує відповідну подію. Саме така взаємозалежність компонентів є характерною для апаратно-програмних комплексів.

Важливо, що користувач не повинен контролювати всі процеси вручну. Автоматичне опитування і автоматичне журналювання виконуються фонові. Це підвищує зручність експлуатації системи, оскільки інформація накопичується без постійного втручання оператора.

3.2 Розробка структурної схеми

Структурна схема системи відображає основні компоненти комплексу та зв'язки між ними. До складу системи входять користувач, браузер, Blazor Server інформаційна панель, REST API, фонові служба опитування, HTTP-клієнт ESP32, база даних SQLite, мікроконтролер ESP32 та LED-стрічка WS2812B.

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						39
		№ докум.	Підпис	Дата		

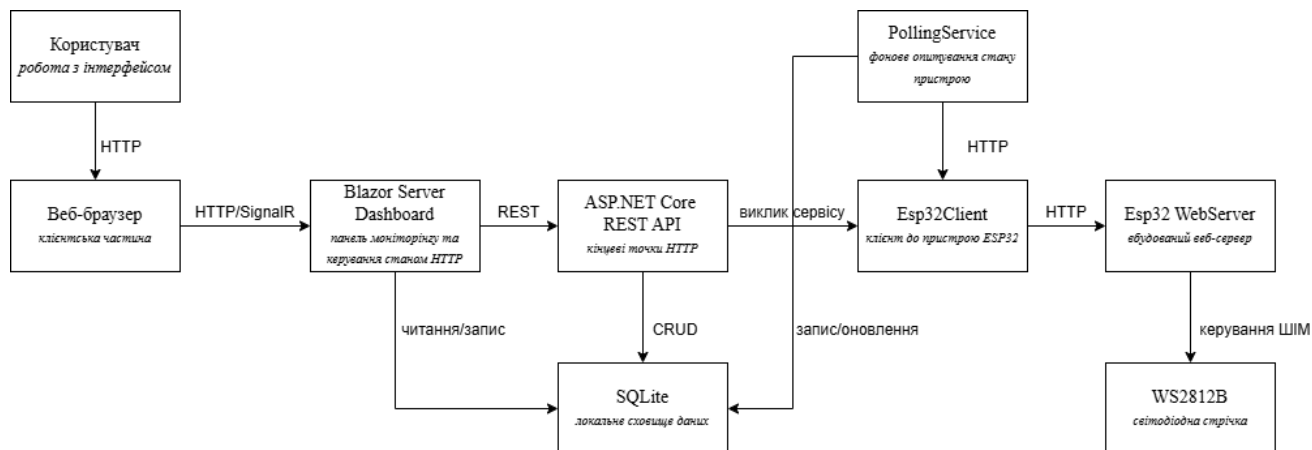


Рисунок 3.3 - Структурна схема системи

Користувач взаємодіє із системою через браузер. Браузер відкриває Blazor Server застосунок, який відображає стан пристрою та журнал змін. REST API приймає команди від інтерфейсу й передає їх на ESP32. Фонова служба 'PollingService' виконує періодичне опитування пристрою та фіксує зміни параметрів.

ESP32 у структурній схемі виступає як окремий мережевий вузол. Він має власний вебсервер і надає HTTP-маршрути для отримання стану та виконання команд. LED-стрічка підключена до ESP32 і не взаємодіє безпосередньо з вебзастосунком. Такий підхід дозволяє розділити задачі: ESP32 відповідає за апаратний рівень, а серверна частина - за моніторинг, журналювання й інтерфейс користувача. База даних SQLite використовується для збереження журналу змін. Вона розміщується локально поруч із вебзастосунком і не потребує окремого сервера. Це спрощує розгортання системи та робить її зручною для навчального використання.

Структурна схема демонструє, що система має модульну побудову. Заміна одного компонента не потребує повної перебудови всього комплексу. Наприклад, у майбутньому замість SQLite можна використати іншу базу даних, а замість Blazor Server - інший вебінтерфейс, залишивши незмінним HTTP-протокол взаємодії з ESP32. Така модульність є важливою ознакою якісного

проектного рішення. Фонова служба опитування є окремим логічним компонентом, тому вона не залежить від того, чи відкритий у користувача браузер. Навіть якщо сторінка інформаційної панелі не переглядається, сервіс може продовжувати опитувати ESP32 та записувати зміни до бази даних. Це підвищує практичну цінність системи, оскільки журнал подій формується постійно.

Структурна схема також дозволяє визначити межі відповідальності кожного компонента. Браузер відповідає тільки за взаємодію з користувачем, серверна частина - за логіку обробки та збереження, ESP32 - за апаратну взаємодію, а база даних - за накопичення історії. Такий поділ є зручним для пояснення роботи системи та для подальшого вдосконалення.

Якщо в майбутньому виникне потреба додати новий пристрій, структурна схема може бути розширена без принципової зміни підходу. Достатньо додати новий екземпляр клієнта або підтримку списку пристроїв у серверній частині. Це підтверджує, що запропонована архітектура має потенціал розвитку.

3.3 Розробка функціональної схеми

Функціональна схема системи описує послідовність взаємодії між компонентами під час моніторингу та керування. Основним процесом є періодичне опитування ESP32. Фонова служба надсилає HTTP-запит `GET /status`, отримує відповідь у форматі JSON, порівнює її з попереднім станом і записує зміни до бази даних.

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
						41
		№ докум.	Підпис	Дата		

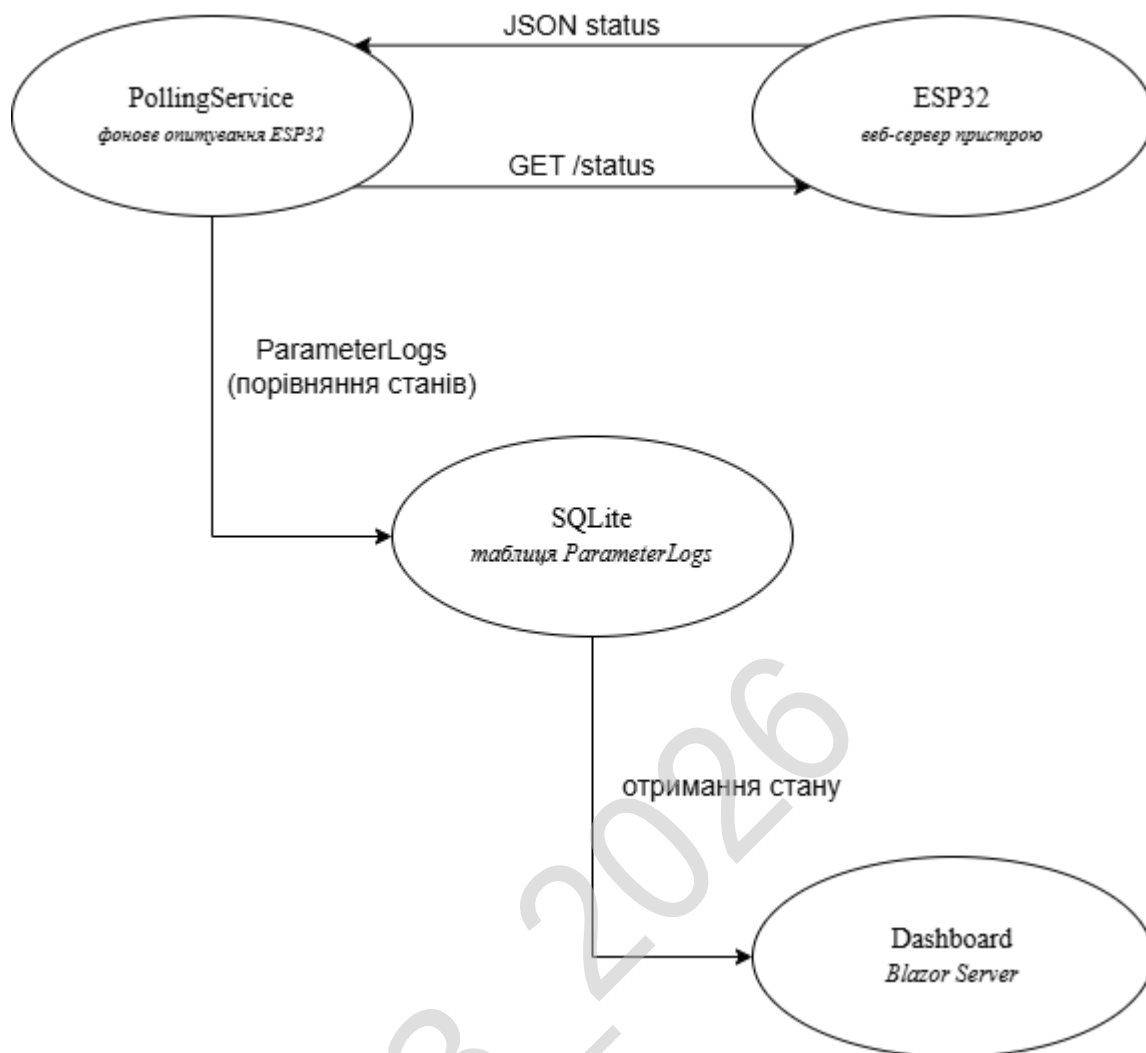


Рисунок 3.4 - Функціональна схема процесу моніторингу та журналювання

Під час керування LED-ефектом користувач обирає потрібний режим на сторінці інформаційної панелі. Blazor-сторінка надсилає запит до REST API, після чого контролер формує HTTP-запит до ESP32. Якщо команда виконана успішно, у базі даних створюється запис журналу з джерелом `user`. Функціональна схема також включає сценарій повернення до автоматичного моніторингу. У цьому випадку користувач натискає відповідну кнопку, REST API надсилає команду `/monitoring` на ESP32, а активний ефект повертається до режиму, у якому колір LED-стрічки визначається станом Internet-з'єднання.

Механізм журналювання є центральним елементом функціональної схеми. Для кожної зміни створюється запис, який містить час, параметр, старе значення, нове значення та джерело. Така структура дозволяє аналізувати як автоматичні зміни стану, так і дії користувача. Функціональна схема дозволяє простежити повний шлях даних. Спочатку ESP32 формує поточний стан на основі власних змінних, після чого віддає його у відповідь на HTTP-запит. Далі серверна частина перетворює JSON-відповідь у модель 'DeviceStatus', порівнює її з попереднім станом і створює об'єкти 'ParameterLog'. Після збереження у SQLite ці записи стають доступними для відображення на інформаційній панелі.

Перевагою такого підходу є те, що журнал не залежить від зовнішніх сервісів. Усі дані зберігаються локально, тому система може працювати у межах локальної мережі. Це особливо важливо для лабораторного стенду, де доступ до Internet може бути нестабільним або навмисно вимикатися для перевірки реакції пристрою.

У функціональній схемі також передбачено обробку помилок. Якщо ESP32 не відповідає, система не створює некоректний запис про зміну всіх параметрів, а позначає пристрій як недоступний. Такий підхід запобігає спотворенню журналу та дозволяє окремо аналізувати доступність самого пристрою. Функціональна схема є важливою не тільки для опису поточної реалізації, але й для перевірки її коректності. Якщо послідовність дій визначена чітко, легше виявити можливі помилки, наприклад повторне журналювання однакових значень або неправильну обробку недоступності пристрою. Тому схема виступає не лише ілюстрацією, а й інструментом аналізу проектного рішення. Завдяки функціональній схемі можна показати, що журналювання виконується не випадково, а як результат порівняння станів. Це важлива відмінність від простого періодичного запису всіх отриманих даних. У розроблюваній системі журнал містить саме зміни, що робить його більш компактним та інформативним.

3.4 Розробка діаграми процесів

Діаграма процесів описує навігацію користувача між екранами вебзастосунку ESP32 Monitor і основні сценарії роботи з моніторингом, журналом та керуванням LED-індикацією. На відміну від блок-схем алгоритмів (розділ 4), тут акцент зроблено на інтерфейсному рівні: які стани екрана доступні, як між ними перемикаються, де користувач ініціює дію, а де дані оновлюються у фоні без його участі.

У графічному поданні, узгодженому з прикладами оформлення дипломних робіт, усі вузли зображено у вигляді еліпсів (овалів); стрілки показують можливі переходи. Двонапрямлені зв'язки між центральним вузлом і гілками означають, що після перегляду підрозділу користувач може повернутися до головного інтерфейсу без перезапуску браузера. Центральний вузол відповідає головному каркасу застосунку (меню та загальна навігація); від нього відгалужуються головна сторінка моніторингу з журналом, сторінка ефектів і довідковий розділ про авторське право.

Фонове опитування ESP32 і запис у SQLite подій з джерелом `auto` на діаграмі процесів не виділено окремим еліпсом, щоб не перевантажувати рисунок: ці процеси виконуються на сервері паралельно з відображенням головної сторінки й відображаються користувачу як оновлення таблиці журналу та карток стану.

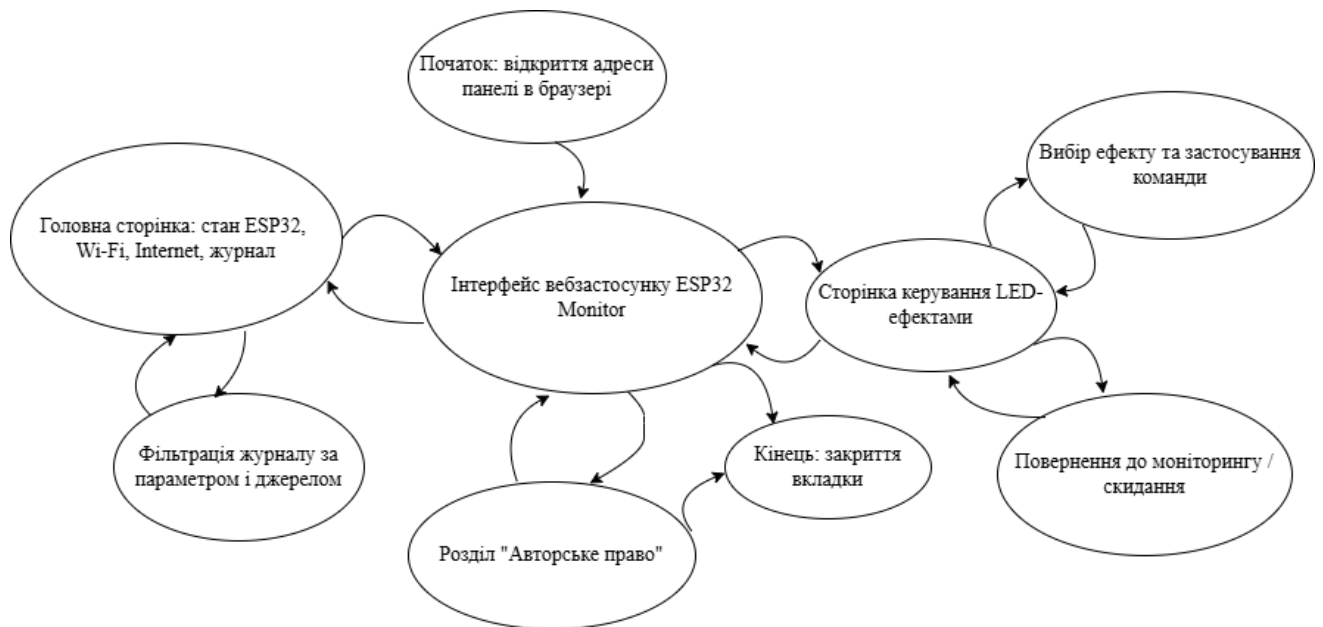


Рисунок 3.4 - Діаграма процесів роботи моніторингової інформаційної панелі

Узгодження діаграми з реалізацією: пункти меню Blazor-застосунку відповідають переходам між вузлами; REST API і 'PollingService' забезпечують дані для головної сторінки та прийом команд зі сторінки ефектів. Якщо ESP32 недоступний, на головній сторінці відображається стан offline, при цьому навігація між розділами зберігається.

3.5 Розробка принципової схеми пристрою

Принципова схема пристрою відображає підключення основних апаратних компонентів. У складі пристрою використовується плата ESP32, адресна LED-стрічка WS2812B, кнопка скидання та джерело живлення. Сигнальний вхід LED-стрічки підключається до GPIO 4 ESP32. Кнопка скидання підключається до GPIO 14 та працює з внутрішнім підтягуванням до живлення.

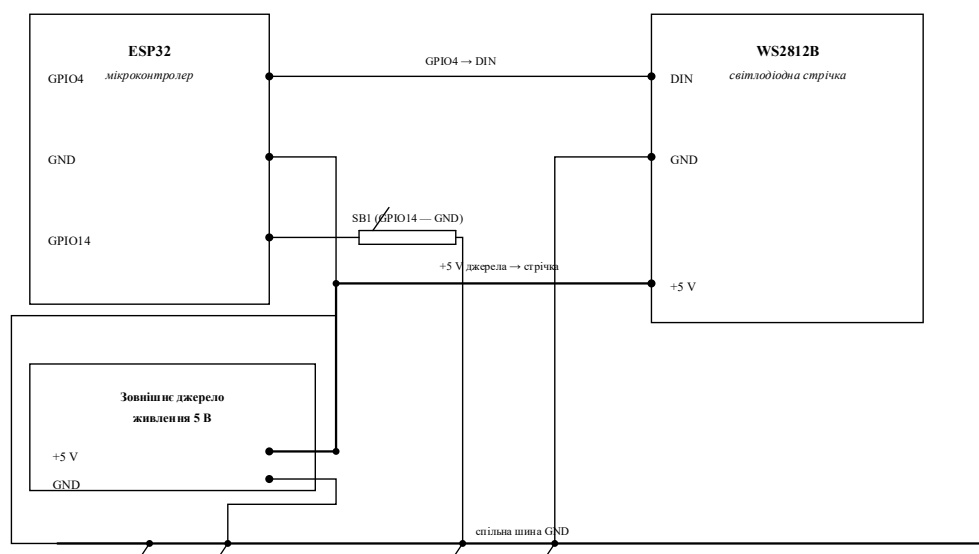


Рисунок 3.5 - Принципова схема підключення ESP32, LED-стрічки та кнопки скидання

Підключення LED-стрічки потребує врахування струмового навантаження. Один світлодіод WS2812B при максимальній яскравості білого кольору може споживати значний струм, тому живлення стрічки не слід виконувати безпосередньо від плати ESP32. Для стабільної роботи використовується окреме джерело живлення відповідної потужності. Обов'язковою умовою коректної роботи є спільна земля ESP32 та джерела живлення LED-стрічки. Без спільної землі цифровий сигнал керування може сприйматися некоректно, що призводитиме до мерехтіння або неправильного відображення кольорів. Кнопка скидання використовується для повернення пристрою до режиму первинного налаштування. При утриманні кнопки прошивка очищає збережені Wi-Fi-параметри та перезавантажує ESP32. Після цього пристрій знову створює точку доступу ESP32-WiFi-Monitor, що дозволяє налаштувати підключення до іншої мережі.

Додатково варто враховувати, що сумарний струм стрічки залежить не

лише від кількості світлодіодів, а й від обраного ефекту та середньої яскравості: динамічні режими з великою кількістю одночасно увімкнених пікселів можуть створювати короточасні піки навантаження. Тому при підборі джерела живлення доцільно закладати запас потужності відносно розрахункового максимуму, а також уникати довгих тонких провідників на лінії живлення, які підсилюють падіння напруги. На практиці саме нестабільне живлення часто є причиною «дивної» поведінки адресної стрічки, коли здається, що помилка в програмному коді, тоді як джерело проблеми - апаратне.

Принципова схема має враховувати не тільки логічне підключення, а й особливості живлення. ESP32 зазвичай живиться через USB або стабілізоване джерело відповідної напруги, тоді як LED-стрічка може потребувати окремого 5-вольтового живлення з більшим струмом. При великій кількості світлодіодів доцільно подавати живлення на стрічку не лише з одного кінця, а за потреби дублювати живлення на довгих ділянках, щоб уникнути падіння напруги. Для підвищення надійності схеми доцільно передбачити захист від помилкового підключення та механічно надійне з'єднання провідників. У навчальному макеті це може бути виконано за допомогою клемників або роз'ємів. Важливо, щоб сигнальний провід від ESP32 до LED-стрічки був достатньо коротким і не проходив поруч із джерелами сильних завад. У деяких випадках корисно передбачити послідовний резистор невеликого номіналу на лінії даних між мікроконтролером і першим пікселем стрічки як елемент узгодження рівнів і обмеження струмів на фронтах сигналу, а також конденсатор достатньої ємності біля першого сегмента стрічки для локального згладжування імпульсних струмів. Хоча для навчального макета такі рішення можуть бути спрощені, їх згадка у принциповій схемі демонструє розуміння типових вимог до якісного підключення WS2812B і знижує ризик нестабільної роботи під час демонстрації.

З погляду подальшої модернізації принципова схема може бути доповнена датчиками або іншими виконавчими елементами. Наприклад, до ESP32 можна підключити датчик температури або реле, а програмну панель

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
						47
		№ докум.	Підпис	Дата		

розширити новими параметрами. Це підтверджує, що обрана схема є базовою, але придатною для розвитку. Принципова схема також допомагає уникнути неоднозначності під час складання макета. Якщо всі з'єднання визначені заздалегідь, легше перевірити правильність підключення та знайти помилку у разі нестабільної роботи. Для апаратно-програмного комплексу це особливо важливо, оскільки помилка може виникнути як у коді, так і у фізичному підключенні.

У фінальному оформленні принципову схему доцільно виконати так, щоб на ній були чітко позначені виводи ESP32, лінія даних LED-стрічки, живлення та спільна земля. Рисунок 3.5 полегшить сприйняття апаратної частини роботи та покаже, що програмне забезпечення розглядається у зв'язку з реальним пристроєм. Додатково на схемі можна позначити кнопку скидання та її підключення до відповідного GPIO, щоб читач одразу бачив повний контур «налаштування - робота - повернення до factory mode». Це зменшує кількість усних пояснень під час захисту й робить апаратну частину документованою на рівні, який відповідає вимогам до оформлення інженерних матеріалів.

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
		№ докум.	Підпис	Дата		48

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

Розглянемо практичну реалізацію розроблюваного апаратно-програмного комплексу. На відміну від чисто теоретичного опису архітектури, цей розділ показує, як проектні рішення перетворюються на конкретні алгоритми, цикли обробки, формати даних і послідовність дій користувача та програмних модулів. Для підтвердження правильності прийнятих рішень використовуються блок-схеми, діаграми діяльності у стилі UML, фрагменти коду та приклади тестових сценаріїв, які можна відтворити на стенді.

Реалізація роботи передбачає практичне втілення проектних рішень, описаних у попередньому розділі. Для розроблюваного комплексу реалізація охоплює дві основні частини: прошивку ESP32 (Arduino-фреймворк, бібліотека FastLED, вбудований вебсервер, HTTP-клієнт для перевірки Internet) та серверну інформаційну панель (ASP.NET Core, Blazor Server, Entity Framework Core, SQLite, фонове опитування пристрою). Експериментальна перевірка полягає у послідовному проходженні режимів роботи: первинне налаштування Wi-Fi, режим моніторингу, зміна LED-ефектів, імітація втрати Internet, перезапуск пристрою та аналіз записів журналу.

Прошивка ESP32 відповідає за роботу апаратного рівня: підключення до Wi-Fi, запуск режиму первинного налаштування, перевірку доступності Internet-з'єднання, обробку HTTP-запитів і керування LED-стрічкою. Серверна частина відповідає за періодичне опитування пристрою, відображення стану, обробку команд користувача та запис змін робочих параметрів до бази даних SQLite. Вірність прийнятих проектних рішень підтверджується тим, що кожен компонент виконує окрему, чітко визначену роль: ESP32 не накопичує повну історію подій, а серверна частина не керує стрічкою безпосередньо, а лише

передає команди через HTTP.

Практична реалізація системи побудована таким чином, щоб мінімізувати залежність між частинами комплексу. Прошивка ESP32 може працювати автономно та надавати власний вебінтерфейс для налаштування й діагностики. Серверна панель може продовжувати роботу при тимчасовій недоступності пристрою, не втрачаючи працездатність цілого застосунку. Важливим результатом є підтримка двох типів потоків подій: автоматичного (опитування, виявлення змін стану) та користувацького (керування ефектами, скидання), які розрізняються полем джерела у журналі ('auto' та 'user').

4.1 Алгоритмічна реалізація роботи системи та побудова блок-схем

Перед розглядом блок-схем доцільно коротко пояснити їх призначення у контексті інженерної документації. Блок-схема є графічним представленням алгоритму за допомогою умовних позначень: еліпс позначає початок і кінець, прямокутник - обчислення або дію, ромб - перевірку умови, паралелограм - введення або виведення даних. Такі схеми дозволяють перевірити повноту розгалужень, відсутність «завислих» станів і відповідність програмної реалізації логіці моніторингу. Додатково у сучасній практиці часто використовують UML-діаграми діяльності (activity diagrams), які семантично близькі до блок-схем і зручні для відображення паралельних процесів; у даній роботі основний акцент зроблено на блок-схемах, а діаграми нижче можна експортувати з редактора у вигляді рисунків для вставки у Word.

Основний алгоритм роботи прошивки ESP32 починається з ініціалізації апаратних і програмних компонентів: послідовний порт для діагностики, ініціалізація масиву пікселів FastLED, налаштування GPIO для лінії даних WS2812B та кнопки скидання, ініціалізація сховища 'Preferences' для збереження SSID і пароля. Далі виконується перевірка наявності збережених Wi-Fi-даних. Якщо дані відсутні або підключення неможливе, активується режим

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						50
		№ докум.	Підпис	Дата		

первинного налаштування (точка доступу, Captive Portal). Якщо підключення успішне, запускається режим моніторингу з реєстрацією HTTP-маршрутів, циклічним оновленням LED-ефектів і періодичною перевіркою Internet.

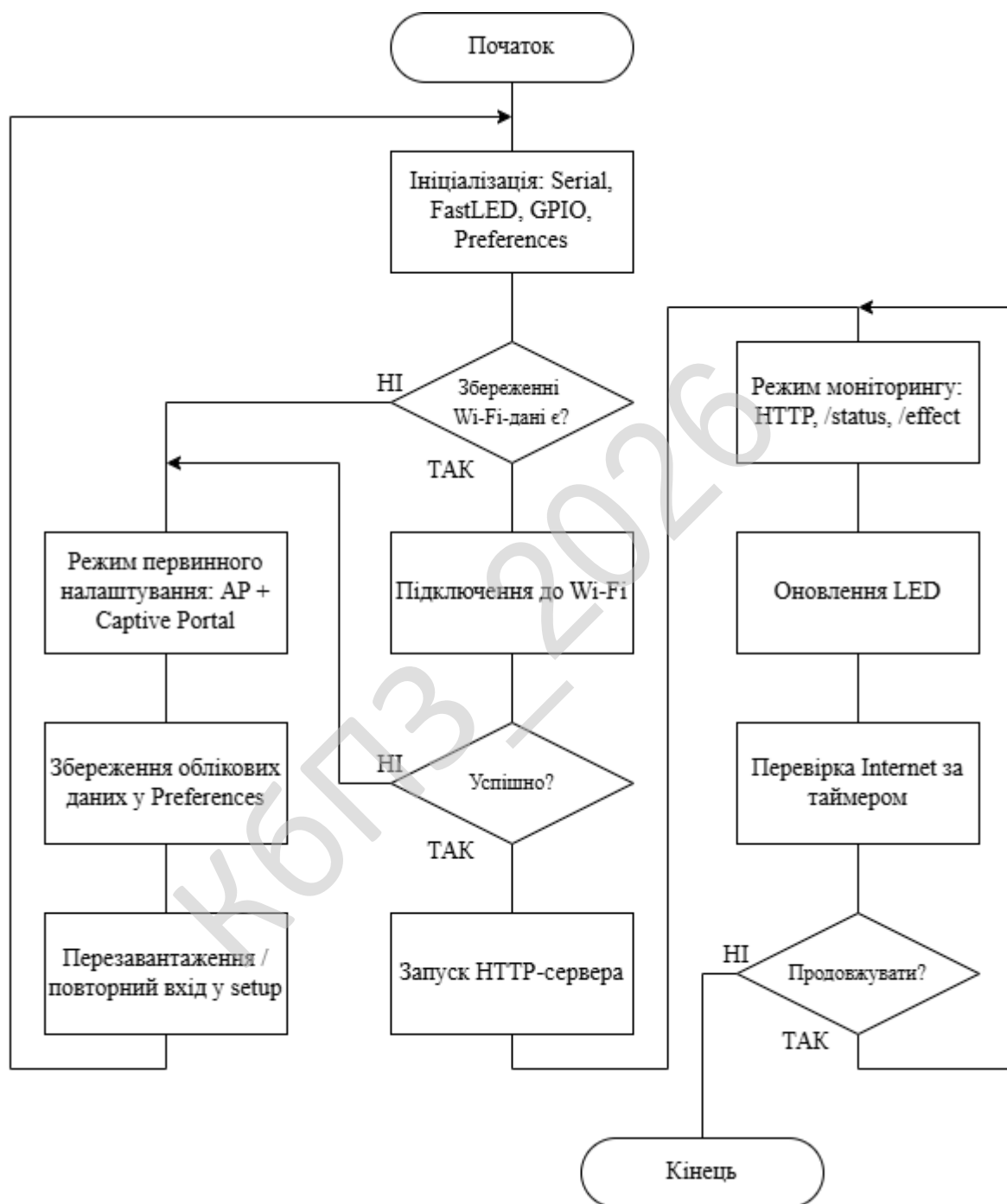


Рисунок 4.1 - Блок-схема основного алгоритму роботи ESP32

У режимі моніторингу ключовим є маршрут 'GET /status', який повертає

поточний стан у форматі JSON. Приклад структури відповіді (поля можуть доповнюватися залежно від версії прошивки):

```
{
  "wifi_connected": true,
  "ssid": "HomeWiFi",
  "ip": "192.168.1.55",
  "internet": true,
  "effect": "breathe_green"
}
```

Формат JSON обрано через лаконічність, підтримку в Arduino та нативну десеріалізацію в ASP.NET Core. На сервері відповідність полів JSON властивостям моделі забезпечується атрибутами серіалізації, що зменшує ризик помилок при перейменуванні полів у прошивці.

Алгоритм серверної частини базується на фоновому опитуванні ESP32. Фонова служба ('BackgroundService') з інтервалом 'PollingIntervalMs' виконує 'GET' до 'http://<ESP32>/status', десеріалізує JSON у модель стану пристрою, порівнює з попереднім знімком і за наявності відмінностей створює записи 'ParameterLog' з джерелом 'auto'. Якщо пристрій недоступний, служба не повинна «падати»: вона фіксує недоступність і очікує наступного циклу.

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						52
		№ докум.	Підпис	Дата		

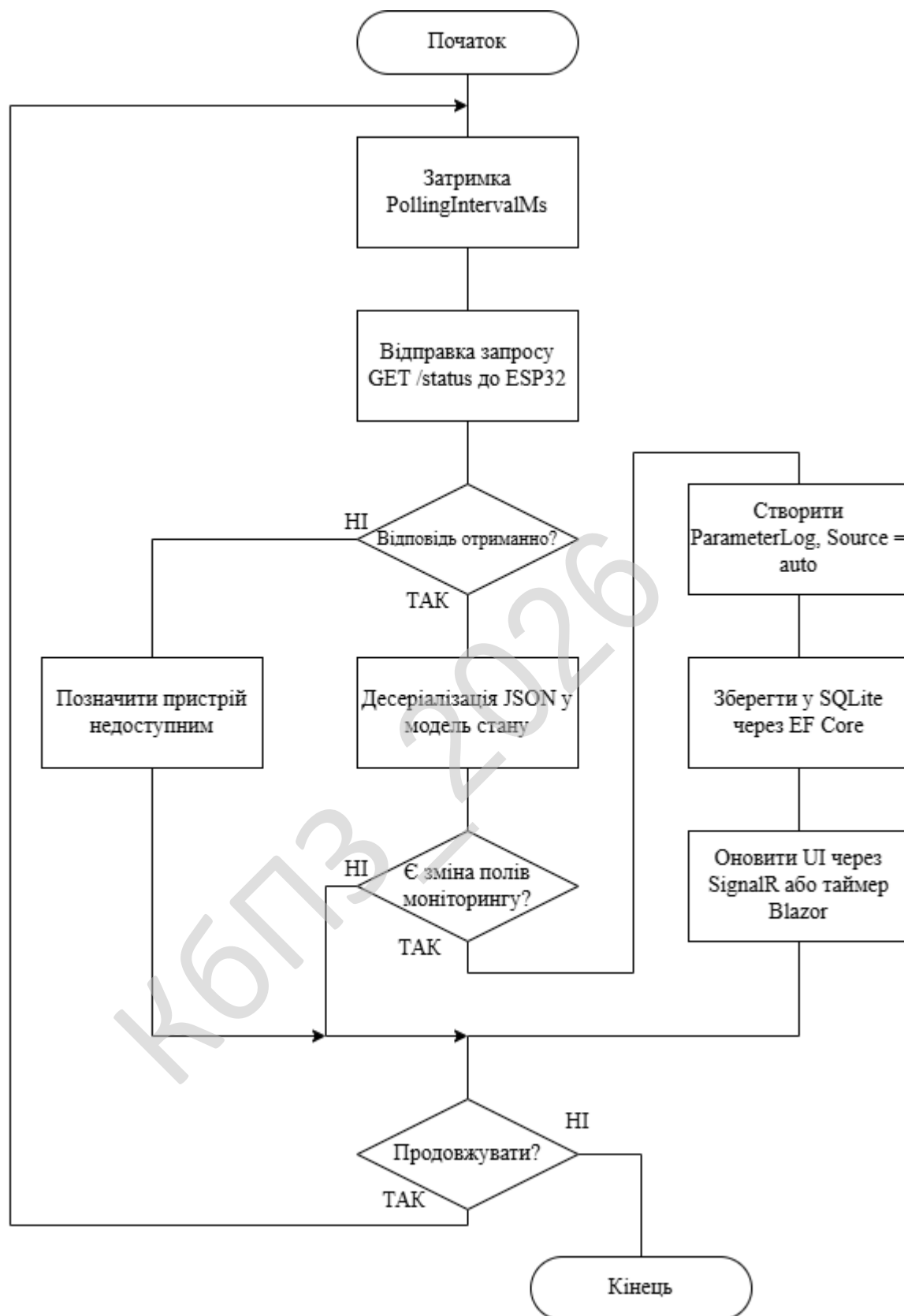


Рисунок 4.2 - Блок-схема алгоритму журналювання змін робочих параметрів

Фрагмент псевдокоду серверного циклу:

```
while (!stoppingToken.IsCancellationRequested)
{
    await Task.Delay(pollingIntervalMs, stoppingToken);
    var newStatus = await esp32Client.GetStatusAsync(stoppingToken);
    if (newStatus is null)
    {
        deviceState.SetReachable(false);
        continue;
    }
    deviceState.SetReachable(true);
    var changes = deviceState.CompareAndApply(newStatus);
    foreach (var c in changes)
        await db.ParameterLogs.AddAsync(ParameterLog.Auto(c),
stoppingToken);
    await db.SaveChangesAsync(stoppingToken);
}
```

Окремий алгоритм пов'язаний із керуванням LED-ефектами. Користувач на сторінці інформаційної панелі обирає режим індикації; Blazor-компонент викликає власний REST API ('EffectController'), який через 'HttpClient' звертається до ESP32 на маршрут '/effect'. При успіху оновлюється кеш стану на сервері та створюються записи журналу з 'Source = user'. Для ефектів 'static' та 'snake' додатково передається параметр кольору, що також журналюється як окремі поля ('static_color', 'snake_color'), якщо це передбачено реалізацією.

Діаграма 4.3 послідовності для сценарію «користувач змінює ефект»:

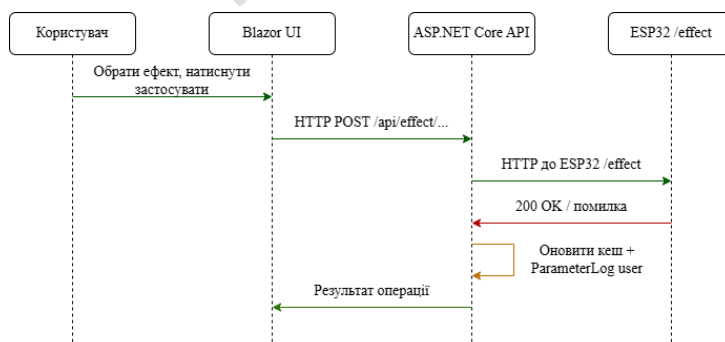


Рисунок 4.3 - Діаграма послідовності зміни LED-ефекту через інформаційну панель

Алгоритмічна реалізація також враховує зменшення зайвих записів у базі: якщо значення `internet` не змінилося між двома циклами, новий рядок журналу не створюється. Це підтверджується експериментально: при стабільному Internet обсяг таблиці зростає повільно, а при імітації обриву з'являються чіткі події переходу `true` → `false` та зворотно.

У прошивці важливо розділити обробку мережевих запитів і оновлення LED: надто часті мережеві операції можуть впливати на плавність анімації, тому перевірка Internet виконується з більшим періодом, ніж кадр оновлення стрічки. Такий баланс є типовим для вбудованих систем і підтверджує коректність проектного компромісу між відгуком вебінтерфейсу та якістю індикації.

4.2 Надійність, обробка помилок та захист програмного забезпечення

Захист і надійність у межах даної роботи розглядаються прагматично: локальна мережа, відкритий HTTP між ПК і ESP32, відсутність зовнішніх користувачів. У таких умовах основні ризики - випадкове спотворення журналу через необроблений виняток, некоректна інтерпретація відповіді пристрою, перевантаження журналу «шумом» та втрата цілісності бази при аварійному завершенні процесу. Тому реалізація спирається на обробку помилок, розмежування джерел подій, обмеження параметрів запитів до журналу та збереження даних у файлі SQLite.

Першим елементом надійності є обробка недоступності ESP32. Якщо `HttpClient` отримує тайм-аут, помилку DNS або відмову з'єднання, серверна частина не повинна припиняти роботу. Коректна поведінка: логування технічної причини (для розробника), встановлення прапорця «пристрій офлайн» у моделі для UI, пропуск створення «фейкових» змін усіх полів одночасно. Після відновлення зв'язку опитування продовжується без ручного перезапуску застосунку.

Другим елементом є розмежування `auto` і `user`. Це не лише зручність

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						55
		№ докум.	Підпис	Дата		

для аналізу, а й елемент логічного захисту: можна відфільтрувати дії оператора та відокремити їх від автоматичних коливань стану мережі. У випускній роботі доцільно навести приклад: зміна `effect` після натискання кнопки завжди має `user`, тоді як зміна `internet` після опитування - `auto`.

Третім елементом є цілісність журналу в SQLite. Використання EF Core дозволяє застосовувати транзакційну семантику `SaveChanges`: або запис потрапляє у файл бази цілком, або операція відкочується при помилці. Для довготривалої експлуатації рекомендовано періодично копіювати файл `.db`, перевіряти наявність вільного місця на диску та уникати ручного редагування файлу сторонніми програмами під час роботи застосунку.

Четвертим напрямом є обмеження некоректних запитів до API журналу. Наприклад, параметри пагінації `page` та `pageSize` мають перебувати у допустимих межах, щоб один запит не зчитував надто великий обсяг історії. Це знижує ризик випадкового або навмисного навантаження на диск і пам'ять.

П'ятим напрямом є модель загроз для локального HTTP. У навчальному стенді шифрування TLS між ПК і ESP32 часто не налаштовується через обмеження сертифікатів на мікроконтролері. У реальному середовищі доцільно обмежити доступ до портів маршрутизатором (Wi-Fi із паролем, ізоляція гостьової мережі), не виносити панель у публічний Internet без reverse-proxy та автентифікації. Для підвищення рівня захисту в подальшому можна додати облікові записи користувачів, політику паролів, HTTPS для панелі та токени для команд керування.

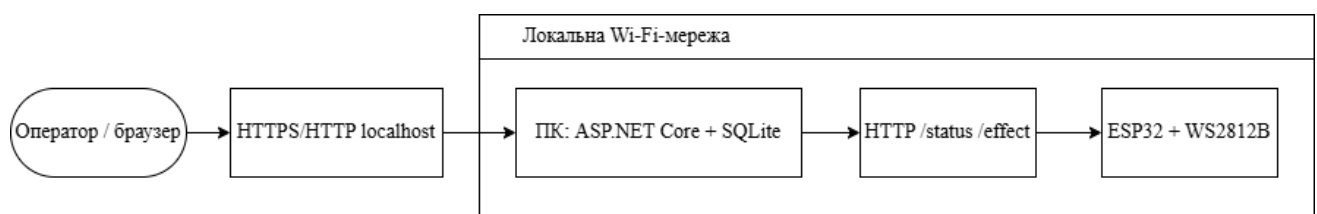


Рисунок 4.4 - Діаграма розгортання логічних вузлів

Надійність також підтверджується ізоляцією відповідальності: 'Esp32Client' інкапсулює мережеві деталі, фонові служби - тільки розклад, контролери - маршрути API, DbContext - персистентність. Такий поділ відповідає принципам модульності та полегшує локалізацію дефекту під час експериментів.

Окремим напрямом захисту є перевірка вхідних параметрів для ефектів і кольорів. Навіть якщо ESP32 частково «пробачає» некоректні значення, серверна валідація запобігає запису семантично безглуздих подій у журнал і зменшує ризик непередбачуваної поведінки індикації.

Також важливо враховувати резервне копіювання та контроль доступу до каталогу з файлом SQLite. Журнал є доказом коректної роботи системи під час демонстрації та захисту; його втрата знижує цінність експериментальної частини роботи.

КБПЗ - 2026

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						57
		№ докум.	Підпис	Дата		

5 ПІДГОТОВКА ТА ВПРОВАДЖЕННЯ СИСТЕМИ В ЕКСПЛУАТАЦІЮ

Впровадження системи в експлуатацію передбачає послідовну підготовку апаратної частини, збирання та завантаження прошивки, первинне налаштування Wi-Fi через Captive Portal, конфігурацію серверної панелі, запуск застосунку та перевірку журналювання. Методика описана так, щоб комплекс можна було відтворити на іншому робочому місці без прихованих кроків.

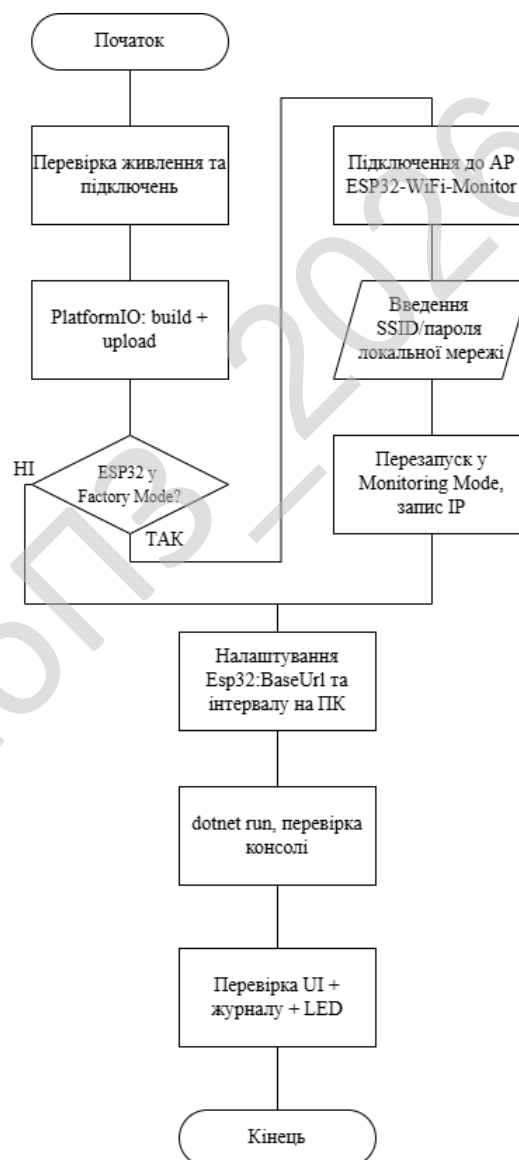


Рисунок 5.1 - Блок-схема етапів впровадження системи в експлуатацію

5.1 Передумови та підготовка робочого місця

Перед монтажом необхідно підготувати: плату ESP32, адресну стрічку WS2812B, стабільне джерело живлення для стрічки (з запасом струму), спільну землю між ESP32 і живленням стрічки, USB-кабель для прошивки та діагностики, ПК з установленим .NET SDK 8 (або сумісною версією), Visual Studio / VS Code з розширенням PlatformIO для проєкту прошивки, доступ до локальної Wi-Fi-мережі. Варто заздалегідь визначитися з IP-діапазоном домашньої або лабораторної мережі: ESP32 і ПК повинні «бачити» один одного; ізольовані VLAN або жорстка клієнтська ізоляція на маршрутизаторі можуть блокувати HTTP між пристроями.

Рекомендовано зробити чек-лист перевірок: цілісність кабелів, відсутність короткого замикання на лінії живлення стрічки, коректність рівня сигналу DATA (за потреби резистор узгодження на першому пікселі згідно з рекомендаціями виробника стрічки), відсутність довгих «кільцевих» проводів живлення, що викликають просадку напруги. Після цього виконується підключення: GPIO 4 це лінія даних до DIN, GPIO 14 це кнопка скидання до землі з внутрішнім підтягуванням, живлення 5 В стрічки окремо від USB ESP32.

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						59
		№ докум.	Підпис	Дата		

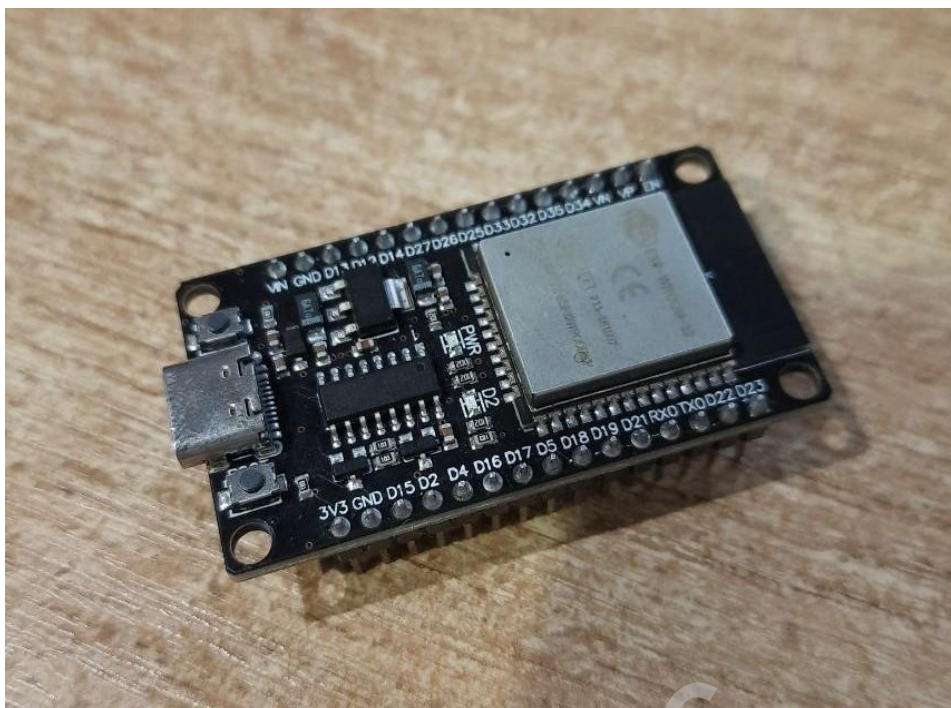


Рисунок 5.2 - Загальний вигляд модуля



Рисунок 5.3 - Макет стрічки



Рисунок 5.4 - Джерело живлення

5.2 Збирання, прошивка та діагностика ESP32

Збирання прошивки у PlatformIO виконується стандартною командою ``pio run``; завантаження - ``pio run -t upload``. Після прошивання обов'язково відкрити Serial Monitor (``pio device monitor``) і переконатися, що у повідомленнях відображається очікуваний режим (Factory або Monitoring), а також IP-адреса у режимі моніторингу. Якщо IP не з'являється, слід перевірити облікові дані Wi-Fi, рівень сигналу та налаштування маршрутизатора (MAC-фільтри, гостьова мережа).

У разі необхідності повторного первинного налаштування використовується утримання кнопки скидання (логіка описана у розділі 3): очищаються збережені параметри Wi-Fi, після перезапуску знову з'являється

точка доступу `ESP32-WiFi-Monitor`. Це важливий експериментальний сценарій для підтвердження того, що система коректно повертається до контрольованого початкового стану.

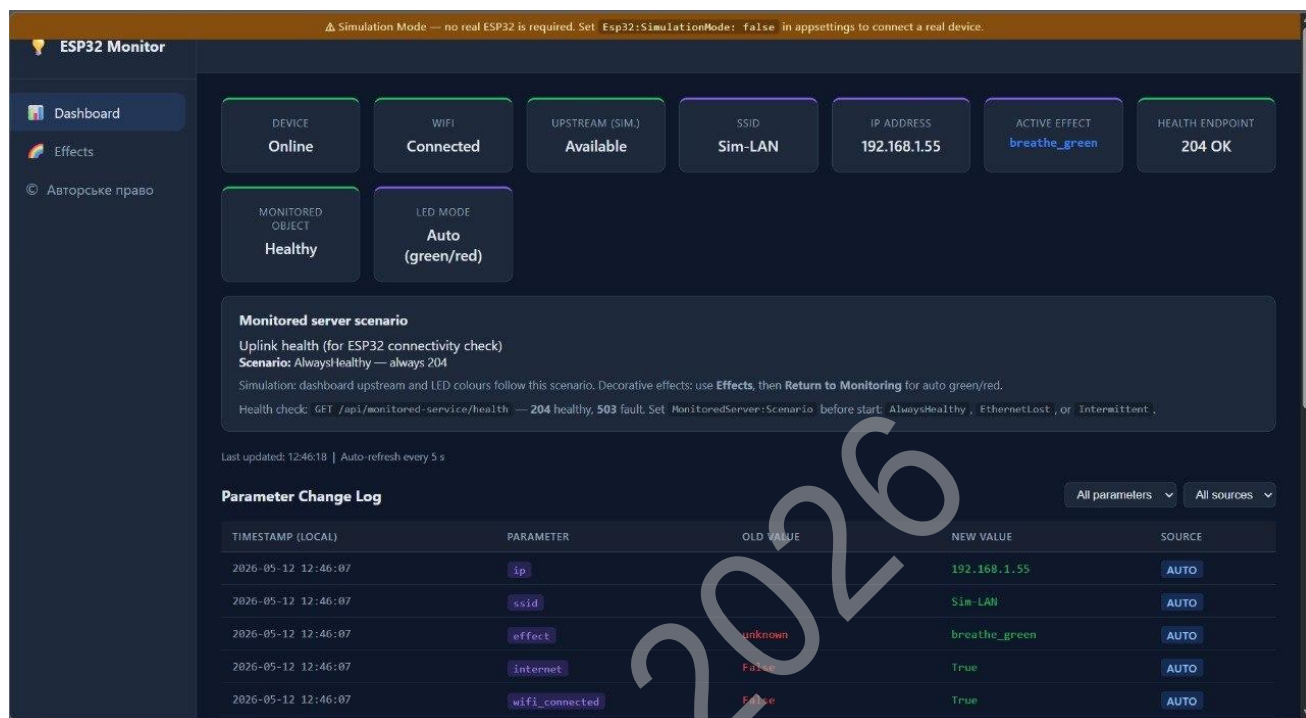


Рисунок 5.5 - Екран демонструє тестовий режим або перевірку на столі

5.3 Налаштування Wi-Fi, Captive Portal та інтеграція з панеллю

Користувач підключається до мережі `ESP32-WiFi-Monitor`, відкриває сторінку налаштування (на мобільному пристрої Captive Portal часто відкривається автоматично). Після введення SSID і пароля локальної мережі ESP32 зберігає параметри в `Preferences` і перезавантажується. На ПК у файлі конфігурації застосунку вказується `Esp32:BaseUrl`, наприклад `http://192.168.1.55`, та `PollingIntervalMs`, наприклад `5000`. Додатково можна задати `App:BaseUrl`, якщо панель звертається до власного API за абсолютними URL.

Приклад фрагмента `appsettings.json` (поля можуть відрізнятися за назвою

в конкретному репозиторії; наведено узагальнено):

```
{
  "App": {
    "BaseUrl": "http://localhost:5000"
  },
  "Esp32": {
    "BaseUrl": "http://192.168.1.55",
    "PollingIntervalMs": 5000
  },
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=esp32monitor.db"
  }
}
```

Після збереження конфігурації запускається `dotnet run` у каталозі серверного проєкту. У консолі з'являться повідомлення про прослуховування порту Kestrel, ініціалізацію бази та старт фонові служби.

```
C:\Users\YevheniiHorbachov\git\ESP32Monitor>dotnet run
info: ESP32Monitor.Services.FakeMonitoredServerService[0]
      FakeMonitoredServerService: Scenario=AlwaysHealthy, Name=Uplink health (for ESP32 connectivity check), health endpoint returns 204 when healthy. Intermittent ticks: H=8, O=2-5
info: ESP32Monitor.Services.PollingService[0]
      PollingService: real ESP32 polling. MonitoredServer scenario=AlwaysHealthy drives /api/monitored-service/health only - set ESP32 check URL to your PC for LEDs to match.
info: System.Net.Http.HttpClient.esp32.LogicalHandler[100]
      Start processing HTTP request GET http://192.168.1.100/status
info: System.Net.Http.HttpClient.esp32.ClientHandler[100]
      Sending HTTP request GET http://192.168.1.100/status
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\YevheniiHorbachov\git\ESP32Monitor
warn: ESP32Monitor.Services.Esp32Client[0]
      Failed to reach ESP32
      System.Threading.Tasks.TaskCanceledException: The request was canceled due to the configured HttpClient.Timeout of 5 seconds elapsing.
      ----> System.TimeoutException: A task was canceled.
      ----> System.Threading.Tasks.TaskCanceledException: A task was canceled.
      at System.Threading.Tasks.TaskCompletionSourceWithCancellation`1.WaitWithCancellationAsync(CancellationTok
```

Рисунок 5.6 - Консоль після запуску ASP.NET Core

5.4 Запуск панелі, перевірка інтерфейсу та сценаріїв моніторингу

У браузері відкривається адреса панелі (типово `http://localhost:5000` або порт з launchSettings). На головній сторінці перевіряються: доступність пристрою, Wi-Fi, Internet, SSID, IP-адреса, активний ефект. Окремо

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						63
		№ докум.	Підпис	Дата		

відкривається сторінка керування ефектами: перевіряється застосування декоративних режимів, повернення до автоматичного моніторингу, коректність відображення повідомлень про помилки при недоступності ESP32.

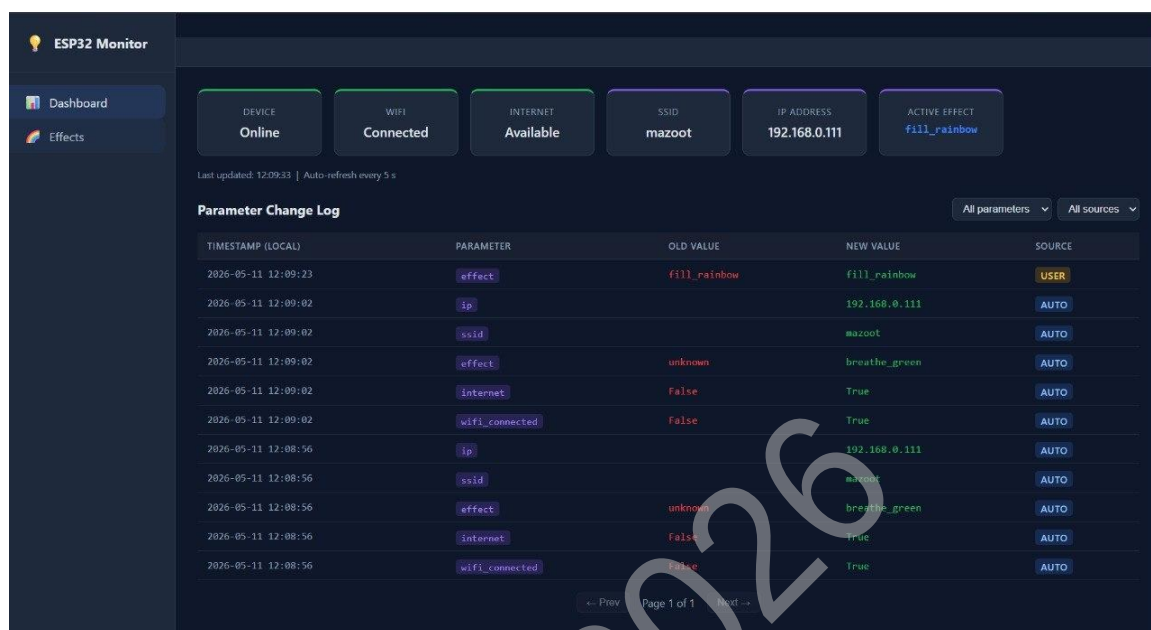


Рисунок 5.7 - Штатний стан панелі

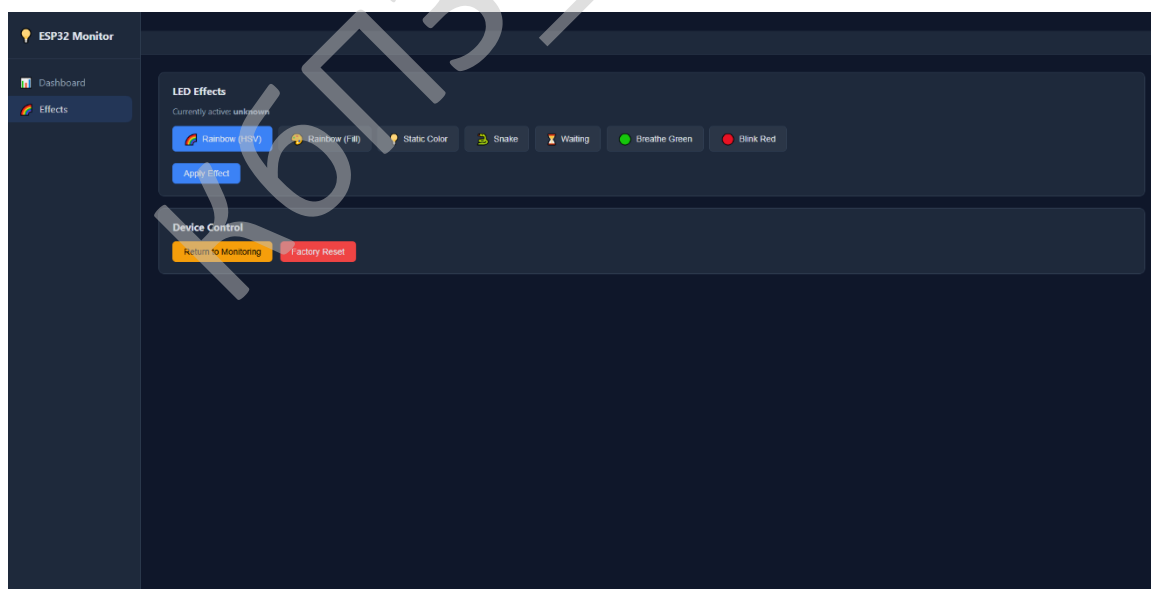


Рисунок 5.8 - Сторінка ефектів

Для експериментальної перевірки Internet доцільно тимчасово відключити WAN на маршрутизаторі, вимкнути роздачу Internet на точці доступу або заблокувати відомий тестовий хост на рівні батьківського контролю - головне, щоб зміна була керованою та оборотною. Очікуваний результат: зміна індикації на стрічці та поява записів у журналі з `Source = auto`.

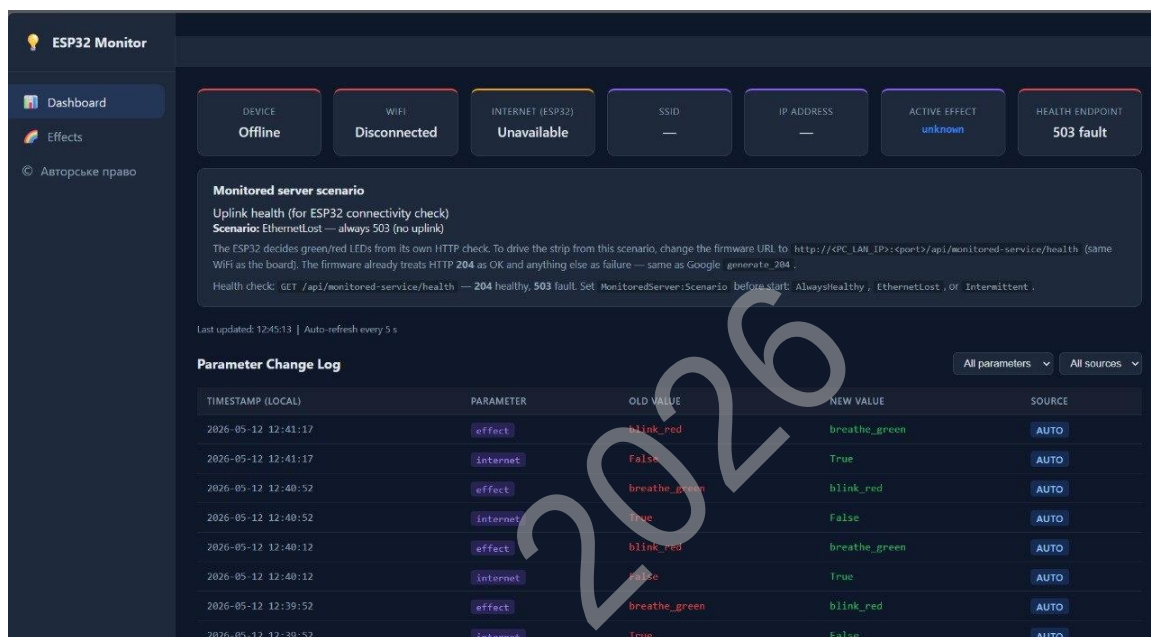


Рисунок 5.9 - Екран ілюструє втрату з'єднання або помилковий стан Internet на панелі

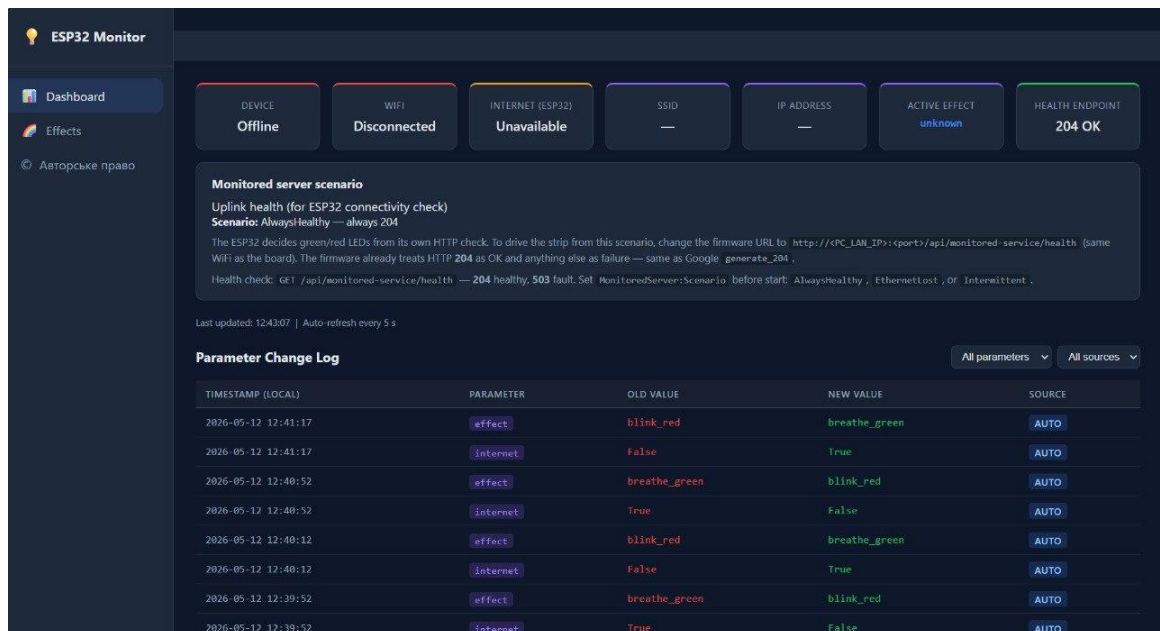


Рисунок 5.10 - Екран пов'язаний із перевіркою доступності зовнішнього ресурсу або «тестового» сервера

5.5 Перевірка журналювання, відтворюваність результатів та типові збої

Перевірка журналювання виконується за сценаріями: зміна ефекту через UI (очікується `user`), перезавантаження ESP32 (можливі зміни `ip`, `wifi\_connected`, `internet` - зазвичай `auto`), повторний запуск `dotnet run` (історія в SQLite зберігається, якщо файл бази не видаляли). У таблиці журналу перевіряють часові мітки, коректність старого та нового значення, відсутність дублікатів при незмінному стані.

Типові проблеми впровадження та шляхи їх усунення:

- Панель не бачить ESP32 - перевірити `Esp32:BaseUrl`, ping з ПК, брандмауер Windows, ізоляцію клієнтів Wi-Fi, чи ESP32 у тій самій підмережі.
- HTTP 404 на маршрутах - переконатися, що ESP32 у Monitoring Mode, а не лише у режимі точки доступу; перезавантажити пристрій після налаштування.

- Мерехтіння стрічки - перевірити спільну землю, якість живлення, довжину лінії DATA, кількість одночасно увімкнених світлодіодів.
- Порожній журнал при очевидних змінах - перевірити, чи запущена фонові служба, чи немає винятків у консолі, чи збігаються імена полів JSON з моделлю.

Після первинної перевірки виконується коротке експлуатаційне тестування: серія з 10-20 циклів «відключити Internet - увімкнути», 5-10 змін ефекту, 3-5 перезапусків ESP32, один цикл Factory Reset. Фіксується час відновлення та відсутність «зависань» панелі. Результати можна оформити таблицею у Word (номер кроку, дія, очікуваний результат, фактичний результат).

Для впровадження на іншому робочому місці доцільно зберегти у додатку до роботи: версію прошивки, версію .NET, хеш коміту репозиторіїв (за наявності), скріншоти ключових екранів, копію 'appsettings.json' без паролів від сторонніх сервісів. Це підвищує відтворюваність експерименту та відповідає вимогам до оформлення розділу з результатами.

Окрему увагу слід приділити підготовці користувача: пояснити різницю між 'auto' та 'user', показати, як фільтрувати журнал за параметром, як не плутати «недоступність Internet» з «недоступністю ESP32».

5.6 Авторське право, склад об'єкта права інтелектуальної власності та правова інформація у застосунку

Після технічного впровадження комплексу доцільно зафіксувати правовий контекст використання програмного забезпечення, зокрема питання авторського права на оригінальний код, права на результати випускної кваліфікаційної роботи та відповідність використаних сторонніх компонентів умовам їхніх ліцензій. У навчальних і демонстраційних проєктах цей блок часто недооцінюють, проте саме він пояснює, хто є носієм прав на текст роботи, на програмну реалізацію та на збірку апаратного макета, а також які обмеження

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						67
		№ докум.	Підпис	Дата		

накладаються на подальше поширення чи комерційне використання результатів.

Авторське право як категорія права інтелектуальної власності охоплює майнові та немайнові права на твори науки, літератури й мистецтва, зокрема на програмний продукт як твір, виражений у формі коду, структури даних, інтерфейсу користувача та супровідної документації. У межах України загальні засади визначаються Законом України «Про авторське право і суміжні права», а також нормами Цивільного кодексу України щодо об'єктів інтелектуальної власності. Для студентської випускної роботи важливо розмежувати: текст пояснювальної записки (науковий твір), програмний код, створений або адаптований автором роботи, та відкриті бібліотеки третіх осіб, які підключаються на умовах відповідних ліцензій і не «заміняють» авторство на власний оригінальний внесок, але вимагають збереження копірайтних повідомлень і умов поширення. Немайнові права включають право авторства, право на ім'я, на недоторканність твору тощо. Майнові права дають змогу використовувати твір у способах, передбачених законом, зокрема відтворювати, розповсюджувати, переробляти. У випускній роботі виконавець зазвичай зберігає немайнові права, а майнові права на використання роботи у навчальному процесі регулюються внутрішніми актами закладу вищої освіти та договором із науковим керівником. Зазначене не скасовує необхідності коректно цитувати джерела коду, ідей архітектури та фрагментів документації з офіційних сайтів Microsoft, Espressif, Adafruit, репозиторіїв GitHub тощо.

Розроблювана система складається з кількох шарів об'єктів, на кожний з яких поширюються різні правові режими. Прошивка для ESP32 може містити власні фрагменти автора роботи та фрагменти, узяті з прикладів і бібліотек з відкритим кодом (Arduino, FastLED). Серверна частина на ASP.NET Core використовує платформу та пакети NuGet, які надаються на умовах ліцензій (зазвичай MIT, Apache 2.0, MS-EULA для runtime). Це означає, що у тексті роботи та у супровідних матеріалах доцільно навести перелік ключових залежностей і коротко зазначити, що їхнє використання відповідає умовам

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						68
		№ докум.	Підпис	Дата		

відповідних ліцензій, а оригінальна частина реалізації (логіка журналювання, інтеграція панелі, сценарії тестування) є результатом самостійної розробки в межах теми кваліфікаційної роботи.

Інтерфейс «Авторське право» у вебзастосунку ESP32 Monitor виконує інформаційну функцію: він дозволяє користувачу ознайомитися з титульними даними випускної роботи без доступу до повного файлу записки. На екрані відображаються найменування міністерства та університету, кафедри, формулювання теми, шифр освітньо-професійної програми та спеціальності, прізвище та ініціали виконавця, прізвище та ініціали наукового керівника, місто і рік захисту. Такий підхід підвищує прозорість демонстраційного стенду: будь-який переглядач бачить, що програмний продукт пов'язаний з конкретною кваліфікаційною роботою, а не є анонімним набором скриптів. З погляду методики оформлення випускної документації, скріншот цього вікна доцільно включити до розділу про впровадження як підтвердження наявності юридично-інформаційного супроводу у користувацькому інтерфейсі.

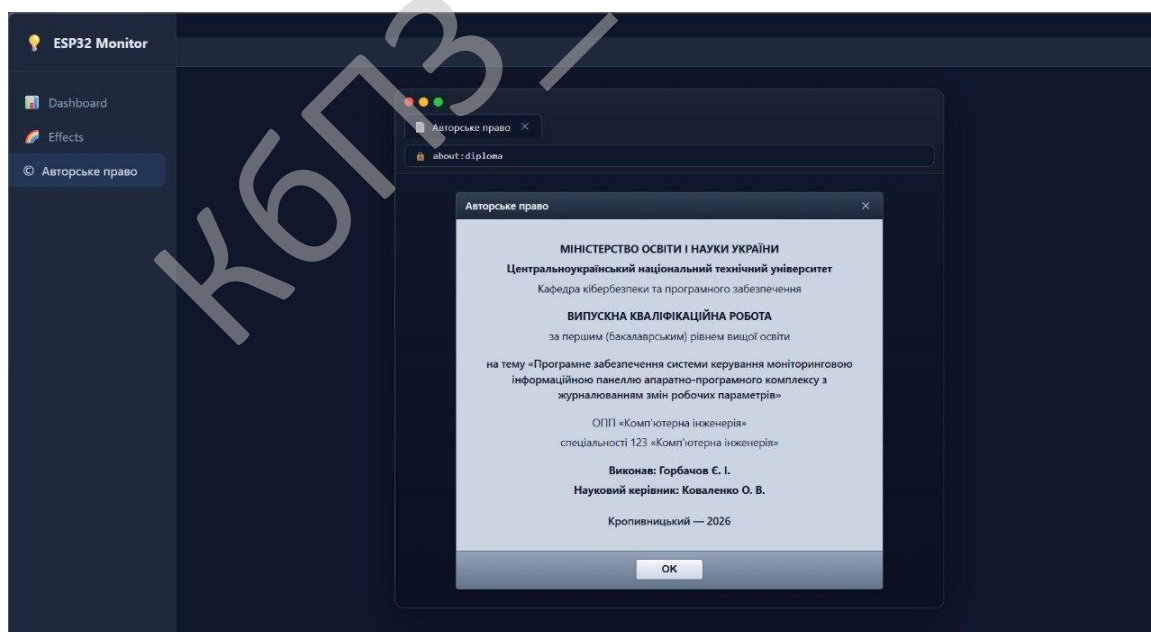


Рисунок 5.11 - Розділ інтерфейсу «Авторське право» з відомостями про випускну кваліфікаційну роботу

					<i>ВКРБ-123.26.0038.00.00.ПЗ</i>	Арк.
		№ докум.	Підпис	Дата		69

Окремо слід зазначити, що відображення персональних даних у навчальному застосунку має відповідати політиці конфіденційності закладу та згоді на обробку даних, якщо система колись вийде за межі локального стенду. У базовому варіанті, коли панель працює лише в локальній мережі, ризики мінімальні, проте формулювання про обмеження відповідальності та про призначення інтерфейсу «лише для навчальних цілей» можна додати в документацію до репозиторію як 'README' або розділ «Юридична інформація».

Практичні рекомендації для супроводу авторських прав у подальшому розвитку проєкту такі: зберігати файл 'LICENSE' у корені кожного репозиторію; не публікувати у відкритому доступі паролі Wi-Fi, токени API та персональні дані третіх осіб; при зміні сторонніх бібліотек перевіряти сумісність ліцензій; при передачі коду іншому розробнику оформлювати передачу прав або використовувати стандартні open-source ліцензії з чіткими зобов'язаннями. Для випускної роботи достатньо показати, що автор усвідомлює склад об'єкта права, розмежовує власний внесок і сторонні компоненти та оформлює відомості про авторство в інтерфейсі й у тексті записки узгоджено.

Титульні дані, наведені на скріншоті, узгоджуються з оформленням титульного аркуша випускної кваліфікаційної роботи: Міністерство освіти і науки України, Центральнoукраїнський національний технічний університет, кафедра кібербезпеки та програмного забезпечення, тема про програмне забезпечення системи керування моніторинговою інформаційною панеллю апаратно-програмного комплексу з журналюванням змін робочих параметрів, освітньо-професійна програма «Комп'ютерна інженерія», спеціальність 123 «Комп'ютерна інженерія», виконав - Горбачов Є. І., науковий керівник - Коваленко О. В., місто Кропивницький, рік 2026. Навіть якщо у фінальній версії титульного аркуша будуть уточнені формулювання, логіка розділу в застосунку залишається незмінною: ідентифікація авторства та зв'язок програмного продукту з офіційною роботою.

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						70
		№ докум.	Підпис	Дата		

Загалом блок про авторське право у розділі впровадження підкреслює, що розробка не обмежується технічними кроками, а включає культуру легального використання програмних засобів і відповідальне представлення результатів. Це узгоджується з вимогами до сучасної інженерної освіти, де компетентності з інтелектуальної власності та етики інформаційних технологій розглядаються нарівні з компетентностями програмування та проектування апаратних комплексів.

У підсумку методика впровадження показує, що система розгортається послідовно й без обов'язкових зовнішніх хмарних сервісів. Для базової роботи потрібні ESP32 із прошивкою, локальна Wi-Fi-мережа, комп'ютер із .NET SDK та налаштований вебзастосунок; експериментальні дані підтверджують коректність проектних рішень через узгодженість стану на панелі, поведінки LED-стрічки та записів у SQLite.

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
		№ докум.	Підпис	Дата		71

6 ОСНОВНІ ВИСНОВКИ

У ході виконання випускної кваліфікаційної роботи було розглянуто, спроектовано та експериментально перевірено програмне забезпечення системи керування моніторинговою інформаційною панеллю апаратно-програмного комплексу з журналюванням змін робочих параметрів. Завдання виявилось типовим для сучасної комп'ютерної інженерії в тому сенсі, що воно одночасно торкається вбудованих систем, комп'ютерних мереж і прикладного програмного забезпечення; водночас воно залишалось достатньо вузьким, щоб на кожному етапі можна було показати не лише «що зроблено», а й чому обрані саме такі рішення. У підсумку сформовано цілісний комплекс на базі мікроконтролера ESP32, адресної LED-стрічки WS2812B, вебзастосунку ASP.NET Core з підсистемою Blazor Server, а також локальної бази даних SQLite, яка виконує роль надійного сховища історії змін.

З теоретичної та прикладної точки зору визначено призначення системи й окреслено область її застосування. Показано, що розроблений комплекс доречний там, де потрібна наочність (швидка візуальна оцінка стану), локальність (відсутність обов'язкової залежності від зовнішніх хмарних сервісів) і відтворюваність (можливість повторити експеримент на стандартному робочому місці). Такий набір вимог характерний для навчальних стендів, лабораторних макетів і невеликих демонстраційних IoT-рішень, де важливо не «перемогти масштаб», а прозоро поєднати апаратну індикацію, мережевий моніторинг і збереження подій.

Виконано огляд існуючих IoT-платформ і засобів моніторингу. Зіставлення показало: індустріальні та універсальні платформи часто дають надлишок функцій для сценарію «один пристрій - одна панель - одна історія змін». Тому обґрунтовано доцільність автономної інформаційної панелі з прямою HTTP-взаємодією з ESP32: такий підхід спрощує аналіз архітектури,

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
		№ докум.	Підпис	Дата		72

зменшує кількість проміжних компонентів і робить поведінку системи зрозумілою як для користувача, так і для рецензента роботи. Паралельно це узгоджується з класичною інженерною логікою: спочатку мінімально життєздатний продукт із чіткими межами, а вже потім розширення за потреби.

Обґрунтовано вибір апаратних і програмних засобів. ESP32 обрано завдяки поєднанню обчислювальних можливостей, Wi-Fi-інтерфейсу та зрілої екосистеми інструментів розробки; WS2812B використано як інженерно виправданий засіб кодування стану в кольорі та динаміці світла. Програмний стек ASP.NET Core, Blazor Server, Entity Framework Core і SQLite дозволив реалізувати сучасний вебінтерфейс, серверну логіку, фонове опитування пристрою та структуроване журналювання без зайвої складності розгортання. Важливо підкреслити: вибір технологій тут не є самоціллю; він підпорядкований задачі спостережуваності (що відбувається з пристроєм зараз) і аудиту (що змінилося згодом і з чієї ініціативи).

У роботі систематизовано опис функціонування комплексу, наведено структурну, функціональну та принципову схеми, розкрито сценарії експлуатації: первинне налаштування через Captive Portal, перехід у режим моніторингу, перевірку доступності Internet, керування декоративними LED-ефектами та накопичення записів журналу. Практична цінність такого опису полягає в тому, що він з'єднує «рівень прошивки» і «рівень застосунку» в єдину картину: зміни на панелі не виглядають випадковими, а набувають пояснення через узгодженість з мережевим станом, алгоритмами опитування та правилами запису в базу.

Поставлену мету досягнуто. Сформовано програмно-апаратне рішення, яке забезпечує моніторинг стану ESP32, керування LED-індикацією та журналювання змін робочих параметрів у локальній базі даних. При цьому результат виглядає не як набір розрізнених фрагментів коду, а як система з визначеними інтерфейсами взаємодії, зрозумілими режимами роботи та можливістю повторити ключові сценарії тестування.

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						73
		№ докум.	Підпис	Дата		

Розроблена система має прикладне значення: її можна використовувати як навчальний полігон для дисциплін, пов'язаних із вбудованими системами та вебтехнологіями, а також як прототип для подальшого ускладнення (мультипристрої, політики безпеки, аналітика часових рядів). Архітектура залишається достатньо простою для критичного аналізу, але водночас відображає структуру сучасного апаратно-програмного комплексу: джерело даних, канал передачі, серверна обробка, інтерфейс користувача та сховище історії. Саме ця «повнота в мініатюрі» і є одним із головних інженерних висновків роботи: повноцінність системи визначається не кількістю модулів, а узгодженістю їхніх ролей.

Разом із досягненнями слід чесно зафіксувати й межі поточної реалізації: масштабування на велику кількість пристроїв, жорсткі вимоги до безпеки та сценарії з гарантованою доставкою подій у реальному часі потребуватимуть додаткових механізмів (ідентифікація користувачів, розподілені протоколи, резервування, політики оновлення прошивки). Це не суперечить виконаній роботі, а природно виводить її в площину наступних науково-технічних задач.

Подальший розвиток доцільно планувати еволюційно: додати автентифікацію та розмежування прав, розширити набір моніторингових параметрів, підтримати кілька ESP32-пристроїв у межах однієї панелі, побудувати графіки та агрегати за журналом, реалізувати експорт подій у файли стандартних форматів. Для сценаріїв, де критичні швидкість оновлення та «пуш-модель» подій, логічно розглянути MQTT або WebSocket як доповнення до HTTP, не відкидаючи вже напрацьованої серверної логіки.

Завершуючи, варто підкреслити головне: виконана кваліфікаційна робота демонструє здатність інтегрувати знання з комп'ютерної інженерії в завершений інженерний артефакт, який можна показати, перевірити й розвинути. Навіть якщо окремі технічні деталі з часом зміняться, залишаться незмінними принципи, які ця робота ілюструє: ясність архітектури, дисципліна журналювання, повага до обмежень платформи та прагнення зробити систему

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						74
		№ докум.	Підпис	Дата		

зрозумілою людині, а не лише «коректною для компілятора». Отримані результати можуть стати фундаментом для поглибленої магістерської підготовки або для практичних проєктів у сфері IoT-моніторингу, а модульність прийнятих рішень свідчить про їхню життєздатність у довгостроковій перспективі.

К6ПЗ_2026

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						75
		№ докум.	Підпис	Дата		

СПИСОК ЛІТЕРАТУРИ

1. Методичні рекомендації до виконання й захисту випускної кваліфікаційної роботи здобувачами вищої освіти на першому (бакалаврському) рівні вищої освіти для студентів спеціальності 123 «Комп'ютерна інженерія». Центральноукраїнський національний технічний університет. Кропивницький, 2025.

2. ESP32 LED Monitoring. GitHub repository. URL: https://github.com/mikroffarad/ESP32_LED_Monitoring.

3. ESP32Monitor. GitHub repository. URL: <https://github.com/zhGorbachov/ESP32Monitor>.

4. Espressif Systems. ESP32-DevKitC Documentation. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html>.

5. Espressif Systems. Arduino ESP32 Documentation. URL: <https://docs.espressif.com/projects/arduino-esp32/en/latest/>.

6. FastLED Documentation. URL: <https://fastled.io/docs/>.

7. Microsoft. ASP.NET Core Blazor documentation. URL: <https://learn.microsoft.com/en-us/aspnet/core/blazor/>.

8. Microsoft. Entity Framework Core documentation. URL: <https://learn.microsoft.com/en-us/ef/core/>.

9. RFC 9110. HTTP Semantics. Internet Engineering Task Force. URL: <https://www.rfc-editor.org/rfc/rfc9110>.

10. SQLite Documentation. URL: <https://www.sqlite.org/docs.html>.

11. PlatformIO Documentation. URL: <https://docs.platformio.org/>.

12. Espressif Systems. ESP32 Series Datasheet. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

13. Espressif Systems. ESP-IDF Programming Guide. URL:

					ВКРБ-123.26.0038.00.00.ПЗ	Арк.
						76
		№ докум.	Підпис	Дата		

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>.

14. Espressif Systems. Wi-Fi Driver Documentation. URL: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html>.

15. Arduino. Arduino Reference. URL: <https://www.arduino.cc/reference/en/>.

16. PlatformIO. PlatformIO Core Documentation. URL: <https://docs.platformio.org/en/latest/core/index.html>.

17. Microsoft. ASP.NET Core fundamentals overview. URL: <https://learn.microsoft.com/en-us/aspnet/core/fundamentals/>.

18. Microsoft. ASP.NET Core SignalR documentation. URL: <https://learn.microsoft.com/en-us/aspnet/core/signalr/>.

19. Microsoft. Entity Framework Core SQLite Provider. URL: <https://learn.microsoft.com/en-us/ef/core/providers/sqlite/>.

20. Microsoft. .NET documentation. URL: <https://learn.microsoft.com/en-us/dotnet/>.

21. ISO/IEC 25010:2011. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. International Organization for Standardization, 2011.

22. ISO/IEC/IEEE 42010:2022. Software, systems and enterprise - Architecture description. International Organization for Standardization, 2022.

23. Adafruit. NeoPixel Uberguide. URL: <https://learn.adafruit.com/adafruit-neopixel-uberguide>.

24. Worldsemi. WS2812B Intelligent Control LED Integrated Light Source Datasheet. URL: <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf>.

25. Tanenbaum A.S., Austin T. Structured Computer Organization. 6th Edition. Pearson, 2012.

26. Patterson D.A., Hennessy J.L. Computer Organization and Design: The Hardware/Software Interface. 5th Edition. Morgan Kaufmann, 2013.

27. Vahid F., Lysecky S. Embedded Systems Design: A Unified Hardware/Software Introduction. 3rd Edition. Wiley, 2017.

					БКРБ-123.26.0038.00.00.ПЗ	Арк.
						77
		№ докум.	Підпис	Дата		

28. Harwani B.M. Beginning Arduino Programming. Apress, 2010.
29. Tolentino M. ESP32 Development using the Arduino IDE. Elektor, 2020.
30. Kurniawan A. Internet of Things Projects with ESP32. Packt Publishing, 2019.
31. Foundations of Software Engineering, 2nd Edition. Sommerville I. Pearson, 2015.
32. IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications.
33. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.
34. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.
35. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns. Addison-Wesley, 1994.
36. Bolton W. Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering. 8th Edition. Pearson, 2022.
37. Ferreira A. ESP32 Programming for the Internet of Things. GitHub Repository, 2023.
38. Segaran T. Programming Collective Intelligence. O'Reilly, 2007.
39. Albahari J., Albahari B. C# 12.0 in a Nutshell: The Definitive Reference. O'Reilly, 2023.
40. Liberty J. Programming C# 8.0. O'Reilly, 2019.
41. Hanselman S., Freeman A. Pro ASP.NET Core MVC 2. Apress, 2017.
42. Troelsen A., Japikse P. Pro C# 8 with .NET Core 3. Apress, 2020.
43. Richter J. CLR via C#: Applied .NET Framework Programming. Microsoft Press, 2012.
44. Klein J. Internet of Things with ESP32. Leanpub, 2019.
45. Ammann P., Offutt J. Introduction to Software Testing. Cambridge University Press, 2016.

46. The MQTT Version 3.1.1 Protocol Specification. OASIS, 2014. URL: <https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>
47. The WebSocket Protocol. RFC 6455. IETF, 2011. URL: <https://datatracker.ietf.org/doc/html/rfc6455>
48. ISO/IEC/IEEE 12207:2017. Systems and software engineering – Software life cycle processes.
49. Knuth D.E. The Art of Computer Programming. Addison-Wesley, 1997.
50. Mazidi M.A., Naimi S., Naimi S. The AVR Microcontroller and Embedded Systems Using Assembly and C. Pearson, 2011.
51. Martin R.C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017.
52. Fielding R.T. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.
53. Khanna V.K. Microprocessor and Microcontroller Systems. 2nd Edition. New Age International, 2015.
54. Moroney L. Programming the Internet of Things: An Introduction to Building Integrated, Device-to-Cloud IoT Solutions. O'Reilly, 2021.
55. Marz N., Warren J. Big Data: Principles and Best Practices of Scalable Real-Time Data Systems. Manning, 2015.
56. ISO/IEC 27001:2022. Information technology — Security techniques — Information security management systems — Requirements.