МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПОРТАТИВНОЇ СИСТЕМИ ЗБОРУ ТА ОБРОБКИ ДАНИХ СЕНСОРІВ

Виконав:
студент групи 2К-22
спеціальності
123 Комп’ютерна інженерія
Дмитро ЄВТУШЕНКО
Керівник:
Наталя ФАЛЬЧЕНКО

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Циклова комісія комп'ютерної інженерії та інформаційних технологій

Спеціальність 123 «Комп'ютерна інженерія»

Освітня програма 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувачка ЦК КІ та ІТ

_____ Наталя ФАЛЬЧЕНКО

(підпис)

«_____»_____2025 р.

З А В Д А Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Євтушенко Дмитру Віталійовичу

1. Тема кваліфікаційної роботи: Проєктування та реалізація портативної системи збору та обробки даних сенсорів

Керівник роботи: Фальченко Наталя Григорівна

затверджені наказом закладу фахової передвищої освіти від 06.10.2025 року № 84/1-у

2. Строк подання студентом кваліфікаційної роботи: 08.06.2026 р.

3. Вихідні дані до кваліфікаційної роботи: Відомості про архітектуру портативної системи збору даних; способи використання мікроконтролерних платформ та сенсорів для зчитування фізичних величин; методи реалізації алгоритмів обробки даних: аналіз даних сенсорів; способи використання цифрових інтерфейсів та бездротових протоколів для передачі інформації

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити): Аналіз сучасних засобів проєктування систем моніторингу та збору даних; обґрунтування вибору апаратної платформи та сенсорного забезпечення; розробка структурної та електричної принципової схем пристрою; створення вбудованого програмного забезпечення для мікроконтролера; реалізація алгоритмів первинної обробки та візуалізації даних; оцінка автономності та тестування системи.

5. Дата видачі завдання 15.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання з підписами керівника і студента
1	Вступ	14.10.2025	
2	Розділ 1 Аналіз принципів побудови систем збору даних	09.12.2025	
3	Розділ 2 Проєктування апаратної та програмної частини системи	10.03.2026	
4	Розділ 3 Реалізація, налагодження та тестування системи	28.04.2026	
5	Висновки	12.05.2026	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	26.05.2026	
7	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	02.06.2026	
8	Подання кваліфікаційної роботи на затвердження завідувачу ЦК (за 7 днів до захисту)	08.06.2026	

Студент _____ Дмитро ЄВТУШЕНКО

(підпис)

Керівник роботи _____ Наталя ФАЛЬЧЕНКО

(підпис)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці портативної системи збору та обробки даних сенсорів, яка функціонує на базі сучасних мікроконтролерних рішень. Метою роботи є створення автономного апаратно-програмного комплексу для оперативного моніторингу фізичних величин із використанням енергоефективних технологій. У роботі обґрунтовано вибір компонентної бази та підходів до реалізації системи, зосереджено увагу на точності зчитування сигналів, обробці даних у реальному часі, а також забезпеченні тривалої автономної роботи пристрою.

Робота містить 40 сторінок, 3 таблиці, 5 рисунків, 1 додаток та 15 джерел за переліком посилань.

У процесі проєктування створено модульну архітектуру системи, що включає основні функціональні блоки: модуль зчитування даних із сенсорів, блок первинної цифрової обробки та інтерфейс передачі інформації. Процес взаємодії з датчиками реалізовано через цифрові інтерфейси I2C та SPI. Для візуалізації результатів та налаштування параметрів системи розроблено відповідне програмне забезпечення.

Проведено тестування роботи портативної системи в типових сценаріях експлуатації, виявлено та усунено похибки вимірювань, а також здійснено оптимізацію енергоспоживання мікроконтролера у режимах очікування. Визначено ключові напрямки для подальшого вдосконалення пристрою, серед яких - інтеграція з хмарними сервісами. В цілому в кваліфікаційній роботі виконано поставлені завдання.

Ключові слова: Система збору даних, мікроконтролер, сенсор, портативний пристрій, цифрова обробка сигналів, енергоефективність.

ABSTRACT

This qualification paper presents the development of a portable system for collecting and processing sensor data based on modern microcontroller solutions. The aim of the project is to create an autonomous hardware and software complex for real-time monitoring of physical quantities using energy-efficient technologies. The paper substantiates the choice of components and implementation methods, focusing on signal acquisition accuracy, real-time data processing, and ensuring long-term autonomous operation.

The paper consists of 40 pages, 3 tables, 5 figures, 1 appendices, and 15 references.

During the design phase, a modular system architecture was implemented, covering the main functional units: a sensor data acquisition module, a primary digital processing unit, and a data transmission interface. The sensor interaction process was handled through I2C and SPI digital interfaces. Software was developed for data visualization and system parameter configuration.

The portable system functionality was tested in typical operating scenarios, with measurement errors identified and fixed, and microcontroller power consumption optimized for standby modes. The paper outlines key directions for future development, including integration with cloud services. Overall, the tasks set in the qualification work have been completed.

Keywords: Data acquisition system, microcontroller, sensor, portable device, digital signal processing, enegy efficiency.

ЗМІСТ

ВСТУП

Актуальність теми полягає у безперервному розширенні сфер застосування автоматизації та цифровізації в сучасному світі, що зумовлює критично високий попит на системи дистанційного моніторингу та збору даних. Стрімкий розвиток мікроелектроніки, інформаційних технологій та глобальної концепції Інтернету речей докорінно змінив підходи до вимірювання, реєстрації та аналізу фізичних величин. Сучасний етап розвитку комп'ютерної інженерії характеризується вираженим переходом від громіздких, стаціонарних і дорогих вимірювальних комплексів до компактних, енергоефективних та повністю автономних пристроїв, що здатні працювати в режимі периферійних обчислень.

Створення портативних пристроїв, здатних автономно зчитувати, обробляти та передавати інформацію з різноманітних сенсорів, є одним із пріоритетних напрямів комп’ютерної інженерії. Такі системи знаходять застосування в екологічному моніторингу, промисловій автоматизації, “розумних” будинках та медицині. Зокрема, контроль параметрів навколишнього середовища, таких як рівень штучного та природного освітлення, є критично важливим для забезпечення ергономіки робочих місць, автоматизації агропромислових комплексів (точного землеробства та теплиць), а також для оптимізації енергоспоживання у муніципальних системах освітлення концепції “Розумного міста”.

Традиційні системи збору даних часто стикаються з проблемами високої вартості розгортання, прив'язки до провідної інфраструктури живлення та значного навантаження на канали зв'язку через трансляцію великих масивів невідфільтрованих “сирих” даних. Розробка енергоефективного та компактного рішення на базі сучасних мікроконтролерів дозволяє мінімізувати витрати на розгортання вимірювальних мереж та забезпечує високу мобільність отримання даних у реальному часі. Використання сучасних систем на кристалі (SoC), таких як ESP32, відкриває нові можливості

для створення інтелектуальних автономних вузлів. Наявність інтегрованих бездротових інтерфейсів, низька вартість та архітектурна гнучкість роблять ці платформи промисловим стандартом розробки портативних систем моніторингу.

Проте розробка таких пристроїв вимагає вирішення ряду апаратно-програмних суперечностей, пов'язаних з обмеженістю апаратних ресурсів мобільних платформ, необхідністю мінімізації енергоспоживання при активному використанні радіомодулів, а також забезпеченням високої завадостійкості та стабільності зчитування даних у реальному часі. Все це зумовлює високу актуальність теми кваліфікаційної роботи, яка спрямована на пошук оптимальних інженерних та алгоритмічних рішень для побудови надійного і доступного портативного пристрою з віддаленим керуванням.

Об’єктом дослідження є система збору та обробки інформації з датчиків фізичних величин за допомогою мікроконтролерних систем.

Предметом дослідження є способи розробки архітектури ПЗ, алгоритми первинної обробки сигналів, протоколи передачі даних та методи оптимізації енергоспоживання портативної системи.

Метою роботи є проєктування та практична реалізація портативної системи на базі мікроконтролера, призначеної для збору даних з групи сенсорів, їх локальної обробки та візуалізації результатів.

Практичне значення отриманих результатів роботи полягає у створенні та тестуванні діючого, апаратно та програмно оптимізованого прототипу портативної системи моніторингу рівня освітленості. Апробація результатів кваліфікаційної роботи здійснювалася шляхом їх представлення на XVIII Всеукраїнській науково-практичній конференції «Тенденції розвитку ІТ-технологій в Україні», 23-24 квітня, м. Черкаси. Тема доповіді: "Інженерія портативних засобів вимірювання та опрацювання даних".

РОЗДІЛ 1 АНАЛІЗ ПРИНЦИПІВ ПОБУДОВИ СИСТЕМ ЗБОРУ ДАНИХ

1.1 Призначення та сфера застосування портативних систем моніторингу

Стрімкий розвиток мікроелектроніки, інформаційних технологій та концепції Інтернету речей докорінно змінив підходи до вимірювання фізичних величин. Сучасний етап комп’ютерної інженерії характеризується переходом від громіздких, стаціонарних вимірювальних комплексів до компактних, енергоефективних та повністю автономних пристроїв. Портативні системи збору даних (Data Acquisition Systems, DAQ) являють собою інтегровані апаратно-програмні комплекси, які забезпечують автоматизоване зчитування сигналів із сенсорів, їх первинне цифрове перетворення, локальну обробку та передачу до централізованих баз даних або хмарних сховищ [1].

Головним призначенням таких систем є забезпечення безперервного або періодичного моніторингу параметрів навколишнього середовища, технологічних процесів чи стану об'єктів у режимі реального часу, незалежно від наявності стаціонарної інфраструктури живлення та дротових ліній зв'язку. Упровадження портативних рішень дозволяє вирішити ключову проблему традиційних вимірювальних мереж - високу вартість розгортання та обслуговування ліній передачі даних на великих або важкодоступних територіях.

Сфери застосування портативних систем моніторингу надзвичайно широкі і продовжують масштабуватися завдяки здешевленню мікроконтролерних платформ (див. рис. 1.1). Основними напрямками на сьогодні є:

1. Промисловий моніторинг на сучасному виробництві характеризується впровадженням концепції предиктивного обслуговування обладнання. Замість планових зупинок для огляду техніки, мініатюрні модулі, оснащені МЕМС-акселерометрами та датчиками температури, кріпляться

безпосередньо на рухомі частини верстатів, зокрема на підшипники або двигуни. Вони безперервно збирають дані про рівень вібрації та теплові показники, що дозволяє виявляти аномалії на ранніх етапах, запобігати критичним поломкам і суттєво знижувати фінансові втрати підприємства.

2. Екологічний моніторинг та концепція розумних міст обумовлені необхідністю щільного контролю якості навколишнього середовища в умовах глобальних кліматичних змін та урбанізації. Портативні станції моніторингу якості повітря (вимірювання рівня CO_2, SO_2, NO_2, а також дрібнодисперсного пилу PM2.5 та PM10) можуть бути розміщені на громадському транспорті, дронах або просто кріпитися до стовпів вуличного освітлення. Завдяки автономному живленню такі системи утворюють розгалужену бездротову сенсорну мережу (Wireless Sensor Network, WSN), що збирає масиви геопросторових даних для аналізу екологічної ситуації мікрорайонів у реальному часі.

3. Розумне сільське господарство базується на використанні портативних систем збору даних як основи точного землеробства.. Пристрої, оснащені датчиками вологості ґрунту, рівня освітленості, температури та кислотності (pH), розміщуються на полях або в теплицях. Зібрані дані дозволяють автоматизувати системи крапельного зрошення та дозування добрив, що не лише підвищує врожайність, але й значно економить водні ресурси. У цьому секторі особливо цінується портативність, оскільки системи мають працювати автономно протягом усього аграрного сезону.

4. Біомедична інженерія отримала новий поштовх до розвитку завдяки мініатюризації електронних компонентів, що дозволило створити ефективні натільні системи моніторингу здоров'я пацієнтів. Портативні пристрої збирають дані про частоту серцевих скорочень (ЕКГ, фотоплетизмографія), рівень сатурації кисню в крові (SpO_2) та температуру тіла.

Дані передаються на смартфон лікаря або пацієнта через протоколи BLE (Bluetooth Low Energy), що дозволяє здійснювати віддалений нагляд за хворими в період реабілітації.

Рисунок 1.1 - Класифікація сфер застосування DAQ

Проєктування таких систем висуває перед розробниками жорсткі інженерні вимоги (див. табл. 1.1), які відрізняються від вимог до стаціонарних комп'ютерів. Ключовими серед них є:

Вимоги щодо енергоефективності та автономності зумовлені тим, що портативний пристрій живиться від акумуляторної батареї (найчастіше Li-Ion або Li-Po), тому його апаратна архітектура та програмне забезпечення повинні бути оптимізовані для мінімізації енергоспоживання. Це досягається шляхом використання режимів глибокого сну мікроконтролера, з яких система виходить лише на кілька мілісекунд для зчитування даних із сенсорів і їх відправки, після чого знову переходить у стан енергозбереження.

Необхідність локальної обробки даних полягає у застосуванні підходу, за якого первинна обробка та фільтрація даних (наприклад, видалення шумів, обчислення середнього значення) відбувається безпосередньо на борту мікроконтролера. Це дозволяє суттєво зменшити обсяг інформації, що передається через бездротовий канал зв'язку, тим самим економлячи ресурс батареї та зменшуючи навантаження на мережу.

Критерії захищеності та надійності вимагають особливої уваги, оскільки портативні системи часто експлуатуються в агресивних умовах (підвищена вологість, пил, перепади температур), конструкція пристрою повинна відповідати міжнародним стандартам захисту (наприклад, IP65 або IP67). Електронна база повинна витримувати індустріальні температурні діапазони (від -40°C до +85°C).

Забезпечення модульності та масштабованості передбачає, що апаратна структура повинна підтримувати підключення різних типів датчиків через уніфіковані цифрові шини (I2C, SPI, UART). Це робить розроблену систему універсальною платформою, яку можна швидко адаптувати під нові задачі моніторингу шляхом простої заміни сенсорного модуля.

Таблиця 1.1 - Основні технічні та експлуатаційні вимоги до портативних систем моніторингу

Характеристика	Технічна вимога	Обґрунтування
Енергоспоживання	Струм у режимі сну (Deep Sleep) < 50 мкА; підтримка Li-Ion/Li-Po акумуляторів (3.7 В)	Забезпечення тривалої автономної роботи (від кількох тижнів до місяців) без необхідності підзарядки у польових умовах.
Масо-габаритні показники	Компактність (орієнтовно до 100x100x50 мм), мала вага (до 300 г)	Зручність транспортування, можливість прихованого монтажу або кріплення на рухомих частинах обладнання (для вібродіагностики).
Точність зчитування	Роздільна здатність АЦП не менше 12 біт; підтримка цифрових шин I2C/SPI	Мінімізація похибок при зчитуванні фізичних величин та висока завадостійкість під час передачі сигналів від сенсора до мікроконтролера.
Ступінь захисту (IP)	Клас захисту корпусу IP65 або вище; робочий температурний діапазон від -20°C до +60°C	Стійкість до впливу вологи, пилу, агресивних середовищ та перепадів температур при зовнішній експлуатації.
Комунікаційні можливості	Підтримка бездротових протоколів передачі (Wi-Fi, Bluetooth Low Energy або LoRaWAN)	Можливість дистанційної передачі зібраних масивів даних на хмарний сервер або мобільний пристрій для подальшого аналізу.
Обчислювальна потужність	Наявність апаратних таймерів, підтримка переривань, частота ядра від 16 МГц	Забезпечення можливості локальної цифрової обробки сигналів (Edge Computing) та використання алгоритмів фільтрації шумів.

В результаті розробка сучасної портативної системи збору та обробки даних є комплексною інженерною задачею. Вона вимагає глибокого розуміння мікроконтролерних архітектур, принципів цифрової обробки сигналів та методів оптимізації енергоспоживання. Створення такого пристрою відкриває широкі перспективи для автоматизації рутинних вимірювальних процесів і є високоактуальним напрямком у сучасній інженерії.

1.2 Огляд сучасних сенсорів та методів первинної обробки сигналів

У структурі будь-якої портативної системи збору даних (DAQ) сенсорний рівень відіграє роль первинної ланки, яка відповідає за взаємодію обчислювального ядра з фізичним світом. Ефективність, точність та енергоспоживання всієї системи безпосередньо залежать від характеристик обраних перетворювачів. Еволюція мікроелектроніки протягом останнього десятиліття призвела до парадигмального зсуву: класичні аналогові датчики, що вимагали складних зовнішніх кіл обв’язки (операційних підсилювачів, фільтрів низьких частот), масово витісняються високоінтегрованими цифровими сенсорами.

Сучасний цифровий інтелектуальний сенсор є складною мікросхемою, на кристалі якої, окрім самого чутливого елемента, розміщено прецизійний аналогово-цифровий перетворювач, блок температурної компенсації, енергонезалежну пам'ять із заводськими калібрувальними коефіцієнтами та контролер цифрової шини. Такий підхід радикально знижує вплив електромагнітних завад на етапі передачі сигналу від сенсора до мікроконтролера, оскільки передається вже оцифрований пакет даних, захищений контрольними сумами.

За фізичним принципом дії та технологією виготовлення сенсори, що застосовуються в портативних системах моніторингу, класифікують за кількома ключовими напрямами.

Однією з провідних технологій є МЕМС-сенсори (Micro-Electro-Mechanical Systems). Вона дозволяє створювати мікроскопічні механічні структури (балки, пружини, мембрани) безпосередньо на кремнієвій підкладці разом з електронною логікою. МЕМС-технологія виступає стандартом для розробки сучасних цифрових акселерометрів, гіроскопів (наприклад, MPU6050) та барометрів. Зміна фізичного стану, зокрема прискорення, викликає зміну ємності між рухомими та нерухомими гребінчастими структурами датчика, яка миттєво оцифровується. Завдяки цій технології сенсори вирізняються міліметровими розмірами та мікроамперним струмом споживання, що є ідеальним рішенням для автономних пристроїв.

Іншу важливу групу становлять напівпровідникові сенсори навколишнього середовища, які охоплюють комплексні датчики температури, вологості та тиску. Яскравими представниками цього типу є модуль BME280 від Bosch Sensortec або серія SHT3x від Sensirion. Вони забезпечують високу лінійність вимірювань і дозволяють зчитувати відразу кілька фізичних параметрів через єдиний цифровий інтерфейс (див. табл. 1.2) [4].

Таблиця 1.2 - Порівняльна характеристика сучасних цифрових сенсорів навколишнього середовища

Модель сенсора	Виробник	Інтерфейс зв'язку	Робоча напруга, В	Точність вимірювання (Темп. / Вологість)	Струм споживання (Вимірювання / Сон)	Додаткові можливості
BME280	Bosch Sensortec	I2C, SPI	1.71 - 3.6	+-1.0 °C / +-3.0 %	3.6 мкА (при 1 Гц) / 0.1 мкА	Вимірювання атмосферного тиску
SHT31	Sensirion	I2C	2.15 - 5.5	+-0.2 °C / +-2.0 %	800 мкА / 0.2 мкА	Висока надійність, апаратне попередження
DHT22 (AM2302)	Aosong Electronics	1-Wire	3.3 - 6.0	+-0.5 °C / +-2.0 - 5.0 %	1.5 мА / 40 мкА	Бюджетне рішення, низька частота опитування

При проєктуванні апаратної частини портативних пристроїв для забезпечення стабільного обміну інформацією між цифровими сенсорами та центральним мікропроцесором зазвичай обирають один із двох найбільш поширених протоколів послідовної передачі даних.

Першим з них є шина I2C (Inter-Integrated Circuit), яка стала де-факто стандартом для портативної електроніки. Її головна перевага полягає у використанні лише двох ліній: даних (SDA) та тактування (SCL), незалежно від кількості підключених пристроїв. Кожен сенсор на такій шині має власну 7- або 10-бітну апаратну адресу. Оскільки виходи пристроїв мають конфігурацію “відкритий колектор”, шина потребує встановлення підтягуючих резисторів. Завдяки економії портів мікроконтролера, інтерфейс I2C ідеально підходить для підключення відносно повільних датчиків температури, вологості чи тиску [6].

Альтернативним рішенням є шина SPI (Serial Peripheral Interface), що працює за схемою одночасної двосторонньої передачі даних з використанням чотирьох ліній (MISO, MOSI, SCK, CS). Хоча SPI вимагає більшої кількості контактів мікроконтролера, оскільки для кожного нового сенсора потрібна окрема лінія вибору пристрою (Chip Select), цей протокол забезпечує значно вищу швидкість передачі даних - до десятків мегабіт за секунду. Його доцільно використовувати для взаємодії з високочастотними MEMC-акселерометрами або модулями збереження даних, такими як SD-карти.

Незважаючи на високу якість сучасних цифрових сенсорів, отримані “сирі” дані завжди містять певний рівень апаратного шуму. Крім того, у промислових або польових умовах на роботу системи активно впливають електромагнітні наведення від силових кабелів чи радіопередавачів. З огляду на це, обов'язковим етапом стає первинна цифрова обробка сигналів (Digital Signal Processing - DSP) безпосередньо на борту мікроконтролера.

Замість того, щоб відправляти всі зчитані дані на віддалений сервер, мікроконтролер застосовує математичні алгоритми фільтрації. Це дозволяє суттєво зменшити обсяг мережевого трафіку та підвищити достовірність інформації, що передається кінцевому користувачу.

В інженерії для портативних систем найчастіше застосовують два базові алгоритми [5]:

– Фільтр ковзного середнього (Moving Average Filter).

Це різновид цифрового фільтра низьких частот (ФНЧ). Його суть полягає у створенні вікна заданого розміру N, у якому обчислюється середнє арифметичне значення останніх вибірок. При отриманні нового значення найстаріше відкидається. Цей алгоритм ефективно згладжує високочастотний "білий шум" (наприклад, тепловий шум АЦП).

Математичний розрахунок згладженого показника для кожної поточної вибірки наведено у формулі (1.1):

$$y[n] = \frac{1}{N}\sum_{i=0}^{N-1} x[n-i], \qquad (1.1)$$

де $y[n]$ - *відфільтроване значення;* $x[n]$ - *вхідне зчитане значення;*

N - *розмір вікна фільтрації.*

– Медіанний фільтр (Median Filter).

На відміну від ковзного середнього, медіанний фільтр є нелінійним. Він сортує масив з N останніх вибірок за зростанням і обирає значення, що знаходиться рівно посередині відсортованого списку. Цей метод є надзвичайно ефективним для боротьби з імпульсними завадами (так званий шум "сіль і перець" або випадкові "викиди"), які виникають при поганому контакті або короткочасних електромагнітних імпульсах.

Якщо сенсор температури показує значення [22, 22, 85, 23, 22], ковзне середнє суттєво викривить результат через аномалію "85", тоді як медіанний фільтр просто відкине її, повернувши коректне значення "22".

Впровадження алгоритмів медіанної фільтрації та ковзного середнього у вбудоване програмне забезпечення портативної системи дозволяє досягти високої стабільності показників. Крім того, обчислювальна складність цих методів є відносно низькою, що дозволяє реалізовувати їх навіть на мікроконтролерах з малою тактовою частотою без суттєвого впливу на енергоспоживання пристрою.

1.3 Порівняльний аналіз мікроконтролерних платформ для реалізації системи

Обчислювальне ядро (мікроконтролер або система на кристалі - SoC) є центральним вузлом будь-якої портативної системи збору даних. Воно визначає не лише продуктивність та швидкість цифрової обробки сигналів, але й загальний енергетичний бюджет пристрою, його габарити та комунікаційні можливості. У сучасній практиці проєктування вбудованих систем для реалізації портативних рішень найчастіше розглядаються платформи на базі архітектур ARM Cortex-M та Xtensa. Для обґрунтування апаратної бази розроблюваної системи проведемо порівняльний аналіз двох найбільш поширених родин мікроконтролерів: STM32 від STMicroelectronics та ESP32 від Espressif Systems (див. табл. 1.3).

Архітектурні особливості та переваги платформи STM32 полягають у тому, що дані мікроконтролери базуються на 32-бітних RISC-ядрах ARM Cortex-M (M0+, M3, M4, M33). Ця архітектура використовує гарвардську модель пам'яті, що передбачає розділені шини для інструкцій та даних, дозволяючи виконувати вибірку команди та читання операндів за один такт. Важливою перевагою для систем реального часу є наявність контролера вкладених векторних переривань (NVIC - Nested Vectored Interrupt Controller), який забезпечує детермінований і мінімальний час реакції на зовнішні події (наприклад, сигнал готовності даних від сенсора) [3].

Для портативних систем збору даних особливий інтерес становить лінійка STM32L (Low Power), наприклад, STM32L4. Ці мікроконтролери оптимізовані для наднизького енергоспоживання (Ultra-Low Power) завдяки технології динамічного масштабування напруги та розгалуженій системі тактування. У режимі Standby струм споживання може становити менше 1 мкА. Крім того, STM32 мають потужну аналогову периферію: багатоканальні 12- або 16-бітні АЦП послідовного наближення (SAR ADC), апаратні операційні підсилювачі та компаратори. Використання контролера прямого доступу до пам'яті (DMA - Direct Memory Access) дозволяє передавати дані від

периферії (I2C/SPI) в оперативну пам'ять (SRAM) без участі центрального процесора, що додатково економить заряд батареї.

Головним недоліком базових серій STM32 у контексті сучасних портативних IoT-рішень є відсутність вбудованих радіомодулів. Для передачі даних на сервер розробнику необхідно додавати на плату зовнішні трансивери (Wi-Fi або LoRa), що збільшує масо-габаритні показники пристрою, ускладнює розведення друкованої плати та підвищує кінцеву вартість виробу.

Архітектурні особливості та переваги платформи ESP32 полягають у тому, що система на кристалі ESP32 є високоінтегрованим рішенням, створеним спеціально для додатків Інтернету речей (IoT). Базова архітектура спирається на двоядерний процесор Xtensa Dual-Core 32-bit LX6 (у новіших ревізіях також використовуються ядра RISC-V). Головною перевагою платформи є інтеграція радіочастотного тракту стандарту 802.11 b/g/n (Wi-Fi) та Bluetooth (Classic і BLE) безпосередньо в кристал мікросхеми [2].

Наявність двох ядер дозволяє реалізувати симетричну багатопроцесорну обробку (SMP). У типовому сценарії, який підтримується операційною системою реального часу FreeRTOS, одне ядро відповідає за обслуговування мережевого стеку (TCP/IP, обробка Wi-Fi з'єднання), тоді як інше ядро повністю виділяється під завдання користувача: опитування сенсорів, математичну фільтрацію та підготовку пакетів даних [14]. Це гарантує, що ресурсоємні процеси передачі даних не заблокують зчитування критично важливих показань із датчиків.

ESP32 також містить унікальний співпроцесор з наднизьким енергоспоживанням (ULP - Ultra-Low Power coprocessor). У режимі Deep Sleep основні ядра Xtensa та радіомодуль повністю знеструмлюються, і струм споживання падає до 10-15 мкА. У цей час ULP-співпроцесор залишається активним: він може періодично прокидатися, зчитувати дані з АЦП або шини I2C і, якщо значення перевищує заданий поріг, “будити” основну систему для екстреної передачі повідомлення.

Таблиця 1.3 - Порівняльна характеристика мікроконтролерів для портативних систем

Характеристика	STM32L476	ESP32 (WROOM-32)
Архітектура ядра	ARM Cortex-M4F (32-bit)	Xtensa Dual-Core LX6 (32-bit)
Тактова частота	До 80 МГц	До 240 МГц
Бездротові інтерфейси	Відсутні (потрібен зовн. модуль)	Вбудовані Wi-Fi 2.4 ГГц, Bluetooth v4.2 BR/EDR та BLE
Flash-пам'ять / SRAM	1 МБ / 128 КБ	4 МБ (зовнішня) / 520 КБ
Струм у режимі сну (Deep Sleep)	~1-3 мкА	~10-15 мкА
Апаратна підтримка ОС	Bare-metal, FreeRTOS	Нативна підтримка FreeRTOS

Порівняльний аналіз даних таблиці 1.3 показує, що мікроконтролер STM32L476 має вищу енергоефективність у режимі сну, що є важливим для автономності пристрою. Проте платформа ESP32 суттєво переважає за тактовою частотою, обсягом пам'яті та наявністю вбудованих модулів Wi-Fi й Bluetooth. Відсутність інтегрованого радіотракту у STM32 вимагає використання зовнішніх плат розширення, що ускладнює схемотехніку та збільшує загальні габарити портативної системи.

1.4 Обґрунтування вибору апаратної платформи

Аналіз технічних характеристик показує, що мікроконтролери STM32 мають перевагу у прецизійних завданнях із жорсткими обмеженнями на енергоспоживання, проте вимагають додаткових витрат на реалізацію бездротового зв'язку. У свою чергу, платформа ESP32 пропонує безпрецедентний рівень інтеграції обчислювальних та комунікаційних можливостей “з коробки”.

Оскільки розроблювана портативна система збору та обробки даних передбачає передачу накопичених масивів інформації на зовнішні пристрої (мобільний додаток або хмарний сервер) і потребує високої обчислювальної потужності для цифрової фільтрації сигналів, використання мікроконтролера ESP32 є найбільш раціональним. Інтегрований модуль Bluetooth Low Energy

(BLE) забезпечить енергоефективну передачу даних на невеликі відстані, а наявність Wi-Fi дозволить пристрою підключатися до глобальної мережі для синхронізації часу (NTP) та відправки логів. Дещо вище споживання струму в сплячому режимі порівняно з STM32 може бути компенсоване оптимізацією алгоритмів (Firmware) та використанням ULP-співпроцесора.

У першому розділі було проведено аналіз принципів побудови сучасних портативних систем збору та обробки даних (DAQ). Визначено, що ключовими вимогами до таких апаратно-програмних комплексів є висока енергоефективність, компактність, захищеність та можливість локальної цифрової обробки сигналів безпосередньо на вузлі збору інформації.

Розглянуто сучасну елементну базу сенсорного рівня. Встановлено, що перехід від аналогових датчиків до високоінтегрованих цифрових сенсорів, які використовують шини передачі даних I2C та SPI, дозволяє суттєво знизити вплив електромагнітних завад та підвищити точність вимірювань. Для боротьби з апаратним шумом та імпульсними завадами обґрунтовано застосування програмних алгоритмів фільтрації: ковзного середнього та медіанного фільтра.

На основі порівняльного аналізу архітектурних особливостей популярних мікроконтролерних платформ (STM32 та ESP32) здійснено вибір апаратної бази для розроблюваної системи. Платформу ESP32 визнано найбільш раціональним рішенням завдяки наявності вбудованих радіомодулів (Wi-Fi, Bluetooth Low Energy), потужній двоядерній архітектурі, підтримці ОС FreeRTOS та наявності енергоефективного ULP-співпроцесора.

Отримані результати теоретичного аналізу та обґрунтована апаратна база слугуватимуть основою для подальшого проєктування структурної та функціональної схем пристрою.

РОЗДІЛ 2 ПРОЄКТУВАННЯ АПАРАТНОЇ ТА ПРОГРАМНОЇ ЧАСТИНИ СИСТЕМИ

2.1 Проєктування структурної схеми системи

Розробка будь-якої мікропроцесорної системи починається зі створення її структурної схеми, яка визначає основні апаратні вузли пристрою та логічні зв’язки між ними. Для реалізації портативної системи збору та обробки даних було обрано модульний підхід, що дозволяє легко масштабувати систему та замінювати окремі компоненти без зміни загальної архітектури.

Апаратна частина розроблюваної системи логічно поділяється на п'ять основних блоків:

1. Блок обробки та керування (Центральний мікроконтролер) - є ядром системи, що відповідає за ініціалізацію периферії, опитування датчиків, математичну обробку даних та координацію інших модулів.
2. Блок збору даних (Сенсорний модуль) - відповідає за перетворення фізичних величин (у нашому випадку - рівня освітленості) у цифровий сигнал.
3. Блок локальної індикації (Дисплей) - забезпечує візуалізацію отриманих результатів безпосередньо на місці вимірювання.
4. Блок комунікації (Мережевий інтерфейс) - вбудований бездротовий модуль для передачі інформації на зовнішні пристрої.
5. Блок живлення - забезпечує стабільною напругою (5В та 3.3В) мікроконтролер та периферійні пристрої.

Обмін даними між центральним мікроконтролером, датчиком та дисплеєм відбувається за допомогою єдиної цифрової шини I2C, що дозволяє мінімізувати кількість використовуваних сигнальних ліній (пінів). Структурна організація розробленої системи наведена у вигляді UML-діаграми (див. рис. 2.1).

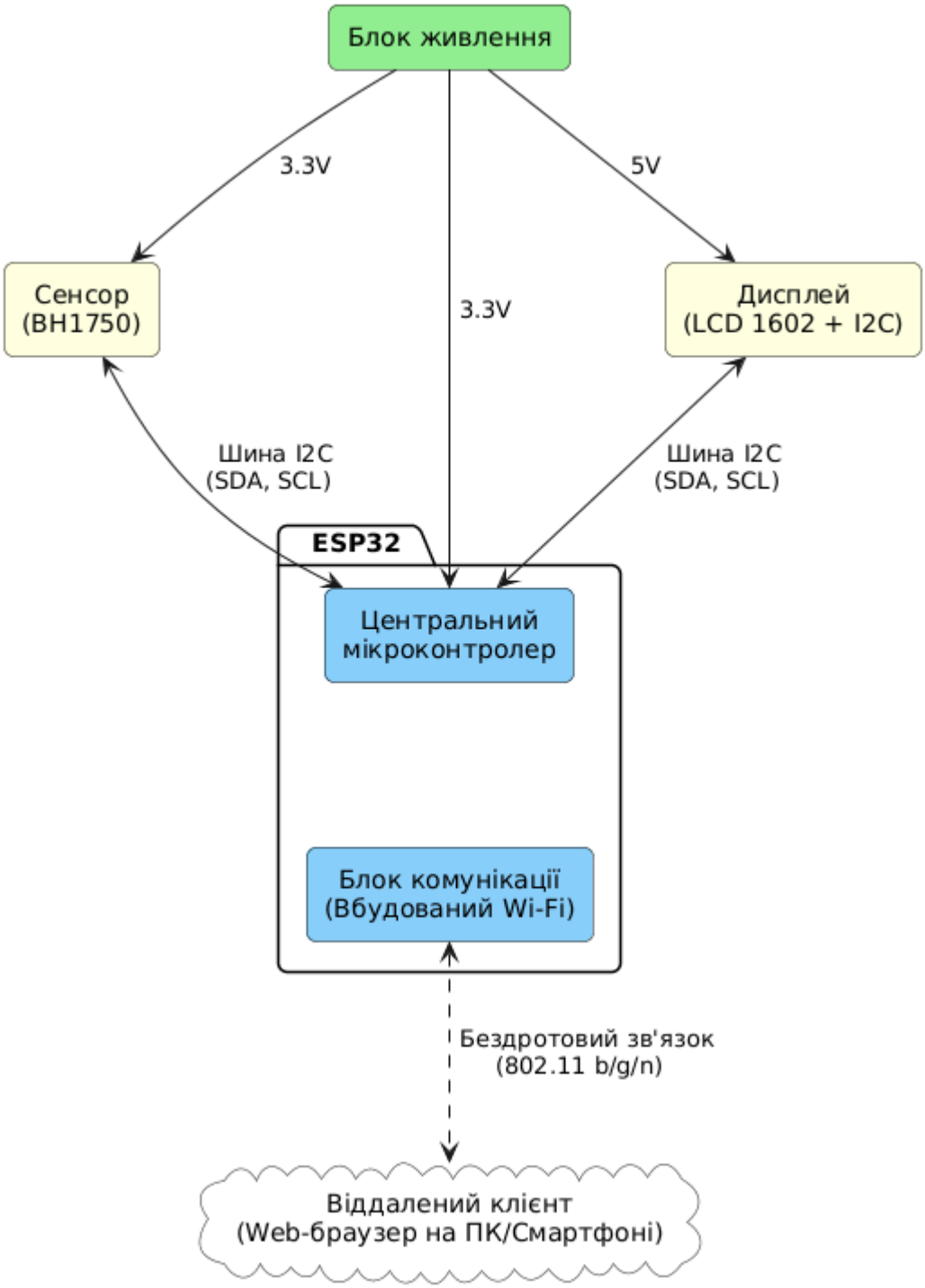

Рисунок 2.1 - Структурна схема портативної системи збору даних

Як видно з наведеної діаграми, архітектура розробленої системи має централізовану топологію на рівні логічних зв'язків, де головним керуючим вузлом виступає мікроконтролер ESP32. Він виконує роль системного координатора, який ініціює опитування сенсорів, синхронізує роботу локальної індикації та керує бездротовим обміном даними. Використання

єдиного обчислювального центру дозволяє оптимізувати внутрішній трафік пристрою та суттєво спростити загальний алгоритм керування.

Такий підхід забезпечує високу стійкість системи до апаратних збоїв. Наприклад, тимчасова втрата зв'язку з дисплеєм або датчиком через фізичні фактори не призведе до критичної помилки чи повного зависання програми. Мікроконтролер здатен ізолювати проблемний вузол, продовжити обробку мережевих запитів у фоновому режимі та своєчасно інформувати віддаленого клієнта про статус периферії. Крім того, подібна модульна організація зв'язків робить апаратну платформу легко масштабованою - у разі потреби розширення функціоналу до системи можна швидко підключити додаткові I2C-датчики без зміни базової архітектури проєкту.

2.2 Розробка функціональної схеми системи

На основі розробленої структурної архітектури деталізуємо функціональне призначення кожного вузла та алгоритм їхньої апаратної взаємодії. Функціональна схема розкриває внутрішню будову кожного апаратного блоку та фізичні принципи перетворення сигналів на всіх етапах: від зчитування фізичної величини до її відображення та передачі в мережу. Детальний аналіз цих вузлів дозволяє виявити та нівелювати потенційні апаратні конфлікти (наприклад, неузгодженість логічних рівнів напруги) ще на етапі проєктування.

Головним обчислювальним та керуючим вузлом на єдиній системній шині даних виступає центральний мікропроцесорний модуль на базі ESP32-WROOM-32D.

Функціонально його робота спирається на високопродуктивний двоядерний процесор архітектури Xtensa Dual-Core 32-bit LX6. Наявність двох незалежних ядер є критично важливою перевагою для систем Інтернету речей: одне ядро апаратно виділяється під обслуговування мережевого стеку TCP/IP та радіотракту Wi-Fi, що дозволяє уникнути блокувань і затримок під час передачі пакетів. Друге ядро повністю присвячене виконанню коду

користувача - генерації тактових імпульсів на шині I2C, опитуванню сенсорів та математичній фільтрації даних. З точки зору енергетики, ядро мікроконтролера працює з логічними рівнями 3.3 В. Оскільки під час активної передачі даних по Wi-Fi пікове споживання струму може досягати 500 мА, на платі розробника інтегровано LDO-регулятор напруги (Low-Dropout Regulator), який перетворює вхідні 5 В (від USB або акумулятора) у стабільні 3.3 В для живлення чипа та підтяжки сигнальних ліній [2].

За безпосередню взаємодію з навколишнім середовищем відповідає блок первинного перетворення на основі сенсора освітленості BH1750. Датчик BH1750FVI є високоінтегрованою цифровою мікросхемою, що працює в режимі підпорядкованого пристрою. На відміну від класичних аналогових фоторезисторів (наприклад, GL5528), які потребують зовнішніх дільників напруги та використання АЦП мікроконтролера, BH1750 здійснює повний цикл обробки сигналу на власному кристалі.

Функціонально він складається зі спеціалізованого світлочутливого елемента (спектральна чутливість якого апаратно наближена до кривої чутливості людського ока для відсікання інфрачервоного спектра), операційного підсилювача та прецизійного 16-бітного аналогово-цифрового перетворювача інтегруючого типу. Оцифровані дані зберігаються у внутрішніх регістрах мікросхеми та передаються центральному мікроконтролеру через інтерфейс I2C у вигляді готових значень в люксах (lx), що повністю знімає обчислювальне навантаження з ESP32 [7].

Для забезпечення візуального контролю параметрів безпосередньо на місці розгортання системи використовується модуль локальної візуалізації, який поєднує символьний дисплей LCD 1602 та розширювач портів PCF8574. Рідкокристалічний екран формату 16x2 побудований на базі стандартного контролера HD44780 [15]. Класичне підключення такого дисплея є вкрай неефективним для сучасних мікроконтролерів, оскільки вимагає використання паралельної шини даних та займає від 6 до 10 виводів (GPIO) загального

призначення. Крім того, логіка роботи дисплея розрахована на 5 В, що створює ризик пошкодження 3.3-вольтових портів ESP32.

Для вирішення цієї інженерної проблеми у функціональну схему інтегровано спеціальний апаратний модуль - I2C-адаптер (розширювач портів вводу-виводу) на базі мікросхеми PCF8574T, яка виконує роль мосту-перетворювача (конвертера інтерфейсів). Принцип її дії полягає в тому, що мікросхема PCF8574T підключається до мікроконтролера всього двома сигнальними лініями (SDA та SCL) і виступає як ще один пристрій на шині I2C зі стандартною адресою 0x27. Вона приймає від ESP32 послідовний цифровий код, буферизує його та апаратно перетворює у 8-бітний паралельний сигнал, який подається безпосередньо на контакти контролера HD44780 [15]. На платі адаптера також розміщено підстроювальний резистор (потенціометр) для апаратного регулювання контрастності матриці та транзисторний ключ із перемичкою для програмного або ручного керування світлодіодною підсвіткою екрана. Використання адаптера PCF8574T не лише звільняє порти мікроконтролера для підключення додаткових сенсорів, але й виступає буфером для узгодження логічних рівнів, роблячи систему надійною та захищеною від паразитних струмів [8].

Фінальну ланку архітектури становить блок комунікації (радіотракт Wi-Fi), який відповідає за передачу даних до зовнішніх пристроїв. Цей мережевий інтерфейс апаратно інтегрований у систему на кристалі ESP32 і відповідає стандарту IEEE 802.11 b/g/n (діапазон 2.4 ГГц). У функціональній схемі зазначений блок забезпечує стабільний зв'язок пристрою з локальною бездротовою інфраструктурою підприємства або розумного будинку. Завдяки вбудованій друкованій антені, модуль гарантує надійну трансляцію пакетів даних на відстань до 50 метров у приміщенні, що дозволяє розгорнути віддалений веб-сервер для моніторингу показань сенсорів у режимі реального часу [9].

2.3 Алгоритмізація процесів збору та фільтрації даних

Надійність роботи портативної системи збору даних визначається не лише апаратною базою, але й архітектурою вбудованого програмного забезпечення. Зважаючи на використання мікроконтролера ESP32, який одночасно обслуговує апаратні переривання від датчиків та ресурсоємний мережевий стек Wi-Fi, класична лінійна алгоритмізація з використанням жорстких затримок (функцій блокування потоку) є неприпустимою. Тому для розробки програмної частини було обрано подієво-орієнтовану архітектуру на базі неблокуючих системних таймерів.

Важливою складовою ПЗ виступає алгоритм ініціалізації та апаратного опитування сенсора. Робота програми починається з конфігураційного блоку setup(), де мікроконтролер послідовно виконує низку обов'язкових процедур. Спочатку здійснюється ініціалізація апаратного модуля I2C та налаштування швидкості шини на 100 кГц. Наступним кроком є програмне скидання датчика BH1750 шляхом відправки команди 0x07 (Reset). Після цього відбувається переведення датчика в режим безперервного вимірювання високої роздільної здатності за допомогою команди 0x10 (Continuous H-Resolution Mode). Завершується початковий етап підняттям Wi-Fi станції та ініціалізацією веб-сервера.

Оскільки датчику BH1750 потрібно орієнтовно 120 мілісекунд для виконання одного циклу аналого-цифрового перетворення, програма не блокує процесор на цей час, а замість цього використовує системний лічильник мілісекунд. Сам алгоритм зчитування “сирих” даних працює на побітовому рівні. Мікроконтролер відправляє запит на читання за адресою 0x23 і приймає два байти даних (старший HighByte та молодший LowByte). Оскільки 16-бітний результат передається частинами, мікроконтролер виконує операцію побітового зсуву вліво на 8 позицій для старшого байта та логічне додавання (АБО) з молодшим байтом. Після цього, згідно з технічною документацією виробника, отримана сума ділиться на оптичний коефіцієнт 1.2 для отримання реального фізичного значення у люксах (див. рис. 2.2).

Окрему увагу в роботі приділено програмній реалізації та оптимізації алгоритмів фільтрації. Як було обґрунтовано в теоретичній частині, для усунення високочастотного шуму АЦП застосовується алгоритм ковзного середнього. Проте класична математична реалізація цього методу з повним перерахунком суми всього масиву при кожній новій вибірці створює надлишкове обчислювальне навантаження на процесор. Для розв'язання цієї проблеми алгоритм був програмно оптимізований з використанням структури даних “циклічний буфер” (Circular Buffer).

У статичній оперативній пам'яті (SRAM) мікроконтролера виділяється масив фіксованого розміру (наприклад, на 10 елементів) та глобальна змінна, що зберігає загальну суму цих елементів. Завдяки цьому алгоритм обробки нового значення має обчислювальну складність $O(1)$ і виконується за чітко визначеною послідовністю дій. Спочатку за допомогою вказівника чи індексу, який циклічно змінюється від 0 до 9, алгоритм знаходить найстаріше значення у буфері. Далі це найстаріше значення віднімається від загальної збереженої суми, а на його місце в масив одразу записується щойно зчитане нове значення від датчика, яке згодом додається до загальної суми.

Розрахунок оновленої суми елементів циклічного буфера на кожному кроці ітерації здійснюється у формулі (2.1):

$$Sum_{new} = Sum_{old} - X_{oldest} + X_{new}, \quad (2.1)$$

Після визначення поточної суми, остаточний відфільтрований результат обчислюється у формулі (2.2) шляхом ділення отриманого значення на розмір буфера:

$$Y_{filtered} = \frac{Sum_{new}}{N}, \quad (2.2)$$

де N - розмір вікна циклічного буфера.

Такий підхід повністю усуває необхідність використання циклів for для перерахунку масиву, економлячи такти процесора ESP32 для більш пріоритетних завдань мережевого обміну.

Логічним завершенням архітектури вбудованого ПЗ став алгоритм асинхронної веб-комунікації. Обробка клієнтських запитів винесена в окремий асинхронний потік. Замість того, щоб у головному циклі програми постійно перевіряти наявність нових клієнтів методом поллінгу, використовується механізм зворотних викликів. Коли клієнтський додаток (веб-браузер) надсилає HTTP GET-запит на спеціальний URI маршрут /read_lux, обробник переривань мережевого стека автоматично призупиняє фонові завдання, зчитує останнє розраховане значення з буфера фільтрації (змінна Yfiltered), конвертує його в текстовий формат і миттєво відправляє пакет даних клієнту. Це забезпечує мінімальний час відгуку системи та виключає можливість “пропуску” клієнтського запиту під час опитування датчика.

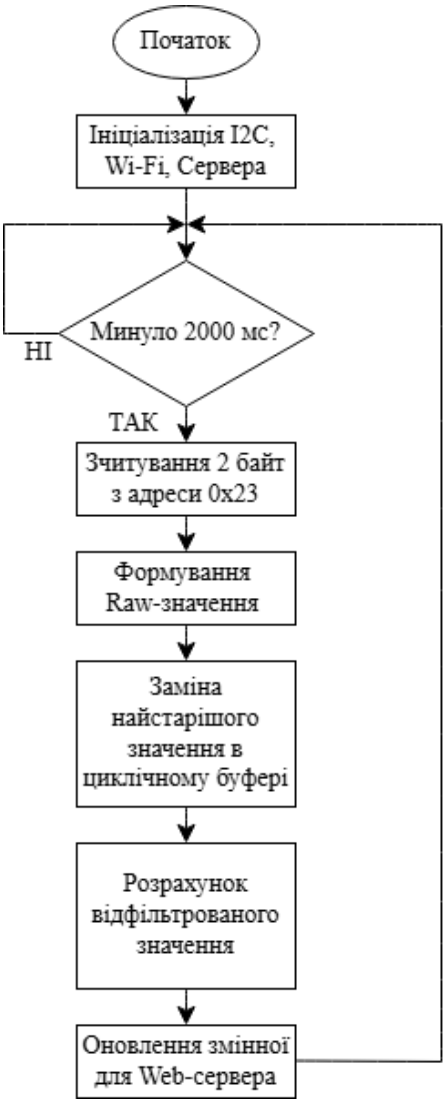

Рисунок 2.2 - Блок-схема алгоритму роботи портативної системи

2.4 Вибір інтерфейсів передачі даних та протоколів зв’язку

Для забезпечення ефективного та надійного обміну інформацією як між внутрішніми компонентами пристрою, так і із зовнішнім світом, розроблена система використовує багаторівневу комунікаційну архітектуру. Її можна логічно розділити на апаратний рівень, призначений для локальної взаємодії мікроконтролера з периферією, та мережевий рівень для віддаленого моніторингу.

Як основний внутрішньосистемний інтерфейс апаратного рівня обрано напівдуплексну синхронну послідовну шину I2C (Inter-Integrated Circuit). Вибір саме цього стандарту обґрунтований жорсткими вимогами до мінімізації габаритів портативної системи та економії портів вводу-виводу (GPIO) мікроконтролера ESP32. Шина I2C дозволяє підключати десятки периферійних пристроїв, використовуючи лише дві фізичні сигнальні лінії: лінію передачі даних (SDA) та лінію тактування або синхронізації (SCL), імпульси на якій генерує виключно Master-пристрій. У межах розробленої системи на шині функціонують два підпорядковані (Slave) пристрої: цифровий датчик освітленості BH1750 з апаратною адресою 0x23 та I2C-експандер дисплея PCF8574T з адресою 0x27. Обидва пристрої підключені до портів GPIO 21 та GPIO 22 мікроконтролера. Оскільки транзистори вихідних каскадів I2C-пристроїв мають конфігурацію “відкритий сток” (Open-Drain), для формування логічної одиниці на лініях використано підтягуючі резистори (Pull-up) номіналом 4.7 кОм, підключені до шини живлення 3.3 В. Система функціонує у стандартному швидкісному режимі з частотою тактування 100 кГц, що забезпечує високу завадостійкість під час зчитування показань [6].

На мережевому рівні інтеграція портативної системи збору даних у локальну обчислювальну мережу здійснюється за допомогою бездротового зв'язку стандарту Wi-Fi. Для цього використовується вбудований у мікроконтролер радіочастотний трансивер стандарту IEEE 802.11 b/g/n, що працює у неліцензованому діапазоні частот 2.4 ГГц. Програмне забезпечення конфігурує ESP32 для роботи в режимі бездротового клієнта (Station Mode).

У цьому режимі пристрій автентифікується на локальному маршрутизаторі з використанням протоколу безпеки WPA2-PSK. Конфігурація мережевих параметрів, таких як IP-адреса, маска підмережі та шлюз, відбувається автоматично за допомогою протоколу динамічного налаштування вузла (DHCP). Такий підхід робить систему справді портативною (Plug-and-Play): при переміщенні в нове місце достатньо лише змінити SSID та пароль мережі в налаштуваннях, після чого пристрій автоматично увійде в робочий режим.

Прикладний рівень взаємодії базується на протоколі HTTP та технології AJAX. Головним завданням цього комунікаційного блоку є візуалізація зібраних та відфільтрованих даних для кінцевого користувача, для чого у пам'яті мікроконтролера розгорнуто асинхронний веб-сервер. Хоча базовим протоколом виступає HTTP, його класична модель із повним завантаженням HTML-документа при кожному оновленні є вкрай неефективною для вбудованих систем. Вона перевантажує процесор, забиває радіоканал і призводить до візуального “мерехтіння” сторінки.

Для оптимізації веб-інтерфейсу було імплементовано технологію AJAX (Asynchronous JavaScript and XML) [1]. Алгоритм взаємодії побудовано таким чином, що при першому зверненні клієнта за IP-адресою пристрою веб-сервер відправляє “важку” статичну оболонку, яка містить HTML-каркас, CSS-стилі та JavaScript-логіку. Після завантаження сторінки у браузері вбудований клієнтський скрипт починає працювати у фоновому режимі. Кожні кілька секунд він відправляє на мікроконтролер короткий цільовий GET-запит на специфічний маршрут (наприклад, /read_lux). Мікроконтролер, не перериваючи процес опитування сенсорів, відповідає на запит пакетом типу Plain Text, який містить лише актуальне числове значення освітленості. Отримавши ці дані, JavaScript на стороні клієнта асинхронно оновлює лише один конкретний елемент у DOM-дереві веб-сторінки. Таке архітектурне рішення забезпечує надшвидкий відгук системи, дозволяє відображати зміну

фізичних величин у режимі реального часу та суттєво знижує енерговитрати на радіопередачу пакета.

У другому розділі було здійснено комплексне проєктування апаратної та програмної архітектури портативної системи збору та обробки даних. За результатами розробки спроєктовано структурну та функціональну схеми пристрою на базі високопродуктивного двоядерного мікроконтролера ESP32. При цьому обґрунтовано застосування модульного підходу з використанням шини I2C як головного локального інтерфейсу. Застосування I2C-експандера PCF8574T для дисплея LCD 1602 та цифрового датчика BH1750 дозволило не лише мінімізувати кількість задіяних портів GPIO мікроконтролера, але й забезпечити безпечне узгодження різних логічних рівнів напруги (3.3 В та 5 В).

Також було розроблено подієво-орієнтовану архітектуру програмного забезпечення. Відмова від класичних лінійних алгоритмів із жорстким блокуванням процесора на користь системних таймерів дозволила ефективно розділити обчислювальне навантаження: одне ядро безперебійно обслуговує ресурсоємний стек Wi-Fi, тоді як інше виконує опитування периферії без втрати швидкодії. Разом із цим оптимізовано математичний алгоритм цифрової фільтрації шляхом програмної реалізації фільтра ковзного середнього. Застосування структури “циклічний буфер” дозволило знизити обчислювальну складність алгоритму до математичного мінімуму ($O(1)$), економлячи такти процесора для мережевих завдань.

Крім того, визначено стек технологій для віддаленого моніторингу та розгорнуто асинхронний веб-сервер. Впровадження технології AJAX забезпечило миттєве оновлення показників освітленості на клієнтському пристрої у фоновому режимі, що радикально знижує мережевий трафік та економить заряд акумулятора. Спроєктовані апаратні рішення та розроблені алгоритми утворюють надійне підґрунтя для етапу фізичної збірки системи, написання кінцевого програмного коду та тестування пристрою в умовах реального часу.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ, НАЛАГОДЖЕННЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Реалізація вбудованого програмного забезпечення

Програмне забезпечення для мікроконтролера ESP32 було розроблено мовою C++ у спеціалізованому інтегрованому середовищі розробки Arduino IDE [10]. Архітектура вбудованого ПЗ базується на подієво-орієнтованому підході з використанням апаратних таймерів. Відмова від класичних лінійних затримок (функції delay) на користь неблокуючого системного лічильника millis() дозволила реалізувати псевдопаралельне виконання завдань: мікроконтролер одночасно обслуговує опитування сенсорів, оновлення локального екрана та обробку асинхронних мережевих запитів від клієнтів.

Ключовою інженерною та алгоритмічною проблемою під час реалізації стала необхідність паралельного підключення двох периферійних пристроїв - цифрового датчика освітленості BH1750 та символьного дисплея LCD 1602. Оскільки в умовах проєкту використовувалися з'єднувальні провідники типу “мама-мама” без додаткових апаратних розгалужувачів шини, було прийнято нестандартне рішення: використання вбудованого в ESP32 матричного комутатора сигналів (GPIO Matrix) для динамічного перепризначення виводів шини I2C [2]. Взаємне розташування компонентів портативної системи та схему їхнього фізичного з'єднання провідниками без використання зовнішніх апаратних розгалужувачів наведено на рисунку 3.1.

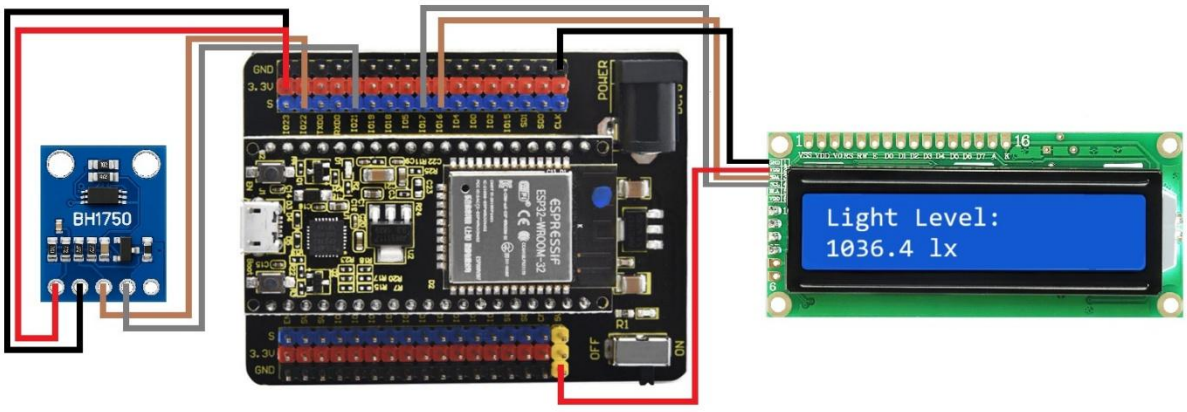

Рисунок 3.1 - Схема з'єднання елементів портативної системи

У програмному коді реалізовано почергове використання двох незалежних груп портів: GPIO 21 (SDA) та GPIO 22 (SCL) виділено для взаємодії із сенсором, а GPIO 16 та GPIO 17 - для передачі даних на I2C-експандер дисплея. Щоб уникнути апаратних конфліктів та ефекту “зависання” шини через залишкові електричні потенціали, перед кожним перемиканням портів викликається метод повного програмного скидання ліній Wire.end() із наступною мікрозатримкою. Це дозволяє очистити внутрішні регістри мікроконтролера та гарантує коректну передачу стартових умов для наступного пристрою [6].

Зчитування даних із сенсора BH1750 виконано на низькому рівні методом прямого звернення до регістрів (адреса 0x23) [7]. Програма відправляє команду 0x10 для запуску режиму безперервного вимірювання високої роздільної здатності. Отримані два байти інформації (High Byte та Low Byte) об'єднуються у 16-бітну змінну за допомогою операції побітового зсуву вліво на 8 розрядів та логічного додавання (АБО). Після цього результат ділиться на калібрувальний оптичний коефіцієнт 1.2 згідно з технічною документацією виробника [7].

Для усунення імпульсних завад та флуктуацій показників реалізовано програмний фільтр ковзного середнього [5]. З метою мінімізації навантаження на процесор, класичний математичний масив було замінено структурою даних “циклічний буфер” на 10 елементів. Використання оператора ділення по модулю ($\%\, N$) для обчислення індексу масиву дозволило досягти обчислювальної складності алгоритму $O(1)$, що значно економить такти ядра для обслуговування мережевого стеку [5].

3.2 Розробка інтерфейсу користувача для візуалізації даних

Для забезпечення зручного та кросплатформного віддаленого моніторингу параметрів освітленості було спроєктовано графічний веб-інтерфейс. В основі мережевої взаємодії лежить розгорнутий на базі ESP32 асинхронний веб-сервер [9]. Мікроконтролер конфігурується в режимі

бездротової станції та автоматично отримує IP-адресу від локального маршрутизатора за допомогою протоколу DHCP. Для забезпечення гібридного контролю, отримана IP-адреса виводиться на фізичний LCD-дисплей під час ініціалізації пристрою, що робить систему незалежною від зовнішніх серіальних моніторів та зручною в розгортанні (plug-and-play).

Сам інтерфейс користувача (Front-End частина) розроблено з використанням сучасних стандартів HTML5 та каскадних таблиць стилів CSS3 [11]. Дизайн веб-сторінки побудовано за принципом адаптивності, що гарантує коректне відображення інформації як на широкоформатних моніторах, так і на екранах мобільних пристроїв. Для забезпечення високого рівня контрастності та ергономіки, показники виводяться великим шрифтом у спеціальному виділеному блоці-картці. HTML-код інтерфейсу зберігається безпосередньо у Flash-пам'яті ESP32 у вигляді константної рядкової змінної, що економить цінну оперативну пам'ять (SRAM) [11].

Для підвищення інформативності та забезпечення комплексного аналізу вимірювань у реальному часі, початковий інтерфейс користувача було модернізовано до рівня інтерактивної панелі керування (IoT Dashboard). Оновлена архітектура веб-інтерфейсу дозволяє здійснювати одночасний моніторинг трьох функціональних блоків: поточних вимірювань, аналітики поточної сесії та системної телеметрії пристрою.

Важливою інженерною особливістю реалізації є те, що обчислення статистичних параметрів - максимального (MAX), мінімального (MIN) та середнього (AVG) значень освітленості - повністю винесено на сторону клієнта за допомогою обробки масивів даних у JavaScript.

Це дозволило реалізувати додатковий аналітичний функціонал без створення надлишкового обчислювального навантаження на ядро мікроконтролера ESP32.

Для забезпечення надійності функціонування системи у структурі дашборду передбачено блок системної телеметрії. Завдяки зчитуванню внутрішніх параметрів мікроконтролера, на сторінку динамічно виводяться

час безперервної роботи пристрою (Uptime), рівень сигналу бездротової мережі (Wi-Fi RSSI) та обсяг вільної оперативної пам'яті (Free RAM). Також реалізовано інтерактивний механізм світлозвукової тривоги: користувач може самостійно задати критичний поріг освітленості за допомогою поля вводу. При падінні показників нижче встановленої норми JavaScript динамічно змінює CSS-стилі, викликаючи миттєве блимання індикатора та виведення попереджувального повідомлення. Повний лістинг вихідного коду розробленого програмно-апаратного комплексу (включаючи конфігурацію мікроконтролера та скрипти веб-інтерфейсу) наведено в Додатку А.

Графічний вигляд розробленої автономної панелі керування (IoT Dashboard) під час моніторингу навколишнього середовища у реальному часі наведено на рисунку 3.2.

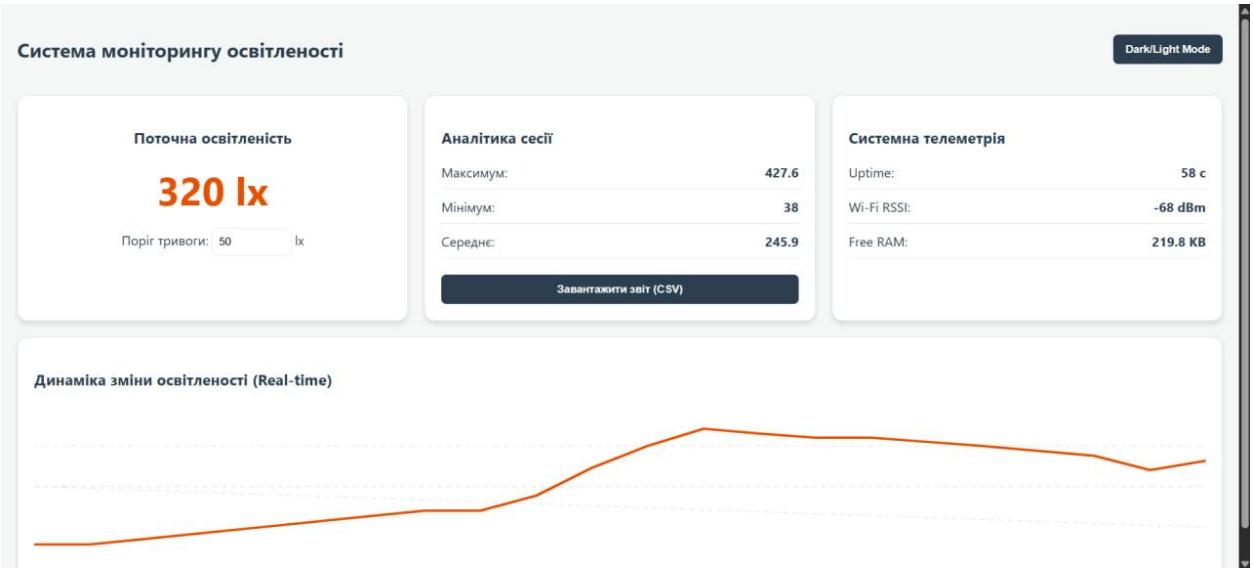

Рисунок 3.2 - Інтерфейс користувача портативної системи

Ключовою технологією, що забезпечує оновлення даних у реальному часі без перезавантаження сторінки, стала реалізація механізму AJAX (Asynchronous JavaScript and XML) [12]. Вбудований у веб-сторінку клієнтський JavaScript-код кожні 2000 мілісекунд (синхронно з циклом опитування сенсорів мікроконтролером) ініціює об'єкт XMLHttpRequest. Цей об'єкт генерує цільовий HTTP GET-запит на прихований маршрут сервера

/read_lux. Мікроконтролер, використовуючи лямбда-функції обробки маршрутів, миттєво формує відповідь у форматі Plain Text, яка містить лише актуальне числове значення освітленості. Після отримання відповіді, JavaScript динамічно змінює вміст DOM-дерева веб-сторінки [12], забезпечуючи плавну та непомітну для користувача актуалізацію даних.

3.3 Енергоаудит системи та оптимізація автономності

У процесі апаратної реалізації та збирання прототипу пристрою було виявлено ряд особливостей, що впливають на енергоспоживання та стабільність роботи компонентів. ESP32 із увімкненим радіомодулем Wi-Fi генерує імпульсні навантаження на систему живлення (до 300-400 мА під час передачі пакетів). Водночас модуль символьної індикації LCD 1602 потребує стабільного живлення для підтримки яскравості світлодіодної підсвітки.

Для вирішення проблеми падіння напруги та оптимізації загального енергетичного балансу системи було проведено розподіл ліній живлення. Чутливий цифровий сенсор BH1750, робота аналого-цифрового перетворювача якого критично залежить від чистоти напруги, підключено безпосередньо до стабілізованої лінії 3.3 В мікроконтролера [13]. Це повністю усуває ризик пошкодження його внутрішньої логіки та зменшує рівень теплових шумів. Натомість дисплейний модуль LCD 1602 із I2C-адаптером PCF8574T заживлено від шини 5 В (VIN) [8]. Таке розведення живлення дозволило отримати максимальну контрастність рідкокристалічної матриці без створення паразитних просадок напруги на вимірювальній частині схеми.

У поточній апаратній ревізії система орієнтована на безперервну роботу від зовнішнього джерела живлення (мережевого адаптера або портативного акумулятора підвищеної ємності). У перспективі для переведення пристрою в повністю автономний режим від Li-Ion акумулятора передбачається використання програмних режимів глибокого сну мікроконтролера ESP32, за яких живлення дисплея примусово відключатиметься транзисторним ключем, а система прокидатиметься лише для відправки телеметрії через Wi-Fi.

Серйозною проблемою під час розгортання портативних систем моніторингу у польових умовах є залежність веб-інтерфейсів від сторонніх інтернет-ресурсів та бібліотек, що завантажуються через зовнішні сервери контенту (CDN). У разі відсутності підключення до глобальної мережі Інтернет, завантаження таких бібліотек блокується, що призводить до аварійної зупинки виконання JavaScript-скриптів у браузері користувача.

Для вирішення цієї проблеми та забезпечення стовідстовкової автономності пристрою, у проєктуванні Front-End частини було відкинуто використання сторонніх важких фреймворків. Візуалізацію динаміки зміни освітленості реалізовано за допомогою вбудованої технології апаратної векторної графіки HTML5 SVG (Scalable Vector Graphics). Координати ліній графіка прораховуються математично безпосередньо у браузері клієнта на основі динамічного масиву люксів. Такий підхід дозволив отримати плавний графік у реальному часі без підключення до Інтернету, суттєво заощадити Flash-пам'ять мікроконтролера та забезпечити роботу системи в умовах повної ізоляції мережі (наприклад, при прямому підключенні до точки доступу Wi-Fi мікроконтролера ESP32).

Додатково в алгоритм побудови SVG-графіка було імплементовано функцію динамічного масштабування осі ординат. Скрипт автоматично визначає поточні мінімуми та максимуми утримуваного буфера вибірок (останні 30 точок) та адаптує під них робочі межі сітки екрана. Це забезпечує високу вимірну чутливість графіка до мінімальних змін фізичної величини (в межах 1-2 lx) при кімнатному чи штучному освітленні.

3.4 Тестування роботи системи в умовах реального часу

Налагодження, діагностика та фінальне тестування розробленого апаратно-програмного комплексу проводилися поетапно. На першому етапі перевірялася коректність ініціалізації шини I2C. Під час випробувань було виявлено проблему програмного конфлікту стандартних бібліотек при динамічному перемиканні портів GPIO. Бібліотечні функції втрачали зв'язок

із сенсором, генеруючи виняток про відсутність конфігурації пристрою. Ця проблема була успішно локалізована та усунена шляхом переходу на низькорівневе пряме управління регістрами датчика через базовий клас Wire. Додаткове впровадження програмних затримок перед ініціалізацією портів дозволило нівелювати вплив електричної ємності з'єднувальних провідників.

На другому етапі проводилося комплексне функціональне тестування сенсорної підсистеми та алгоритму цифрової фільтрації. Шляхом створення умов різкої зміни освітленості (штучне затінення пристрою непрозорим екраном та різке спрямування сфокусованого пучка світла в лінзу сенсора) досліджувалася перехідна характеристика системи. Алгоритм ковзного середнього з вікном у 10 вибірок продемонстрував високу ефективність: імпульсні зміни (викиди) згладжувалися, формуючи плавну криву зміни освітленості на екрані.

На фінальному етапі було проведено стрес-тестування мережевої підсистеми. Кілька клієнтських пристроїв одночасно підключалися до веб-сервера мікроконтролера, генеруючи постійний потік AJAX-запитів. Тестування підтвердило надійність архітектури: мікроконтролер ESP32, використовуючи багатозадачні можливості операційної системи реального часу, стабільно обслуговував HTTP-з'єднання без жодних пропусків у циклах апаратного опитування сенсора BH1750, забезпечуючи безперервний та достовірний моніторинг навколишнього середовища [5].

Під час стрес-тестування оновленої комунікаційної підсистеми було перевірено функцію довгострокового збору даних. Для забезпечення можливості подальшого наукового або промислового аналізу накопичених масивів інформації, в інтерфейс було інтегровано функцію “Завантажити звіт (CSV)”. При натисканні на відповідний елемент керування, JavaScript зчитує накопичену історію вимірювань (параметри часу клієнта та відповідні показники датчика), динамічно формує текстовий масив даних, розділених комою, та ініціює скачування файлу формату .csv на пристрій користувача,

який згодом може бути імпортований у табличні процесори на кшталт Microsoft Excel.

Також під час тестування ергономіки інтерфейсу та зручності первинного підключення було модернізовано конфігураційний блок setup() вбудованого ПЗ. Впровадження програмної затрими delay(10000) після успішної автентифікації у мережі Wi-Fi та підняття веб-сервера дозволило зафіксувати відображення отриманої IP-адреси на фізичному рідкокристалічному дисплеї пристрою на 10 секунд. Це повністю вирішило проблему швидкого затирання інформації основним робочим циклом loop() та забезпечило зручність зчитування мережевих параметрів оператором без підключення серійного монітора.

У третьому розділі було реалізовано програмно-апаратний комплекс портативної системи збору та обробки даних на базі мікроконтролера ESP32, а також проведено його комплексне налагодження та тестування в умовах реального часу. За результатами виконання практичної частини було сформовано низку важливих підсумків.

Практично доведено дієвість обраного підходу до проєктування шляхом ефективного вирішення апаратних обмежень. Важливим інженерним досягненням стала реалізація паралельної роботи двох I2C-пристроїв (сенсора BH1750 та дисплея LCD 1602) за умов використання різних пар портів вводу-виводу (GPIO 21/22 та 16/17). Завдяки розробленому алгоритму динамічного програмного перемикання шини та переходу на низькорівневе пряме зчитування регістрів сенсора, вдалося усунути апаратні конфлікти та помилки конфігурації без застосування додаткових мікросхем-розгалужувачів.

Розроблене вбудоване програмне забезпечення підтвердило свою швидкодію та надійність завдяки глибокій оптимізації. Реалізація подієво-орієнтованої архітектури з неблокуючими таймерами дозволила мікроконтролеру виконувати кілька завдань одночасно. Впровадження оптимізованого алгоритму цифрової фільтрації “ковзне середнє” з

використанням циклічного буфера ($O(1)$) забезпечило високу стабільність вимірювань рівня освітленості та ефективне згладжування імпульсних завад.

Успішна реалізація кросплатформного моніторингу за допомогою спроєктованого веб-інтерфейсу на базі асинхронного сервера та технології AJAX повністю закрила потребу у віддаленому контролі системи. Практичні випробування довели, що використання AJAX-запитів дозволяє динамічно оновлювати показники на клієнтських пристроях у фоновому режимі, не перевантажуючи мережевий трафік та не порушуючи цикл опитування сенсорної периферії.

З метою забезпечення енергетичної стабільності було проведено аудит та розділення ліній живлення (заживлення чутливого сенсора від 3.3 В, а індикатора - від 5 В). Таке апаратне рішення дозволило уникнути просадок напруги під час пікових навантажень радіомодуля Wi-Fi, тим самим гарантувавши безперебійну роботу всього обчислювального комплексу.

Отримані результати підтверджують, що розроблена портативна система збору та обробки даних функціонує стабільно, повністю відповідає визначеним на етапі проєктування технічним вимогам і готова до практичного застосування у сфері екологічного чи промислового моніторингу.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було вирішено завдання, що полягає у проєктуванні, розробці та апаратно-програмній реалізації портативної системи збору та обробки даних на базі сучасних мікроконтролерних технологій. Створений пристрій повністю відповідає технічним вимогам: він забезпечує високу точність вимірювань, ефективно фільтрує дані та має зручну панель для віддаленого моніторингу.

Використання мікроконтролера ESP32 виявилося оптимальним інженерним рішенням для цього проєкту. Завдяки наявності двох ядер вдалося ефективно розділити обчислювальні задачі. Одне ядро відповідає виключно за підтримку веб-сервера та з'єднання по Wi-Fi, тоді як інше виділене для безперебійного опитування датчика. Перехід на єдину цифрову шину I2C дозволив зручно підключити датчик освітленості BH1750 та символьний дисплей LCD 1602. При цьому продумане апаратне розділення ліній живлення на 3.3 В для чутливого сенсора та 5 В для екрана повністю усунуло збої і просадки напруги під час роботи радіомодуля.

Оскільки дисплей та сенсор через обмеження макетної плати довелося підключати до різних портів (GPIO 21/22 та 16/17), було розроблено унікальний алгоритм програмного перемикання шини I2C у процесі роботи. Це дозволило пристроям працювати паралельно без використання додаткових мікросхем-розгалужувачів. Водночас відмова від стандартних бібліотек і перехід на пряме низькорівневе керування датчиком суттєво підвищили загальну стабільність коду. Для згладжування різких стрибків освітленості та шумів було застосовано алгоритм “ковзного середнього”, а використання циклічного буфера зробило цей фільтр максимально швидким і розвантажило процесор мікроконтролера.

Важливим досягненням стала розробка сучасного автономного веб-інтерфейсу, який функціонує прямо з пам'яті пристрою. Завдяки технології AJAX дані на екрані смартфона чи ПК оновлюються кожні дві секунди у

фоновому режимі, не перевантажуючи бездротову мережу. Застосування вбудованої SVG-графіки дозволило малювати плавні графіки без доступу до глобальної мережі Інтернет, а всю зібрану статистику користувач може в один клік завантажити у форматі звітів CSV для подальшого аналізу.

Архітектура розробленої системи дозволяє дуже легко її модернізувати в майбутньому. Функціонал пристрою можна розширити шляхом переходу на автономне живлення від акумулятора та додавання нових датчиків температури чи вологості. Також можлива інтеграція з глобальними хмарними сервісами через протокол MQTT для застосування методів машинного навчання.

Підсумовуючи, можна стверджувати, що мета кваліфікаційної роботи досягнута повною мірою. Розроблена портативна система функціонує стабільно, є економічно вигідною та повністю готовою до практичного використання у складі сучасних комплексів екологічного, промислового чи побутового моніторингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Харченко В. С. Інтернет речей. Промислові та кіберфізичні системи. Навчальний посібник. Харків: НАУ “ХАІ”, 2018. 415 с.

2. Espressif Systems. ESP32 Technical Reference Manual. URL: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf (дата звернення: 13.04.2026).

3. STMicroelectronics. RM0351 Reference manual for STM32L476xx. URL: https://www.st.com/resource/en/reference_manual/rm0351.pdf (дата звернення: 13.04.2026).

4. Bosch Sensortec. BME280 Combined humidity and pressure sensor datasheet. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf (дата звернення: 17.04.2026).

5. Сергієнко А. Б. Цифрова обробка сигналів: підручник. Київ, 2010. 504 с.

6. NXP Semiconductors. I2C-bus specification and user manual. Rev. 6. 2014. URL: https://www.nxp.com/docs/en/user-manual/UM10204.pdf (дата звернення: 20.04.2026).

7. ROHM Semiconductor. BH1750FVI Digital Light Sensor IC Datasheet. URL: https://www.mouser.com/datasheet/2/348/bh1750fvi-e-186247.pdf (дата звернення: 01.05.2026).

8. NXP Semiconductors. PCF8574 Remote 8-bit I/O expander for I2C-bus. Datasheet. URL: https://www.nxp.com/docs/en/data-sheet/PCF8574_PCF8574A.pdf (дата звернення: 01.05.2026).

9. Espressif Systems. ESP32 Wi-Fi & Bluetooth Networking Library. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html (дата звернення: 04.05.2026).

10. Arduino. Arduino IDE Documentation. URL: https://docs.arduino.cc/software/ide-v2 (дата звернення: 09.05.2026).

11. W3C. HTML5 Specification. URL: https://html.spec.whatwg.org/ (дата звернення: 13.05.2026).

12. Mozilla Developer Network. AJAX Documentation. URL: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX (дата звернення: 15.05.2026).

13. Advanced Monolithic Systems. AMS1117 1A Low Dropout Voltage Regulator. Datasheet. URL: http://www.advanced-monolithic.com/pdf/ds1117.pdf (дата звернення: 17.05.2026).

14. Amazon Web Services. FreeRTOS User Guide. URL: https://www.freertos.org/ (дата звернення: 19.05.2026).

15. Hitachi. HD44780U (LCD-II) Dot Matrix Liquid Crystal Display Controller/Driver. URL: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf (дата звернення: 19.05.2026).

ДОДАТКИ

Додаток А - ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
#include <WiFi.h>
#include <WebServer.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

const char* ssid = "";
const char* password = "";
WebServer server(80);
LiquidCrystal_I2C lcd(0x27, 16, 2);
const int N = 10;
float buffer[N];
int bufferIndex = 0;
float currentSum = 0;
float filteredLux = 0;

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>ESP32 IoT Dashboard</title>
  <style>
    :root { --bg: #f4f7f6; --text: #2c3e50; --card: #ffffff; --accent:
#e65100; --border: #e2e8f0; }
    body.dark-mode { --bg: #1a202c; --text: #e2e8f0; --card: #2d3748; --
accent: #f6ad55; --border: #4a5568; }
    body { font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background-color: var(--bg); color: var(--text); margin: 0; padding: 20px;
transition: 0.3s; }
    .header { display: flex; justify-content: space-between; align-items:
center; margin-bottom: 20px; }
    .grid { display: grid; grid-template-columns: repeat(auto-fit,
minmax(300px, 1fr)); gap: 20px; }
    .card { background: var(--card); padding: 20px; border-radius: 12px; box-
shadow: 0 4px 6px rgba(0,0,0,0.1); border: 1px solid var(--border); position:
relative; }
    .lux-value { font-size: 3rem; font-weight: bold; color: var(--accent);
margin: 10px 0; }
    .stat-row { display: flex; justify-content: space-between; margin: 10px
0; padding-bottom: 10px; border-bottom: 1px solid var(--border); }
    button { background: var(--text); color: var(--bg); border: none;
padding: 10px 15px; border-radius: 6px; cursor: pointer; font-weight: bold;
transition: 0.2s; }
    button:hover { opacity: 0.8; }
    input[type=number] { padding: 8px; border-radius: 6px; border: 1px solid
var(--border); width: 80px; }
```

Продовження додатка А

```
.alarm { animation: blink 1s infinite; color: red !important; }
    @keyframes blink { 50% { opacity: 0.5; } }

    .chart-container { width: 100%; height: 200px; margin-top: 15px;
position: relative; }
    svg { width: 100%; height: 100%; stroke: var(--accent); fill: none;
stroke-width: 3; stroke-linecap: round; }
    .grid-line { stroke: var(--border); stroke-width: 1; stroke-dasharray: 4;
}
    .chart-label { font-size: 12px; fill: var(--text); stroke: none; }
  </style>
</head>
<body>
  <div class="header">
    <h2>Система моніторингу освітленості</h2>
    <button onclick="toggleTheme()">Dark/Light Mode</button>
  </div>

  <div class="grid">
    <div class="card" style="text-align: center;">
      <h3>Поточна освітленість</h3>
      <div class="lux-value"><span id="lux_val">--</span> lx</div>
      <p>Поріг тривоги: <input type="number" id="alarm_limit" value="50">
lx</p>
      <div id="alarm_msg" style="color: red; font-weight: bold; display:
none;">УВАГА: Рівень нижче норми!</div>
    </div>

    <div class="card">
      <h3>Аналітика сесії</h3>
      <div class="stat-row"><span>Максимум:</span> <strong
id="stat_max">0</strong></div>
      <div class="stat-row"><span>Мінімум:</span> <strong
id="stat_min">9999</strong></div>
      <div class="stat-row"><span>Середнє:</span> <strong
id="stat_avg">0</strong></div>
      <button onclick="exportCSV()" style="width: 100%; margin-top:
10px;">Завантажити звіт (CSV)</button>
    </div>

    <div class="card">
      <h3>Системна телеметрія</h3>
      <div class="stat-row"><span>Uptime:</span> <strong id="tel_uptime">0
c</strong></div>
      <div class="stat-row"><span>Wi-Fi RSSI:</span> <strong id="tel_rssi">0
dBm</strong></div>
      <div class="stat-row"><span>Free RAM:</span> <strong id="tel_heap">0
KB</strong></div>
    </div>
```

Продовження додатка А

```
</div>

<div class="card" style="margin-top: 20px;">
  <h3>Динаміка зміни освітленості (Real-time)</h3>
  <div class="chart-container">
    <svg id="svg_chart" viewBox="0 0 1000 200" preserveAspectRatio="none">
      <line x1="0" y1="50" x2="1000" y2="50" class="grid-line" />
      <line x1="0" y1="100" x2="1000" y2="100" class="grid-line" />
      <line x1="0" y1="100" x2="1000" y2="150" class="grid-line" />
      <path id="chart_path" d="" />
    </svg>
  </div>
</div>

<script>
  let luxHistory = [];
  let timeHistory = [];
  let minLux = 99999, maxLux = 0, sumLux = 0, count = 0;

  function toggleTheme() { document.body.classList.toggle('dark-mode'); }

  function updateSvgChart() {
    const pathEl = document.getElementById("chart_path");
    if (luxHistory.length < 2) return;

    let maxVal = Math.max(...luxHistory);
    let minVal = Math.min(...luxHistory);

    let margin = (maxVal - minVal) * 0.1;
    if (margin < 5) margin = 5;

    maxVal += margin;
    minVal -= margin;
    if (minVal < 0) minVal = 0;

    let range = maxVal - minVal;
    let points = [];
    let stepX = 1000 / (luxHistory.length - 1);

    for (let i = 0; i < luxHistory.length; i++) {
      let x = i * stepX;
      let y = 185 - ((luxHistory[i] - minVal) / range) * 170;
      points.push(`${x},${y}`);
    }

    pathEl.setAttribute("d", "M " + points.join(" L "));
  }
  setInterval(() => {
    fetch('/data')
```

Продовження додатка А

```
.then(response => response.json())
        .then(data => {
          let currentLux = data.lux;

          let luxEl = document.getElementById("lux_val");
          luxEl.innerText = currentLux;
          let limit = document.getElementById("alarm_limit").value;
          if(currentLux < limit) {
            luxEl.classList.add("alarm");
            document.getElementById("alarm_msg").style.display = "block";
          } else {
            luxEl.classList.remove("alarm");
            document.getElementById("alarm_msg").style.display = "none";
          }

          if(currentLux > maxLux) maxLux = currentLux;
          if(currentLux < minLux) minLux = currentLux;
          sumLux += currentLux; count++;
          document.getElementById("stat_max").innerText = maxLux;
          document.getElementById("stat_min").innerText = minLux;
          document.getElementById("stat_avg").innerText = (sumLux /
count).toFixed(1);

          document.getElementById("tel_uptime").innerText =
Math.floor(data.uptime / 1000) + " c";
          document.getElementById("tel_rssi").innerText = data.rssi + " dBm";
          document.getElementById("tel_heap").innerText = (data.heap /
1024).toFixed(1) + " KB";

          let now = new Date();
          let timeStr = now.getHours() + ":" + now.getMinutes() + ":" +
now.getSeconds();
          timeHistory.push(timeStr);
          luxHistory.push(currentLux);

          if (timeHistory.length > 30) {
            timeHistory.shift();
            luxHistory.shift();
          }
          updateSvgChart();
        }).catch(err => console.log("Помилка AJAX:", err));
    }, 2000);

    function exportCSV() {
      let csvContent = "data:text/csv;charset=utf-8,Time,Lux\n";
      for (let i = 0; i < timeHistory.length; i++) {
        csvContent += timeHistory[i] + "," + luxHistory[i] + "\n";
      }
      let encodedUri = encodeURI(csvContent);
```

Продовження додатка А

```
 let link = document.createElement("a");
      link.setAttribute("href", encodedUri);
      link.setAttribute("download", "lux_log.csv");
      document.body.appendChild(link);
      link.click();
    }
  </script>
</body>
</html>
)rawliteral";

float getLuxDirect() {
  Wire.beginTransmission(0x23);
  Wire.write(0x10);
  if (Wire.endTransmission() != 0) return -1;
  delay(185);

  Wire.requestFrom(0x23, 2);
  if (Wire.available() == 2) {
    uint8_t highByte = Wire.read();
    uint8_t lowByte = Wire.read();
    uint16_t raw = (highByte << 8) | lowByte;
    return raw / 1.2;
  }
  return -1;
}

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

  Wire.begin(16, 17);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("WiFi Connecting");

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

  lcd.clear();
  lcd.print("IP:");
  lcd.setCursor(0, 1);
  lcd.print(WiFi.localIP());

  server.on("/", []() { server.send(200, "text/html", index_html); });
  server.on("/data", []() {
```

Продовження додатка А

```
 String json = "{";
    json += "\"lux\":" + String(filteredLux, 1) + ",";
    json += "\"uptime\":" + String(millis()) + ",";
    json += "\"rssi\":" + String(WiFi.RSSI()) + ",";
    json += "\"heap\":" + String(ESP.getFreeHeap());
    json += "}";
    server.send(200, "application/json", json);
  });

  server.begin();
  delay(10000);
}

void loop() {
  server.handleClient();
  static unsigned long lastMeasure = 0;
  if (millis() - lastMeasure >= 2000) {

    Wire.end();
    delay(20);
    Wire.begin(21, 22);
    delay(20);

    float rawLux = getLuxDirect();

    if (rawLux >= 0) {
      currentSum -= buffer[bufferIndex];
      buffer[bufferIndex] = rawLux;
      currentSum += buffer[bufferIndex];
      bufferIndex = (bufferIndex + 1) % N;
      filteredLux = currentSum / N;
    }
    Wire.end();
    delay(20);
    Wire.begin(16, 17);
    delay(20);
    lcd.setCursor(0, 0);
    lcd.print("Lux Level:      ");
    lcd.setCursor(0, 1);
    if (rawLux < 0) {
      lcd.print("Sensor Error!   ");
    } else {
      lcd.print(filteredLux, 1);
      lcd.print(" lx        ");
    }

    lastMeasure = millis();
  }
}
```