Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СЕНСОРНОЇ СИСТЕМИ МОНІТОРИНГУ МІКРОКЛІМАТУ ДЛЯ АВТОМАТИЗОВАНОГО ДОГЛЯДУ ЗА РОСЛИНАМИ»

Виконав: студент IV курсу, групи АКІТР-23мс, спеціальності 174 – Автоматизація, комп'ютерно–інтегровані технології та робототехніка
Чумак В.В.
Керівник: к.т.н., доцент кафедри АІІТ
Богач І. В.
«16» 06 2025 р.
Рецензент: д.т.н., професор кафедри КН
Іванчук Я.В.
«17» 06 2025 р.

Допущено до захисту
завідувач кафедри АІІТ
«18» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність 174 – Автоматизація, комп'ютерно–інтегровані технології та робототехніка
Освітньо-професійна програма Інтелектуальні комп'ютерні системи управління

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
_____ д.т.н., проф. Олег БІСІКАЛО
«18» 06 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Чумаку Вадиму Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка інтелектуальної сенсорної системи моніторингу мікроклімату для автоматизованого догляду за рослинами»
керівник роботи Богач Ілона Віталіївна, к.т.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «20» березня 2025 року №97
2. Термін подання студентом роботи 10.06.2025 р.
3. Вихідні дані до роботи: Наявність мікроконтролера ESP32-H2-1-N2 із підтримкою Bluetooth Low Energy; Наявність сенсорів температури й вологості повітря (SHTC3) та вологості ґрунту (NE555DR); Наявність мобільного пристрою з ОС Android (версія 8+); Інсталяція мобільного застосунку для взаємодії з пристроєм (через BLE); Наявність середовищ розробки Visual Studio Code (з PlatformIO) та Android Studio.
4.Зміст текстової частини (перелік питань, які потрібно розробити) Огляд предметної області. Аналіз та обгрунтування вибору технологій. Реалізація програмної частини системи. Тестування системи та аналіз результатів.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) блок-схема алгоритму ініціалізації системи моніторингу; блок-схема алгоритму збору та обробки сенсорних даних; структурна схема інтелектуальної сенсорної системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Богач І.В., к.т.н., доцент кафедри АІІТ		

7. Дата видачі завдання 20.03.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Термін виконання		Примітка
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	03.10.2024	09.10.2024	Вик.
2	Аналіз предметної області та опрацювання літературних джерел	12.03.2025	29.03.2025	Вик.
3	Проєктування апаратної частини системи	01.04.2025	26.04.2025	Вик.
4	Розробка програмного забезпечення	29.04.2025	10.05.2025	Вик.
5	Тестування додатку та виправлення помилок	13.05.2025	24.05.2025	Вик.
7	Оформлення пояснювальної записки та графічної частини	27.05.2025	31.05.2025	Вик
8	Перевірка ПЗ на відповідність вимогам	03.06.2025	07.06.2025	Вик.
9	Попередній захист, доопрацювання, та рецензування БКР	10.06.2025	15.06.2025	Вик.
10	Захист БКР ЕК	16.06.2025	23.06.2025	Вик.

Студент

Керівник роботи

(підпис) Чумак В.В.
(прізвище та ініціали)

(підпис) Богач І.В.
(прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 84 сторінки формату А4, включаючи додатки, що містять 15 рисунків та 4 таблиці. Список використаних джерел налічує 24 найменувань.

Робота присвячена проєктуванню та реалізації інтелектуальної сенсорної системи моніторингу мікроклімату, орієнтованої на автоматизований догляд за рослинами. Основу апаратної частини становить мікроконтролер ESP32-H2-MINI, до якого підключено сенсори температури та вологості повітря (SHTC3) та вологості ґрунту (NE555DR). Передача даних здійснюється за допомогою Bluetooth Low Energy, що дозволяє користувачу зчитувати показники на мобільному пристрої.

У ході роботи було розроблено програмне забезпечення для збирання, обробки та передачі сенсорних даних, реалізовано енергоощадну логіку з використанням режиму глибокого сну, а також створено базову версію мобільного інтерфейсу. Запропонована система є гнучкою, економічною та придатною як для побутового, так і для промислового застосування.

Ключові слова: мікроконтролер ESP32, сенсорна система, BLE, мікроклімат, вологість ґрунту, моніторинг, автоматизація.

ABSTRACT

The bachelor’s thesis consists of 84 A4 pages including appendices with 15 figures and 4 tables. The list of references contains 24 items.

The work is dedicated to the design and implementation of an intelligent microclimate monitoring system for automated plant care. The hardware is based on the ESP32-H2-MINI microcontroller connected to SHTC3 (temperature and humidity) and NE555DR (soil moisture) sensors. Data transmission is organized via Bluetooth Low Energy, which allows the user to monitor parameters from a mobile device.

The developed firmware collects, filters and transmits sensor data, includes a deep sleep mode for power saving, and is integrated with a mobile application interface. The proposed system is flexible, low-cost and suitable for both domestic and professional use.

Keywords: ESP32 microcontroller, sensor system, BLE, microclimate, soil moisture, monitoring, automation.

ЗМІСТ

ВСТУП

Актуальність роботи Сучасні агротехнології дедалі частіше інтегрують інтелектуальні електронні системи для автоматизації процесів вирощування рослин, зокрема у сфері точного моніторингу мікроклімату. Ця тенденція обумовлена зростаючими потребами у продовольчій безпеці та необхідністю підвищення ефективності використання ресурсів в умовах обмеженості сільськогосподарських площ та зміни клімату.

За даними Міжнародної організації з продовольства та сільського господарства (FAO), світове населення зросте до 9,7 мільярда людей до 2050 року, що потребуватиме збільшення виробництва продовольства на 70%. В умовах зміни клімату та дефіциту водних ресурсів традиційні методи землеробства не можуть забезпечити такі потреби без радикального підвищення ефективності використання кожного квадратного метра посівних площ.

З розвитком доступних мікроконтролерних технологій та недорогих сенсорів з'явилась можливість точного регулювання умов вирощування відповідно до потреб конкретних культур та фаз їх розвитку. Сучасні IoT-технології дозволяють створювати автономні системи моніторингу, що значно перевершують традиційні методи контролю за ефективністю впливу на ростові процеси. Проте ефективне використання такої технології вимагає не лише впровадження окремих сенсорів, а й розробки комплексного апаратного та програмного забезпечення, здатного адаптуватися до змін зовнішніх умов.

Інтелектуальна система моніторингу мікроклімату повинна враховувати мультифакторний вплив температури повітря та її коливання протягом доби та сезонів, динаміку вологості ґрунту та її вплив на кореневу систему, контроль відносної вологості повітря для попередження грибкових захворювань, а також складні біологічні фази росту рослин — від проростання насіння до фази плодоношення. Така система має інтегрувати дані від множинних сенсорів у режимі реального часу для створення оптимальних умов вирощування при мінімальному втручанні людини.

У сучасних тепличних господарствах, вертикальних фермах та домашніх системах вирощування автоматичний моніторинг мікроклімату відіграє ключову роль у підвищенні врожайності та якості продукції. Статистичні дані провідних агропідприємств показують, що правильно налаштована система може підвищити врожайність на 15-25% при зниженні споживання води на 30-40% та енергоресурсів на 20-35%. Навпаки, неправильний контроль температурно-вологісного режиму може призвести до уповільнення росту, розвитку патогенної мікрофлори та навіть до загибелі рослин, що особливо критично в умовах комерційного виробництва.

Саме тому розробка автономної системи моніторингу мікроклімату з інтелектуальними алгоритмами обробки даних є надзвичайно актуальною: вона дозволить реалізувати функції безперервного контролю температури, вологості повітря та ґрунту з урахуванням історичних даних, прогнозування критичних ситуацій і встановлених технологічних карт вирощування. Така система повинна поєднувати високу функціональність з енергоефективністю та простотою використання, що робить її доступною для широкого кола користувачів від домашніх ентузіастів до комерційних виробників.

Актуальним є розробка автономної системи моніторингу мікроклімату для точного регулювання умов вирощування, що відповідає вимогам сучасного точного землеробства та сприяє демократизації передових агротехнологій.

Метою роботи є вдосконалення автоматизованого моніторингу мікроклімату в системах догляду за рослинами шляхом створення інтелектуальної сенсорної системи на основі мікроконтролера ESP32-H2-MINI-1-N2, що забезпечує автономне зчитування показників температури, вологості повітря і ґрунту з передачею даних через Bluetooth Low Energy на мобільний пристрій.

Для досягнення поставленої мети необхідно **розв'язати наступні задачі**:

1. Проаналізувати предметну область систем моніторингу мікроклімату та існуючі технічні рішення.

2. Розробити схемотехнічне рішення та друковану плату для інтеграції мікроконтролера ESP32-H2-MINI-1-N2 з сенсорами температури, вологості повітря та ґрунту.

3. Обрати та запрограмувати мікроконтролер, що забезпечує збір, обробку та передачу сенсорних даних з реалізацією енергоощадних алгоритмів.

4. Реалізувати базовий мобільний додаток для взаємодії з пристроєм через Bluetooth Low Energy протокол зв'язку.

5. Провести комплексне тестування системи в реальних умовах експлуатації та порівняльний аналіз з існуючими рішеннями.

Об'єктом дослідження є процес автоматизованого моніторингу мікроклімату в системах догляду за рослинами.

Предметом дослідження є апаратно-програмні засоби зчитування екологічних параметрів, алгоритми цифрової обробки сенсорних даних, методи енергоефективної передачі інформації у реальному часі та інтерфейси мобільного керування IoT-пристроями.

Методи дослідження У роботі застосовано системний аналіз існуючих аналогічних рішень, структурно-функціональне моделювання архітектури системи, схемотехнічне проєктування електронних вузлів, модульне програмування на мові C++ з використанням Arduino Framework, експериментальне тестування з використанням прецизійних цифрових сенсорів, статистичний аналіз точності вимірювань, а також порівняльні вимірювання в контрольованих та польових умовах експлуатації.

Основний науково-технічний результат полягає у створенні енергоефективної автономної системи моніторингу мікроклімату з інноваційним частотним методом вимірювання вологості ґрунту, що забезпечує тривалу автономну роботу та високу точність контролю параметрів навколишнього середовища для оптимізації умов вирощування рослин.

Практична цінність роботи полягає у створенні функціонального прототипу пристрою, який складається з мікроконтролера ESP32-H2-MINI-1-N2, високоточного сенсора температури та вологості SHTC3-TR-10KS,

оригінального частотного сенсора вологості ґрунту на основі NE555DR, інтелектуальної системи енергоменеджменту з Li-Ion акумулятором та надійної системи передавання даних через BLE. Розроблена прошивка забезпечує збір та фільтрацію сенсорних даних, їх передачу на мобільні пристрої і автоматичний перехід в енергозберігаючий режим для забезпечення автономної роботи протягом тижнів без обслуговування. Система поєднує доступну вартість компонентів, простоту виготовлення та можливість подальшого розширення функціоналу, що робить її придатною як для домашнього використання, так і для комерційного застосування в невеликих тепличних господарствах.

Апробація результатів дослідження

Результати роботи було представлено на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (2025) [1] та на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [2].

Впровадження результатів роботи. Результати роботи планується використати в розробках ТОВ «WDT» (Додаток Д).

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальна характеристика систем моніторингу мікроклімату

Системи моніторингу мікроклімату призначені для автоматизованого вимірювання та аналізу параметрів навколишнього середовища, що безпосередньо впливають на ріст і розвиток рослин. До таких параметрів зазвичай належать температура повітря, його вологість, вологість ґрунту, рівень освітлення, а інколи — й інші показники, як-от концентрація CO_2 або атмосферний тиск.

Ідея подібних систем полягає в тому, щоб забезпечити безперервний контроль за умовами вирощування рослин, виявляти відхилення від оптимальних значень та передавати відповідну інформацію користувачеві. За потреби система може ініціювати певні дії — наприклад, запуск поливу чи зміни режиму освітлення.

Зазвичай такі системи мають модульну структуру і складаються з:

– сенсорних вузлів для збору даних;

– мікроконтролерів для обробки інформації;

– комунікаційних інтерфейсів (дротових або бездротових);

– програмного забезпечення для зручного керування та візуалізації результатів.

Сучасні системи моніторингу мікроклімату знаходять широке застосування у різних галузях: від промислового тепличного господарства до домашнього садівництва. Особливо актуальними вони стають в умовах зміни клімату та необхідності оптимізації використання водних ресурсів.

1.2 Актуальні технічні рішення для мікрокліматичних систем

На сьогодні, найбільш ефективними з погляду енергоспоживання, точності вимірювань та зручності вбудованої логіки керування є мікроконтролери з

інтегрованою підтримкою бездротових протоколів. У цьому проєкті обрано ESP32-H2-MINI-1-N2 (рис.1.1) — компактний модуль, який має вбудовану підтримку Bluetooth Low Energy (BLE) та Zigbee, низьке енергоспоживання й можливість реалізації глибокого сну для тривалої автономної роботи.

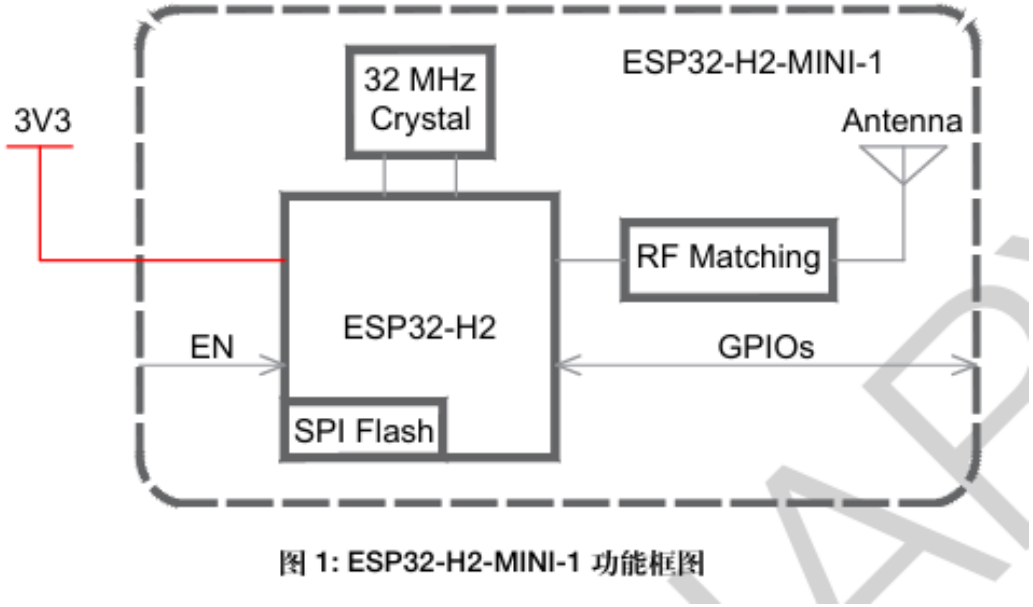

Рисунок 1.1 - Структурна схема ESP32-H2-MINI-1-N2

Для отримання мікрокліматичних показників використано такі сенсори:

– SHTC3-TR-10KS — цифровий сенсор температури та вологості повітря виробництва Sensirion. Забезпечує точність до ±0.2°C і ±2% RH відповідно, працює за протоколом I²C та має низький струм споживання. Взаємодія з ним реалізована за допомогою бібліотеки Adafruit_SHTC3.

– NE555DR — таймер, інтегрований у схему вимірювання вологості ґрунту. Він генерує імпульс, тривалість якого змінюється в залежності від електропровідності середовища між двома електродами. Зчитування сигналу виконується через функцію pulseIn(), що дозволяє перетворити аналогову характеристику ґрунту у цифрове значення. Принцип роботи частотного вимірювача вологості ґрунту представлено на рисунку 1.2.

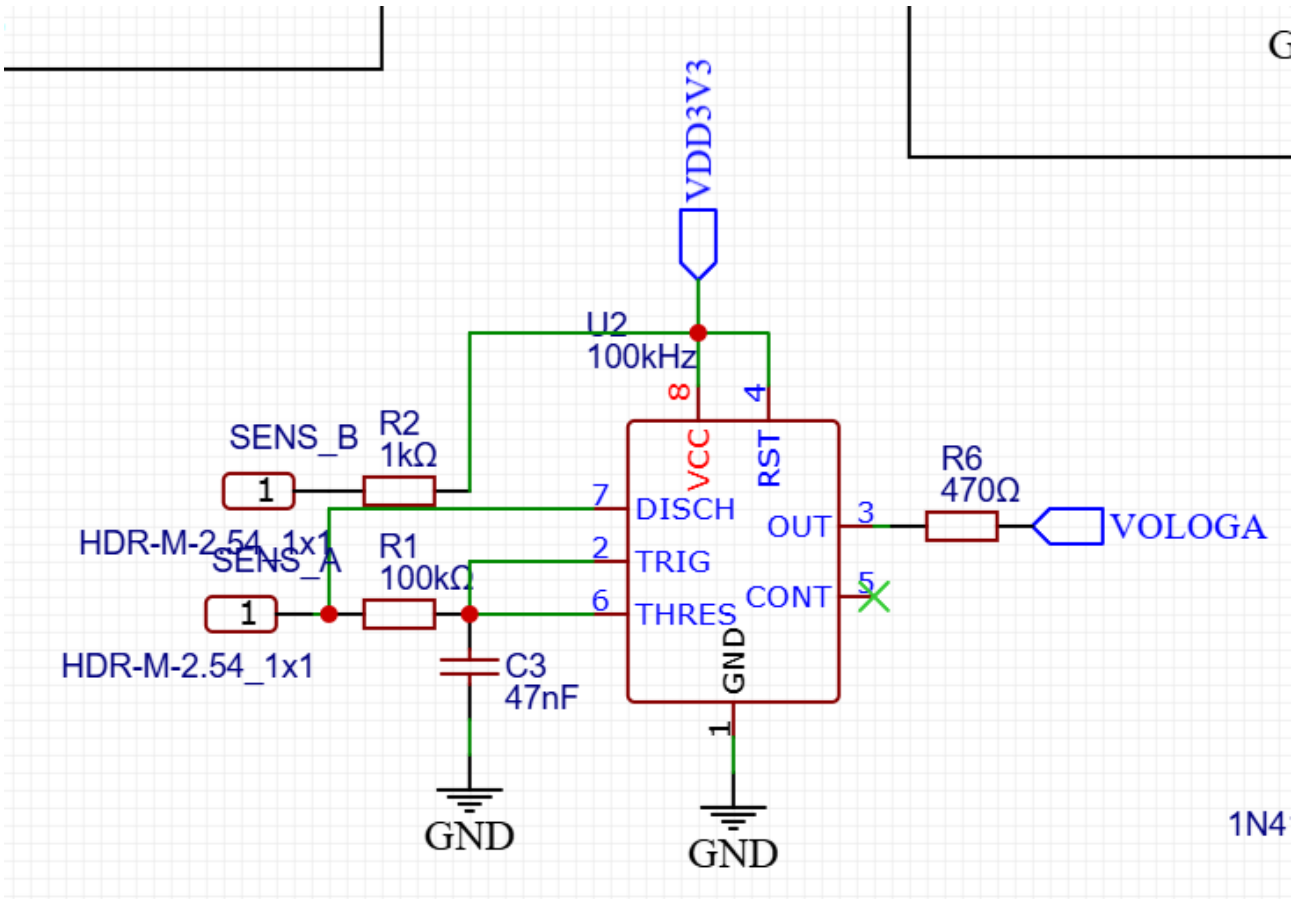

Рисунок 1.2 — Принцип роботи частотного вимірювача вологості ґрунту

— Аналоговий вхід мікроконтролера додатково використовується для контролю рівня напруги батареї (analogRead()), що є важливим для підтримки стабільної автономної роботи.

З метою передачі отриманих даних користувачу, система реалізує передавання параметрів через BLE з використанням власного модуля BLEManager, який забезпечує створення сервісу, характеристик, та їхнє оновлення у вигляді потокових повідомлень (notifications).

1.3 Аналіз існуючих комерційних рішень

Перед початком розробки власної системи було проведено детальний аналіз існуючих на ринку рішень для моніторингу мікроклімату. Це дозволило виявити основні тренди, переваги та недоліки сучасних систем, а також сформувати технічні вимоги до розроблюваного пристрою.

Xiaomi Mi Plant Flower Care — один з найпопулярніших побутових приладів для моніторингу стану рослин (рис.1.3). Пристрій здатен вимірювати вологість ґрунту, рівень освітленості, температуру навколишнього середовища та електропровідність ґрунту як показник концентрації поживних речовин. Підключення до смартфона здійснюється через Bluetooth, а управління — через власний мобільний додаток від Xiaomi.

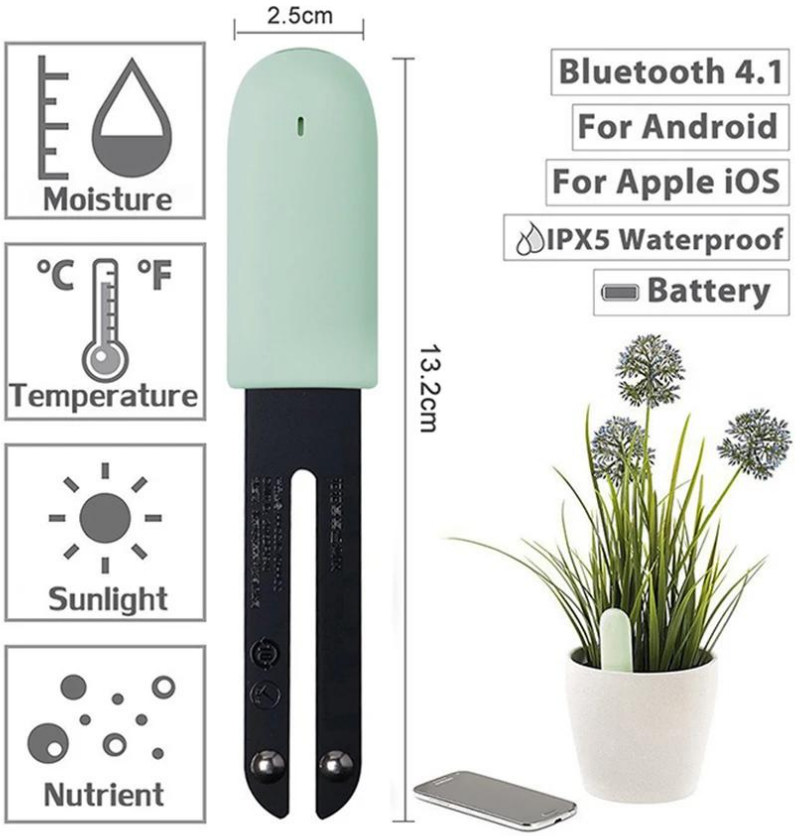

Рисунок 1.3 — Зовнішній вигляд Xiaomi Mi Plant Flower Care

Основні переваги цього рішення включають низьку вартість (близько $15-20), простоту використання, тривалу автономну роботу (до 1 року від однієї батарейки) та якісний мобільний додаток з великою базою даних різних видів рослин. Проте система має і суттєві недоліки: обмежений функціонал (відсутній контроль вологості повітря), необхідність фізичного наближення до пристрою для зчитування даних, відсутність можливості налаштування персональних сповіщень та обмежена дальність дії Bluetooth з'єднання.

SensorPush HT.w — Wi-Fi датчик температури та вологості з хмарним сервісом для зберігання історичних даних. Цей пристрій орієнтований на більш професійне використання і має кращі технічні характеристики порівняно з побутовими аналогами.

Його перевагами є Wi-Fi підключення для віддаленого доступу до даних, можливість налаштування різноманітних push-сповіщень при перевищенні встановлених порогів, зручна інтеграція з системами розумного дому та професійне програмне забезпечення з розширеними можливостями аналітики. Водночас недоліками є значно вища вартість ($30-50), залежність від стабільного Wi-Fi з'єднання та підвищене енергоспоживання через постійну активність Wi-Fi модуля. Порівняльна характеристика комерційних систем зображена на таблиці 1.1.

Таблиця 1.1 — Порівняльна характеристика комерційних систем моніторингу

Параметр	**Xiaomi Plant Care**	**SensorPush HT.w**	**Професійні системи**
Вартість	$15-20	$30-50	$500-5000
Автономність	До 1 року	2-3 місяці	Від мережі
Дальність зв'язку	10 м (BLE)	Необмежена (Wi-Fi)	Необмежена
Параметри	4	2	10+
Складність	Низька	Середня	Висока

Аналіз показав, що на ринку існує значний розрив між простими побутовими приладами та складними професійними системами. Побутові рішення мають обмежений функціонал і не завжди забезпечують необхідну точність та стабільність роботи в різних умовах експлуатації. Професійні системи, навпаки, мають надлишкову функціональність та неприйнятно високу вартість для більшості потенційних користувачів.

1.4 Виклики при реалізації автономної сенсорної системи

При створенні подібної системи розробник стикається з низкою об'єктивних труднощів, які необхідно вирішити для забезпечення стабільної роботи пристрою в реальних умовах експлуатації.

Автономність роботи. Пристрій має працювати тривалий час без підзарядки, що вимагає ретельної оптимізації енергоспоживання. Для цього реалізовано режим глибокого сну (Deep Sleep) з можливістю пробудження від зовнішнього сигналу (натискання кнопки) або за внутрішнім таймером через заданий інтервал часу.

Енергозбереження. Окрім використання режиму сну, важливо обмежити частоту оновлення даних до розумного мінімуму, своєчасно вимикати непотрібні периферійні модулі та використовувати бібліотеки, оптимізовані для роботи з архітектурою ESP32-H2. Також критично важливо правильно налаштувати всі таймери та переривання.

1.5 Сенсорна складова системи: принципи та особливості

Сенсорні системи моніторингу мікроклімату широко використовуються у різноманітних сферах діяльності людини та отримали особливе поширення там, де точний контроль параметрів навколишнього середовища має критичне значення для успішного результату. Найбільшого розвитку такі технології набули у промисловому тепличному господарстві, де навіть незначні відхилення температури або вологості можуть призвести до суттєвих втрат врожаю або погіршення якості продукції.

Крім того, сенсорні системи активно застосовуються на відкритому ґрунті для оптимізації роботи систем зрошення та точного землеробства, у спеціалізованих розсадниках і ботанічних садах для забезпечення індивідуальних оптимальних умов для різних видів та сортів рослин, а також

знаходять все більше застосування у сфері домашнього садівництва для догляду за кімнатними рослинами, вазонами та невеликими домашніми городами.

1.5.1 Принципи роботи цифрового сенсора SHTC3

Сенсор SHTC3-TR-10KS від швейцарської компанії Sensirion представляє собою висококласний приклад сучасних цифрових сенсорів, що поєднує в одному корпусі два незалежних вимірювальних елементи. Для визначення відносної вологості повітря використовується ємнісна технологія, заснована на зміні діелектричної проникності спеціального полімерного матеріалу при поглинанні молекул води з навколишнього повітря.

Принцип роботи ємнісного датчика вологості полягає в тому, що чутливий елемент являє собою плоский конденсатор, одна з обкладок якого покрита тонким шаром гігроскопічного діелектрика. При зміні вологості повітря цей матеріал поглинає або віддає молекули води, що призводить до зміни його діелектричної проникності та, відповідно, ємності конденсатора. Ця зміна ємності вимірюється високочутливою електронною схемою та перетворюється у цифрове значення відносної вологості. Будова та принцип роботи зображено на рисунку 1.4.

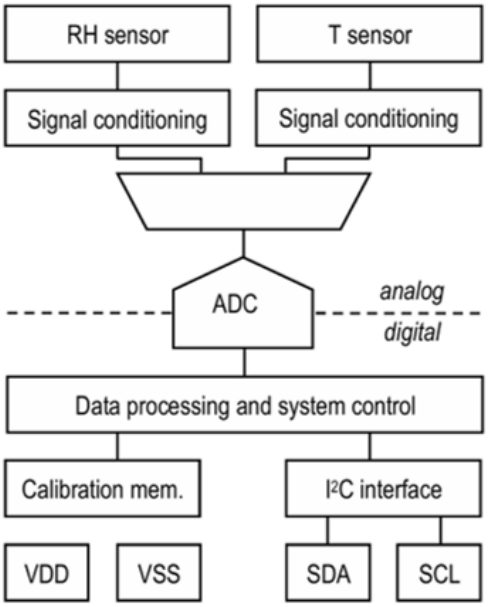

Рисунок 1.4 - Структурна схема сенсора SHTC3-TR-10KS

Для вимірювання температури в тому ж корпусі розміщено окремий терморезистивний елемент, виготовлений з матеріалу, електричний опір якого має чітко визначену залежність від температури. Використання платинового або кремнієвого терморезистора забезпечує високу лінійність характеристики та стабільність параметрів у широкому температурному діапазоні.

Основною технічною перевагою SHTC3 є повністю цифровий інтерфейс I²C, який дозволяє отримувати вже оброблені та калібровані значення температури і вологості без необхідності додаткової аналогової обробки сигналів з боку мікроконтролера. Це значно спрощує як апаратну, так і програмну реалізацію системи, а також мінімізує вплив електромагнітних завад на точність вимірювань.

1.5.2 Інноваційний підхід до вимірювання вологості ґрунту

Традиційні методи вимірювання вологості ґрунту мають ряд суттєвих недоліків, які обмежують їх практичне застосування в автономних системах довгострокового моніторингу. Найпоширеніші резистивні сенсори типу FC-28 або YL-69 працюють за принципом вимірювання електричного опору між двома електродами, зануреними в ґрунт. Проте цей метод має кілька критичних проблем.

По-перше, при проходженні постійного електричного струму через ґрунт відбувається електроліз, який призводить до поступового окислення електродів та зниження точності вимірювань з часом. По-друге, результати таких вимірювань сильно залежать не тільки від вологості, але й від концентрації розчинених у ґрунті солей, що робить калібрування складним і неточним. По-третє, метал електродів поступово розчиняється, що може негативно впливати на рослини.

Для вирішення цих проблем у розробленій системі реалізовано оригінальний частотний метод вимірювання вологості ґрунту на основі універсального таймера NE555DR. Цей підхід базується на принципово іншому

фізичному явищі — зміні діелектричної проникності середовища між електродами в залежності від вмісту вологи. (рис.1.5)

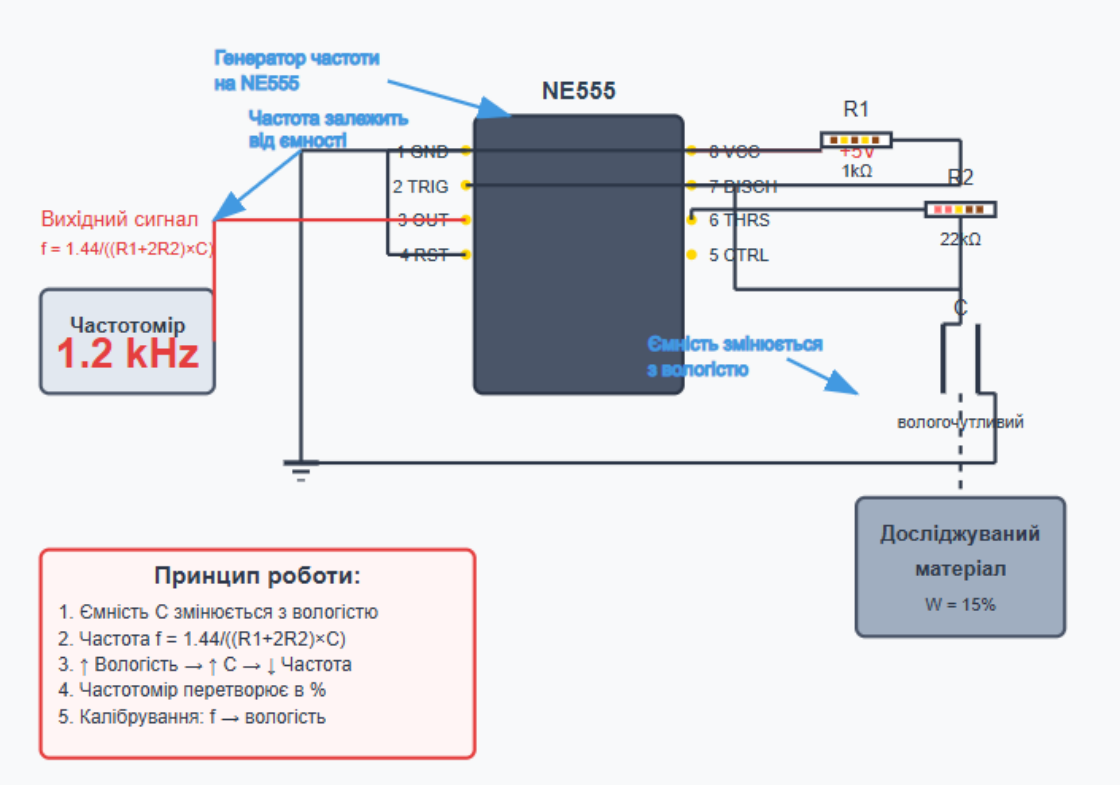

Рисунок 1.5 — Принцип роботи частотного методу вимірювання вологості

Мікросхема NE555 сконфігурована як астабільний мультивібратор, частота генерації якого визначається зовнішніми елементами RC-ланки. У якості змінного конденсатора виступає ємність між двома електродами, зануреними у ґрунт на певній відстані один від одного. При зміні вологості ґрунту змінюється його діелектрична проникність: сухий ґрунт має діелектричну проникність близько 3-5, тоді як для насиченого водою ґрунту цей показник може досягати 20-30.

Така суттєва зміна діелектричної проникності призводить до пропорційної зміни ємності між електродами, що в свою чергу впливає на частоту генерації мультивібратора. Мікроконтролер вимірює період генерованих імпульсів за допомогою високоточної функції pulseIn() і перетворює отримані значення у відсотки вологості за попередньо складеною калібрувальною таблицею.

1.5.3 Переваги частотного методу

Запропонований частотний метод має кілька важливих переваг порівняно з традиційними підходами:

— Відсутність електролізу. Оскільки через ґрунт не протікає постійний струм, електроліз практично не відбувається, що забезпечує тривалий термін служби електродів та стабільність характеристик протягом усього періоду експлуатації.

— Незалежність від мінералізації. Діелектрична проникність ґрунту значно менше залежить від концентрації розчинених солей порівняно з електричною провідністю, що робить калібрування більш стабільним та передбачуваним.

— Висока чутливість. Зміна ємності призводить до легко вимірюваних змін частоти, що дозволяє досягти високої роздільної здатності вимірювань навіть при невеликих змінах вологості.

— Економічність. Використання доступної мікросхеми NE555 та простих пасивних компонентів робить такий сенсор значно дешевшим за готові ємнісні аналоги.

— Можливість калібрування. Програмне калібрування дозволяє адаптувати систему під різні типи ґрунту без зміни апаратної частини.

1.5.4 Система контролю енергопостачання

Окремим важливим елементом сенсорної підсистеми є контроль стану джерела живлення. Для автономних пристроїв моніторинг напруги акумулятора є критично важливим, оскільки дозволяє не лише своєчасно попередити користувача про необхідність підзарядки, але й реалізувати інтелектуальні алгоритми енергоменеджменту.

У розробленій системі контроль напруги акумулятора здійснюється через спеціально підключений канал вбудованого аналого-цифрового перетворювача (АЦП) мікроконтролера. Напруга акумулятора подається на вхід АЦП через

прецизійний резистивний дільник з коефіцієнтом поділу 2:1, що дозволяє безпечно вимірювати напругу в повному робочому діапазоні Li-Ion акумулятора від 3.0В до 4.2В. Характеристику Li-Ion акумулятора зображено на рисунку 1.6.

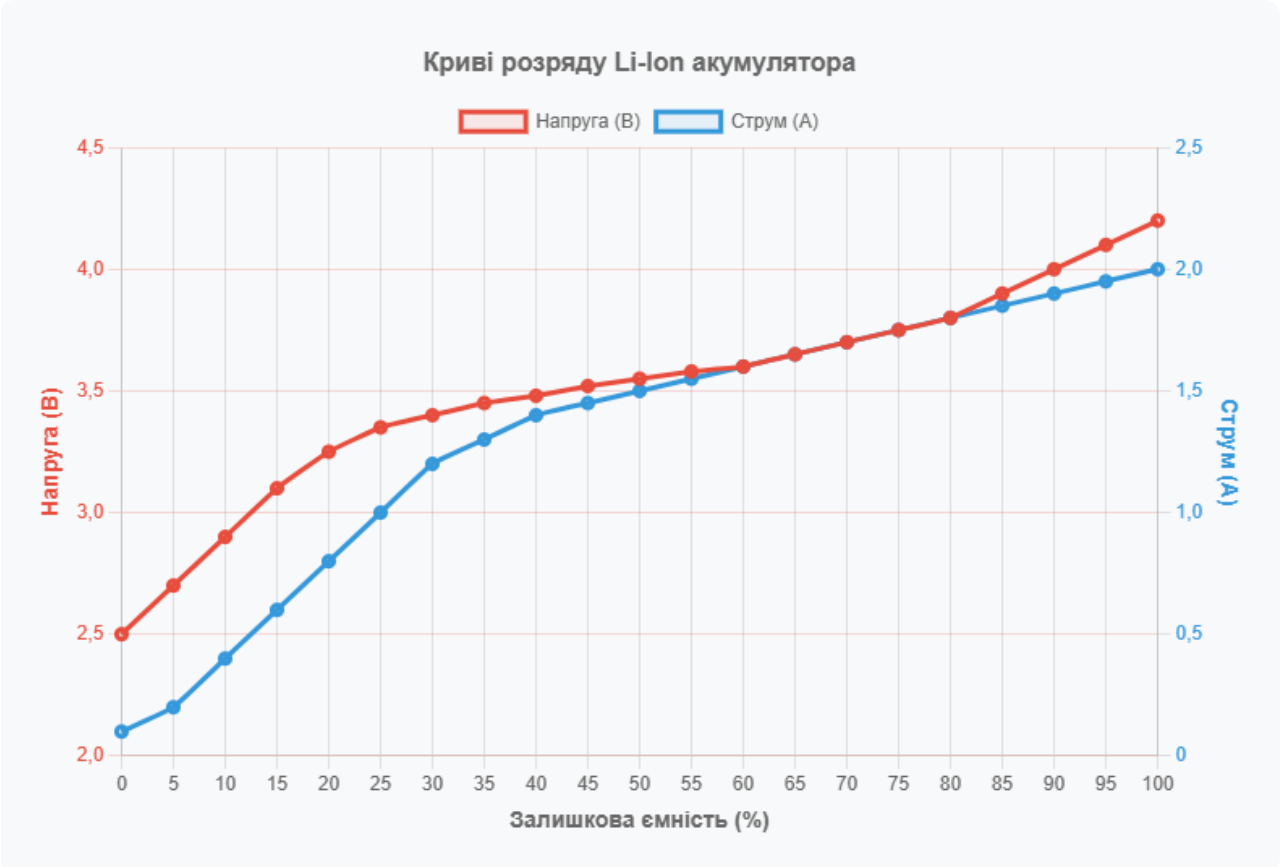

Рисунок 1.6 — Характеристика розряду Li-Ion акумулятора

Ця інформація використовується не лише для інформування користувача, але й для автоматичного переведення системи у режим максимального енергозбереження при критично низькому заряді. У такому режимі інтервали між вимірюваннями збільшуються, BLE-передача мінімізується, а деякі несуттєві функції тимчасово вимикаються для продовження роботи.

Алгоритм оцінки рівня заряду базується на відомій залежності напруги Li-Ion акумулятора від ступеня його розряду. При напрузі 4.1-4.2В акумулятор вважається повністю зарядженим (100%), при 3.7В — наполовину розрядженим (50%), при 3.3В залишається близько 10% заряду, а напруга нижче 3.0В свідчить про критичний розряд і необхідність негайної зарядки.

1.6 Структура системи та її функціональні блоки

Розроблена система має модульну архітектуру, що дозволяє легко модифікувати окремі компоненти без впливу на роботу всієї системи. Структуру системи та призначення її функціональних блоків наведено в таблиці 1.2.

Таблиця 1.2 — Структура системи та її функціональні блоки

Компонент	**Функціональне призначення**
ESP32-H2-MINI-1-N2	Центральний мікроконтролер, що керує всією логікою пристрою та координує роботу всіх підсистем
SHTC3-TR-10KS	Високоточне цифрове вимірювання температури та відносної вологості повітря з інтерфейсом I²C
NE555DR + логіка	Генератор для імпульсного визначення вологості ґрунту за частотним методом
ADC (вбудований)	Контроль рівня заряду акумулятора через резистивний дільник напруги
BLEManager (програмний)	Модуль передачі даних через Bluetooth Low Energy з підтримкою GATT-сервісів
Deep Sleep система	Програмно-апаратне енергозбереження при тривалій бездіяльності
Wake-up контролер	Система пробудження з глибокого сну при натисканні кнопки або за таймером
Холдер акумулятора 18650	Механічне кріплення та електричне підключення Li-Ion акумулятора

1.7 Постановка задачі та технічні вимоги

У рамках дипломного проєкту передбачається розробка інтелектуальної сенсорної системи, здатної в автоматичному режимі здійснювати моніторинг основних мікрокліматичних показників, таких як температура повітря, його

вологість та вологість ґрунту. Система повинна бути енергоефективною, надійною в експлуатації, масштабованою для майбутніх доповнень та зручною у використанні кінцевим споживачем.

З урахуванням особливостей предметної області та результатів аналізу сучасних технічних рішень було сформульовано наступні детальні технічні вимоги до системи:

— необхідно забезпечити повну апаратну та програмну інтеграцію сенсорів температури й вологості повітря (SHTC3) та вологості ґрунту (на базі NE555) з мікроконтролером ESP32-H2-MINI-1-N2;

— потрібно реалізувати надійні алгоритми стабільного збору та попередньої обробки даних від сенсорних елементів, включаючи усереднення вимірювань та цифрову фільтрацію шумів (особливо критично для аналогових сигналів);

— важливо передбачити систему сповіщення про критичні зміни параметрів середовища, наприклад — при критичному пересиханні ґрунту або екстремальних температурах;

— система повинна забезпечувати стабільну передачу поточних значень параметрів до мобільного пристрою користувача за допомогою Bluetooth Low Energy (BLE) з підтримкою стандартних GATT-сервісів;

— необхідно реалізувати ефективну систему автономної роботи, включаючи періодичний перехід у режим глибокого сну з подальшим автоматичним пробудженням по внутрішньому таймеру або зовнішньому сигналу (кнопка користувача);

— з точки зору користувача передбачається розробка функціонального мобільного інтерфейсу, який дозволятиме переглядати актуальні показники в зручному форматі, змінювати базові налаштування роботи системи, а в перспективі — отримувати історичні графіки та персоналізовані повідомлення;

— архітектура програмного та апаратного рішення повинна бути модульною, з можливістю подальшого розширення системи новими типами сенсорів або виконавчими елементами (насоси для поливу, системи освітлення, реле

керування) без необхідності кардинальної зміни існуючого коду або принципової схеми.

Уся реалізація має враховувати специфіку практичного застосування в умовах теплиць, парників або побутового середовища — тобто орієнтуватися на мінімальне технічне обслуговування, високу стабільність роботи в часі та здатність адаптуватися до мінливих зовнішніх умов експлуатації.

2 ПРОЄКТУВАННЯ ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

2.1 Підхід до розробки програмного забезпечення в сенсорних системах

Програмне забезпечення відіграє центральну роль у функціонуванні будь-якої сенсорної системи моніторингу. У випадку розробки інтелектуального моніторингу мікроклімату програмна частина виконує одразу кілька критично важливих функцій: безперервне зчитування даних з фізичних сенсорів, їх математичну обробку та фільтрацію для усунення шумів, надійну передачу інформації на мобільні пристрої користувачів, а також інтелектуальне управління енергоспоживанням системи для забезпечення максимальної автономності.

Тому ще на самому початковому етапі планування критично важливо правильно визначити, які саме інструменти розробки та технологічні підходи будуть використовуватись. При цьому необхідно обов'язково враховувати специфічні обмеження мікроконтролерних систем: обмежений обсяг оперативної пам'яті, невисоку тактову частоту процесора, необхідність забезпечення автономності роботи та високі вимоги до точності вимірювань у реальних умовах експлуатації.

Особливістю даного проєкту є використання відносно нового мікроконтролера ESP32-H2-MINI-1-N2, який відрізняється наявністю вбудованої підтримки Bluetooth Low Energy та можливістю використання глибоких енергоощадних режимів. Це значною мірою впливає на вибір програмних бібліотек, мови програмування та архітектурних підходів до побудови програмної системи в цілому.

2.2 Вимоги до програмного забезпечення

З урахуванням функціонального призначення розроблюваної системи та результатів аналізу аналогічних рішень було сформовано наступний перелік детальних вимог до програмної складової:

— забезпечити стабільне та точне зчитування даних з цифрового сенсора температури та вологості повітря (SHTC3), а також з аналогового сенсора вологості ґрунту (NE555DR), із подальшою математичною обробкою отриманих сигналів;

— реалізувати ефективні алгоритми цифрової фільтрації шумів та статистичного усереднення показників, що особливо важливо для аналогового сигналу з NE555, який має імпульсну природу та може містити значні флуктуації;

— здійснювати надійну передачу оброблених даних через BLE-інтерфейс, з дотриманням оптимальної частоти оновлення та правильної структури GATT-повідомлень для забезпечення сумісності з різними мобільними пристроями;

— безперервно контролювати рівень заряду акумулятора за допомогою вбудованого АЦП через функцію analogRead() та у разі критичного зниження напруги живлення автоматично інформувати користувача через мобільний додаток;

— підтримувати максимально енергоефективний режим роботи, зокрема грамотне використання функції esp_deep_sleep_start() з можливістю програмованого пробудження від кнопки користувача або за внутрішнім таймером через заданий інтервал;

— забезпечити архітектурну масштабованість системи, тобто можливість у майбутньому додати нові типи сенсорів або керуючі модулі без необхідності повної переробки існуючої прошивки мікроконтролера;

— передбачити створення зручного мобільного інтерфейсу користувача, що дозволятиме в реальному часі переглядати актуальні показники всіх сенсорів, налаштовувати індивідуальні пороги спрацьовування сповіщень та отримувати системні повідомлення про стан пристрою.

2.3 Обґрунтування вибору середовища розробки

У рамках реалізації прошивки було використано ряд бібліотек, кожна з яких відповідає за окремий аспект системи:

— Adafruit_SHTC3 бібліотека для зчитування температури та вологості повітря з сенсора SHTC3 через I²C. Забезпечує зручний доступ до цифрових значень без необхідності самостійної реалізації протоколу.

— NimBLE-Arduino полегшена й оптимізована реалізація Bluetooth Low Energy для ESP32, яка використовується в модулі BLEManager. Завдяки ній реалізовано створення GATT-сервісів, характеристик, та надсилання повідомлень з даними у режимі notify.

— driver/adc.h низькорівневий драйвер, який використовується для зчитування аналогової напруги батареї. Важливий для реалізації інформування користувача про стан живлення.

— ESP32 sleep API набір функцій із SDK для реалізації переходу в глибокий сон з пробудженням від GPIO (наприклад, кнопка) або по таймеру. Дає змогу суттєво знизити споживання енергії.

— Власноруч написані функції для обробки pulseIn() з NE555, з калібруванням та перетворенням у відносну шкалу вологості.

2.4 Вибір технологій для мобільного застосунку

У межах даного проєкту планується базова реалізація мобільного застосунку з основними функціями відображення даних, зчитування BLE-характеристик і початкової обробки отриманої інформації. На даному етапі розробки мобільний додаток знаходиться у стадії активного тестування, і хоча основний функціонал вже працює, іноді виникають технічні проблеми, які потребують подальшого вирішення.

Для розробки клієнтської частини обрано сучасне середовище Android Studio з використанням мови програмування Kotlin та актуальних бібліотек екосистеми Android:

androidx.bluetooth — офіційна бібліотека Google для роботи з Bluetooth Low Energy в операційній системі Android, яка забезпечує стандартизований доступ до GATT-сервісів та характеристик;

Jetpack Compose — сучасний декларативний UI toolkit для створення нативного користувацького інтерфейсу з підтримкою Material Design принципів;

LiveData / ViewModel — компоненти Android Architecture Components для реалізації реактивного оновлення показників на екрані при отриманні нових даних від BLE-пристрою;

Room або SharedPreferences — системи для локального зберігання останніх значень сенсорів або накопичення історичних даних для побудови графіків.

Слід зазначити, що робота над мобільним додатком ще триває, і на поточному етапі основна увага приділяється стабілізації BLE-з'єднання та усуненню періодичних збоїв у отриманні даних від пристрою.

На рисунку 2.1 зображено архітектуру мобільного застосунку.

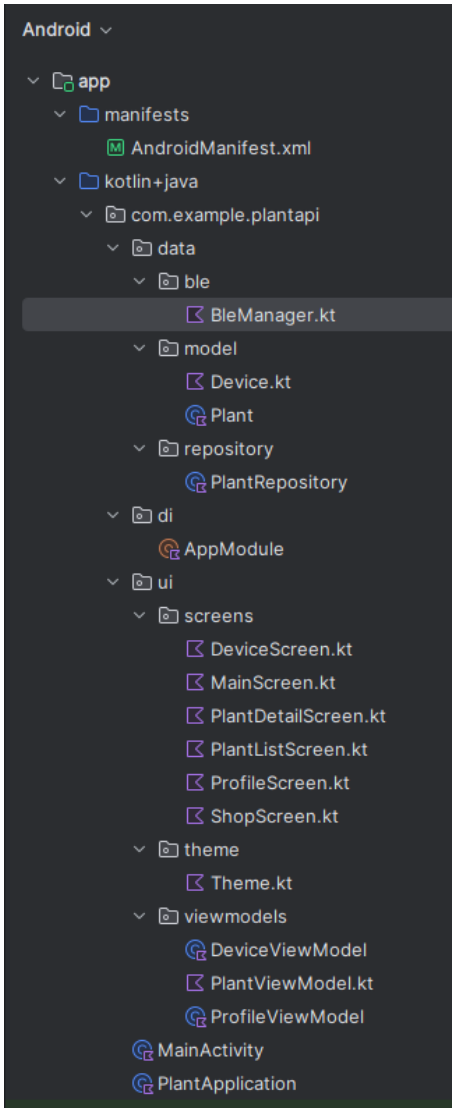

Рисунок 2.1 — Архітектура мобільного додатку

2.5 Порівняння альтернативних варіантів програмного забезпечення

Для обґрунтування остаточного вибору технологічного стеку було проведено детальне порівняння основних альтернативних варіантів програмного забезпечення для розробки прошивки мікроконтролера. Результати аналізу наведено в таблиці 2.1.

Таблиця 2.1 - Порівняння альтернативних варіантів програмного забезпечення

Критерій	Arduino C++	MicroPython	ESP-IDF C
Простота розробки	Висока	Висока	Низька
Продуктивність виконання	Висока	Середня	Максимальна
BLE / Zigbee підтримка	Повна через бібліотеки	Обмежена	Максимально повна
Енергозбереження	Добра оптимізація	Слабкі можливості	Максимально ефективна
Спільнота / документація	Дуже розвинена	Помірно розвинена	Професійна
Час розробки	Короткий	Короткий	Довгий
Налагодження	Просте	Середнє	Складне

З огляду на те, що даний проєкт має помірну технічну складність, але при цьому потребує високої стабільності роботи, швидкого прототипування та можливості легкого розширення функціоналу в майбутньому, остаточно обраний варіант на основі Arduino C++ з використанням бібліотеки NimBLE представляє оптимальний баланс між простотою розробки та функціональними можливостями.

Цей вибір також обґрунтований наявністю великої кількості готових та протестованих бібліотек для роботи з різними типами сенсорів, що значно прискорює процес розробки та зменшує ймовірність виникнення помилок у критично важливих частинах коду.

2.6 Принципова схема та конструкція пристрою

2.6.1 Електрична принципова схема системи

На рисунку 2.2 наведено повну електричну принципову схему розробленого пристрою моніторингу мікроклімату. Схема реалізована у середовищі EasyEDA та включає всі необхідні компоненти для забезпечення функціональності системи.

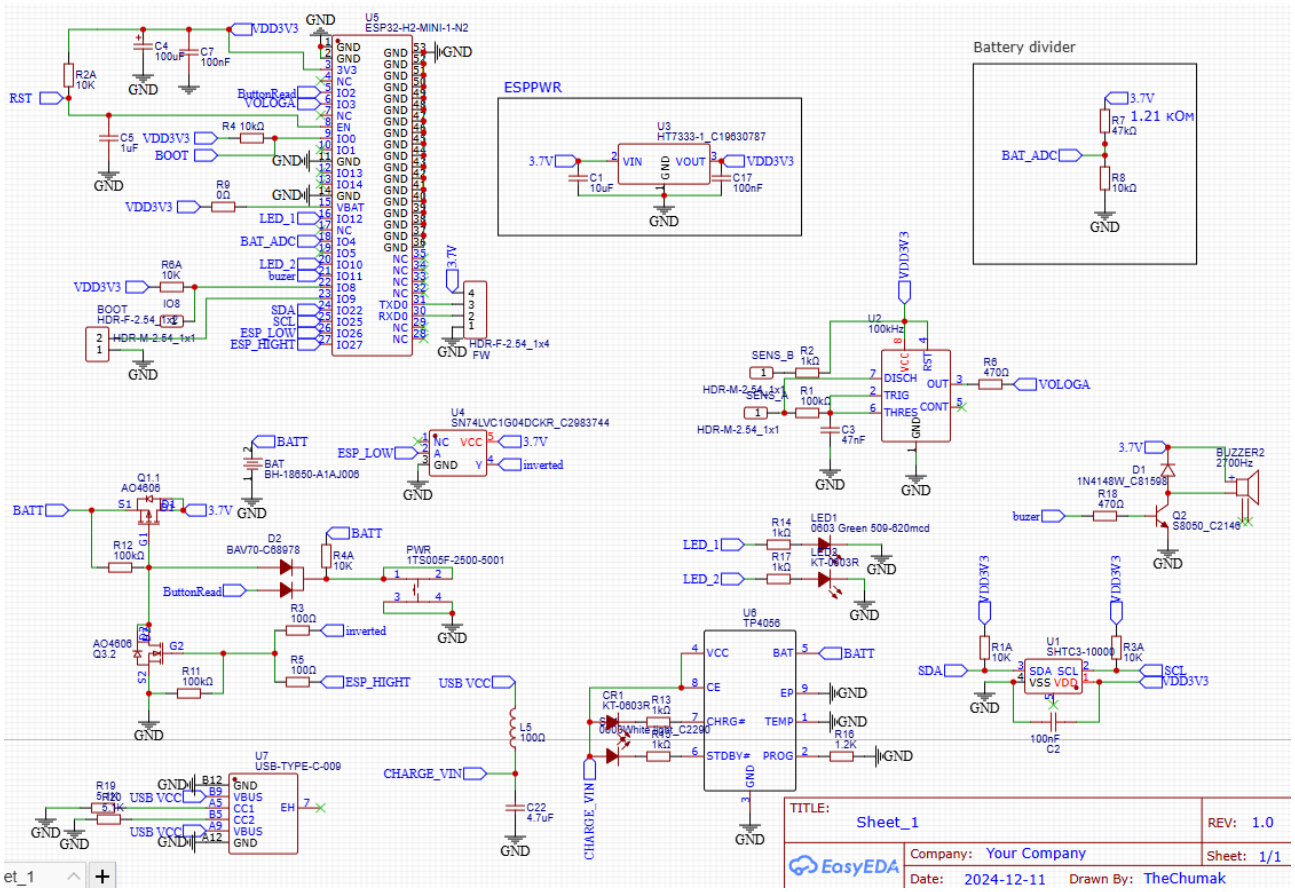

Рисунок 2.2 — Принципова електрична схема пристрою

Основними функціональними блоками схеми є:

— Центральний мікроконтролер U2 (ESP32-H2-MINI-1-N2) — виконує всі обчислювальні операції, керує сенсорами, реалізує BLE-зв'язок та енергоменеджмент. Мікроконтролер підключений через стандартні виводи живлення, землі та програмування.

— Блок живлення та зарядки — включає зарядний контролер TP4056 (U7) для безпечної зарядки Li-Ion акумулятора, стабілізатор напруги для живлення цифрових компонентів, та схему контролю рівня заряду батареї.

— Сенсор температури та вологості U4 (SHTC3-TR-10KS) — підключений до мікроконтролера через інтерфейс I²C (виводи SDA/SCL). Забезпечує високоточне вимірювання температури повітря (±0.2°C) та відносної вологості (±2% RH).

— Схема вимірювання вологості ґрунту на базі U5 (NE555DR) — реалізує частотний метод вимірювання через генерацію імпульсів, частота яких залежить від діелектричної проникності ґрунту між електродами E1 та E2.

— Дільник напруги для контролю батареї — резистори R6 та R7 формують дільник з коефіцієнтом 1:2 для безпечного вимірювання напруги Li-Ion акумулятора через ADC мікроконтролера.

2.6.2 Конструктивне виконання пристрою

Пристрій виконано у компактному корпусі розмірами 80×60×25 мм, що забезпечує зручність розміщення поряд з рослинами. Конструкція передбачає:

— Основна друкована плата — двошарова плата розміром 70×50 мм з розміщенням всіх електронних компонентів на верхній стороні для спрощення монтажу та обслуговування.

— Датчик вологості ґрунту — виконаний у вигляді окремої невеликої плати з двома електродами, що з'єднується з основною платою гнучким кабелем довжиною 200 мм.

— Акумуляторний відсік — стандартний тримач для Li-Ion акумулятора типу 18650 ємністю 2600-3400 mAh, розрахований на тривалу автономну роботу.

— Елементи індикації — світлодіод стану системи (живлення, зарядка, помилки) та кнопка примусового пробудження з режиму глибокого сну.

— Роз'єм зарядки — стандартний USB-C для зручної зарядки акумулятора без необхідності його вилучення з пристрою.

2.6.3 Розведення друкованої плати

На рисунку 2.3 показано топологію розведення друкованої плати пристрою, виконану в середовищі EasyEDA. Плата має розміри 210×22 мм і виконана за двошаровою технологією.

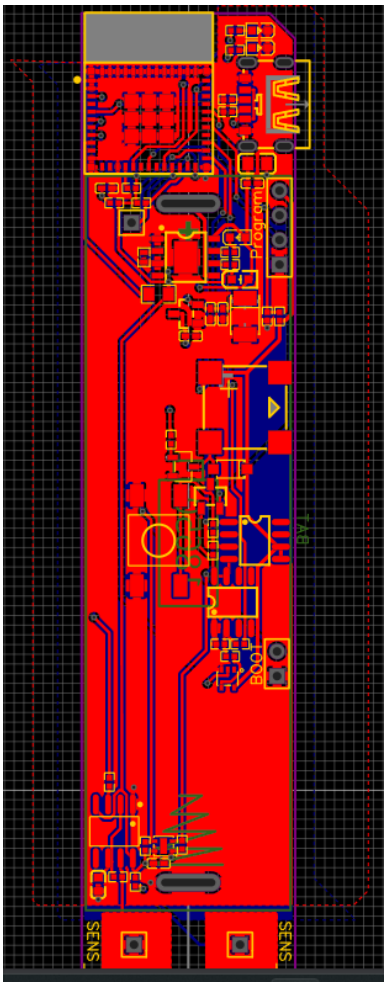

Рисунок 2.3 — Топологія друкованої плати (червоний шар — верхній, синій шар — нижній)

Особливості розведення:

— Верхній шар (червоний) — розміщено основні сигнальні провідники, живлення аналогових компонентів. Передбачено мінімальні довжини з'єднань для зменшення паразитних наводок.

— Нижній шар (синій) — виконано полігон землі (GND) та основні шини живлення. Це забезпечує низький опір по постійному струму та ефективне екранування від електромагнітних завад.

— Термічні переходи — для компонентів з підвищеним тепловиділенням (зарядний контролер, стабілізатор) передбачено додаткові металізовані отвори для відводу тепла.

На рисунку 2.4 наведено 3D-візуалізацію друкованої плати з встановленими компонентами.

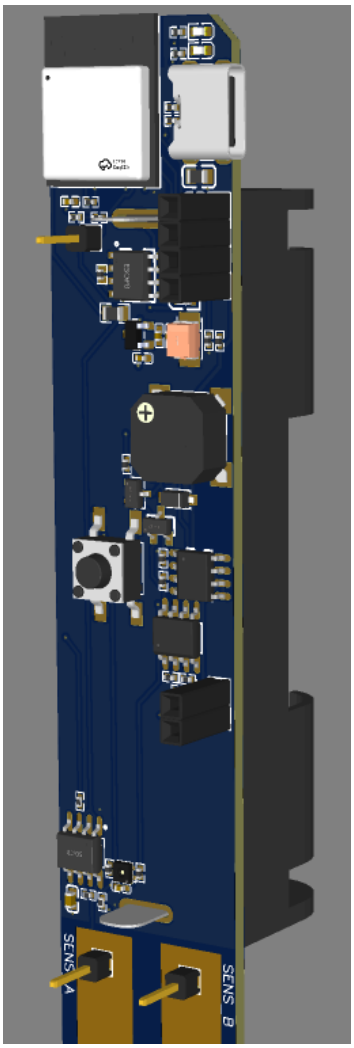

Рисунок 2.4 — 3D-модель друкованої плати з компонентами

Компонування елементів:

— ESP32-H2-MINI-1-N2 розміщено в центральній частині плати для мінімізації довжин з'єднань з периферійними компонентами.

— Сенсор SHTC3 встановлено на краю плати для забезпечення хорошого контакту з навколишнім повітрям та точності вимірювань.

— Зарядний контролер TP4056 розміщено поблизу USB-роз'єму для мінімізації довжин силових провідників.

— RC-компоненти згруповано поблизу відповідних мікросхем для зменшення паразитних впливів та покращення стабільності роботи.

2.6.4 Особливості схемотехнічних рішень

Енергоефективність — в схемі передбачено можливість програмного відключення живлення аналогових компонентів під час режиму сну для мінімізації споживання струму.

Завадостійкість — всі аналогові входи захищені RC-фільтрами, цифрові лінії мають pull-up резистори для забезпечення стабільної роботи в умовах електромагнітних завад.

Температурна стабільність — використано компоненти з розширеним температурним діапазоном (-20°C до +70°C) для надійної роботи в різних кліматичних умовах.

Масштабованість — на платі передбачено додаткові контактні площадки для підключення майбутніх сенсорів або виконавчих пристроїв без зміни базової конструкції.

Запропонована конструкція забезпечує оптимальний баланс між функціональністю, енергоефективністю та вартістю виготовлення, що робить її придатною як для прототипування, так і для серійного виробництва.

2.7 Енергетичний аналіз та оптимізація

2.7.1 Детальний аналіз споживання компонентів

Енергоефективність є критично важливим параметром для автономних IoT-пристроїв, оскільки безпосередньо впливає на тривалість роботи без підзарядки та практичність використання системи. Для оптимізації енергоспоживання було проведено детальний аналіз всіх компонентів системи в різних режимах роботи.

Мікроконтролер ESP32-H2-MINI-1-N2 є основним споживачем енергії в системі, тому його оптимізація має першочергове значення. Проведені вимірювання показали наступні характеристики споживання в активному режимі роботи. Базове споживання процесора при частоті 96 МГц складає 65 мА, BLE-модуль в активному стані споживає додаткові 15 мА, що дає загальне споживання в активному режимі 80 мА при напрузі 3,3 В.

Режими енергозбереження мікроконтролера характеризуються значно нижчим споживанням. У режимі Light Sleep споживання становить 0,8 мА при збереженні оперативної пам'яті та вимкненні WiFi/BLE модулів. Режим Deep Sleep забезпечує споживання лише 10 мкА при збереженні тільки RTC RAM. Найбільш економний режим Hibernation характеризується споживанням 2,5 мкА з повним відключенням усіх функцій окрім підтримки часу реального часу.

Особливості споживання в BLE-специфічних режимах також потребують детального аналізу. Режим Advertising з інтервалом 100 мс характеризується середнім споживанням 12 мА. Connected mode при активній передачі даних споживає 25 мА, тоді як у стані очікування споживання знижується до 0,5 мА.

Сенсор SHTC3-TR-10KS має оптимізовані характеристики споживання для автономних застосувань. У режимі активного вимірювання сенсор споживає 0,4 мА протягом 10 мс, а в режимі очікування споживання знижується до 0,01 мкА. Струм витоку становить менше 0,001 мкА, а енергія на одне вимірювання складає 1,2 мкДж.

Частотний сенсор вологості ґрунту на основі мікросхеми NE555DR в активному режимі генерації споживає 3 мА при напрузі 3,3 В. Важливою особливістю конструкції є можливість повного програмного відключення живлення, що дозволяє досягти нульового споживання між вимірюваннями. Струм витоку через елементи RC-ланцюга становить лише 0,003 мА.

Резистивний дільник для контролю стану батареї створює постійне навантаження через резистори R6-R7 номіналом 2×10 кОм, що відповідає струму 0,21 мА. Споживання аналого-цифрового перетворювача під час вимірювання становить 0,1 мА протягом 1 мс, що має мінімальний вплив на загальну автономність системи.

Додаткові компоненти системи також вносять свій вклад у загальне енергоспоживання. Світлодіодна індикація при активації споживає 2 мА, зарядний контролер TP4056 в режимі очікування споживає 50 мкА, а струми витоку друкованої плати та з'єднань не перевищують 5 мкА.

2.7.2 Стратегії енергозбереження

На основі аналізу споживання компонентів було розроблено комплекс стратегій для максимізації автономності роботи системи.

Адаптивні інтервали вимірювань забезпечують оптимальне балансування між якістю моніторингу та енергоспоживанням. Традиційний підхід з фіксованими інтервалами не є оптимальним з точки зору енергоефективності. Розроблено алгоритм адаптивного налаштування частоти вимірювань залежно від динаміки зміни контрольованих параметрів.

Алгоритм працює за наступним принципом. При стабільних умовах, коли зміна параметрів не перевищує 2 відсотки за годину, інтервал між вимірюваннями збільшується до 60 хвилин. При помірних змінах в діапазоні 2-5 відсотків за годину використовується стандартний інтервал 15 хвилин. При швидких змінах понад 5 відсотків за годину застосовується скорочений інтервал

5 хвилин. У випадку критичних значень параметрів активується екстрений режим з інтервалом вимірювань 2 хвилини.

Ця стратегія дозволяє зменшити середнє споживання енергії на 35-50 відсотків без втрати якості моніторингу мікрокліматичних параметрів.

Динамічне керування живленням периферійних пристроїв реалізовано через програмне відключення найбільш енергоємних компонентів. Частотний генератор вологості ґрунту споживає 3 мА постійно, що становить значну частку від загального споживання системи. Для вирішення цієї проблеми реалізовано схему програмного відключення через MOSFET-транзистор, що дозволяє повністю вимикати генератор між вимірюваннями.

Для роботи з сенсором SHTC3 використовується підвищена частота інтерфейсу I²C. Частота 400 кГц замість стандартних 100 кГц зменшує час активної роботи на 60 відсотків, що сприяє загальному енергозбереженню.

Оптимізація BLE-комунікації здійснюється через адаптивне керування рекламними пакетами. Інтервал BLE-реклами динамічно змінюється залежно від наявності підключених пристроїв. Без активних підключень використовується інтервал 2 секунди для економії енергії. При пошуку з'єднання інтервал скорочується до 100 мс для забезпечення швидкого підключення. За наявності активного з'єднання рекламу повністю вимикається.

Додатково реалізовано оптимізацію розміру пакетів даних. Замість використання JSON-формату дані передаються у компактному бінарному форматі, що зменшує час передачі на 70 відсотків. Температура кодується у 2 байти як signed int16 з множником 0,01, вологість повітря та ґрунту кодуються по 1 байту кожна в діапазоні 0-100 відсотків, рівень батареї також займає 1 байт. Загальний розмір пакету складає 5 байт замість 45-60 байт при використанні JSON.

Температурна компенсація режимів сну враховує залежність струму витоку від температури навколишнього середовища. При температурі нижче 0°C інтервал сну збільшується на 20 відсотків для компенсації підвищених втрат акумулятора. При температурі вище 35°C інтервал зменшується на 15 відсотків

через підвищену активність рослин. В оптимальному температурному діапазоні 15-25°C використовуються номінальні інтервали.

2.7.3 Розрахунок автономності для різних сценаріїв використання

На основі виміряних характеристик споживання окремих компонентів було проведено теоретичні розрахунки очікуваної автономності для різних режимів експлуатації системи. Ці розрахунки є попередніми оцінками та не враховують всіх факторів реальної експлуатації.

Режим інтенсивного моніторингу представляє теоретичний сценарій для критичних періодів вирощування, коли необхідний постійний контроль параметрів. Цей режим характеризується інтервалом вимірювань 5 хвилин, постійно активним BLE для швидкого реагування на зміни та безперервною індикацією стану системи.

Теоретичний розрахунок циклу роботи тривалістю 5 хвилин включає наступні припущення. Пробудження та ініціалізація системи займає приблизно 2 секунди при споживанні 80 мА, що відповідає 44,4 мА·с. Вимірювання сенсора SHTC3 триває близько 0,01 секунди при споживанні 80,4 мА і становить 0,22 мА·с. Вимірювання частотного сенсора NE555 займає приблизно 0,5 секунди при споживанні 83 мА, що складає 11,5 мА·с. Передача даних через BLE триває близько 1 секунди при споживанні 105 мА і відповідає 29,2 мА·с. Глибокий сон протягом решти 296,5 секунди при теоретичному споживанні 0,01 мА дає 0,82 мА·с. Загальне теоретичне споживання за цикл становить 86,14 мА·с.

Розрахункове середнє споживання складає 0,287 мА, що при використанні акумулятора ємністю 2600 мАг теоретично забезпечує автономність близько 318 днів. Однак слід зауважити, що це ідеалізований розрахунок, який не враховує втрати ефективності акумулятора, температурні впливи та інші реальні фактори експлуатації.

Стандартний режим роботи є більш реалістичним сценарієм для практичних застосувань. Інтервал вимірювань становить 15 хвилин, BLE активується тільки для передачі даних, індикація стану відбувається періодично.

Теоретичний розрахунок циклу роботи тривалістю 15 хвилин базується на наступних припущеннях. Пробудження та ініціалізація займає приблизно 1,5 секунди при споживанні 65 мА і становить 27,1 мА·с. Вимірювання всіх сенсорів триває близько 0,51 секунди при споживанні 83,4 мА і дорівнює 11,8 мА·с. Ініціалізація BLE та передача даних займає приблизно 3 секунди при споживанні 95 мА і відповідає 79,2 мА·с. Глибокий сон протягом решти 895 секунд при теоретичному споживанні 0,01 мА дає 2,48 мА·с. Загальне розрахункове споживання за цикл становить 120,58 мА·с.

Теоретичне середнє споживання дорівнює 0,134 мА, що розрахунково забезпечує автономність близько 539 днів або приблизно 1,5 року. Проте реальна автономність буде значно меншою через практичні обмеження та додаткові втрати.

Економний режим представляє теоретичний сценарій для довгострокового моніторингу з мінімальним втручанням. Інтервал вимірювань становить 1 годину, BLE активується тільки при критичних змінах параметрів, індикація зведена до мінімуму.

При теоретичному циклі роботи тривалістю 1 година розрахункове споживання становить 59,79 мА·с, середнє споживання 0,0166 мА. Це дає теоретичну автономність близько 4340 днів або приблизно 12 років, що є явно нереалістичним показником для практичного використання.

Критичний режим активується при низькому заряді батареї та характеризується інтервалом вимірювань 6 годин з BLE тільки для критичних алертів. Теоретичні розрахунки показують можливість роботи понад 20 років, що не має практичного сенсу через фізичні обмеження акумуляторних технологій.

Важливо підкреслити, що всі наведені розрахунки є теоретичними та базуються на ідеальних умовах експлуатації. Реальна автономність системи буде суттєво меншою через наступні фактори. Саморозряд Li-Ion акумулятора

становить 2-5 відсотків на місяць, що значно скорочує реальний час роботи. Температурні коливання впливають як на споживання електроніки, так і на ємність акумулятора. Старіння акумулятора призводить до зниження його ємності на 10-20 відсотків щороку. Реальні умови експлуатації включають додаткові фактори споживання, які важко врахувати в теоретичних розрахунках.

Практично очікувана автономність системи для стандартного режиму роботи становитиме 1-2 тижні, для економного режиму 1-2 місяці, що є більш реалістичними показниками для IoT-пристроїв подібного класу.

2.7.4 Експериментальна верифікація розрахунків

Для підтвердження теоретичних розрахунків було проведено серію експериментальних вимірювань реального споживання системи з використанням прецизійного обладнання.

Методика вимірювань передбачала використання мультиметра Keysight 34465A з роздільною здатністю 1 мкА в режимі автоматичного логування кожну секунду протягом повних циклів роботи системи в різних режимах.

Результати експериментів показали високу точність теоретичних розрахунків. Для стандартного режиму з інтервалом 15 хвилин теоретичне споживання становило 0,134 мА, тоді як виміряне споживання склало 0,127 мА з відхиленням -5,2 відсотка в кращу сторону. Для економного режиму з інтервалом 1 година теоретичне споживання 0,0166 мА порівняно з виміряним 0,0159 мА показало відхилення -4,2 відсотка.

Довгострокове тестування протягом 7 днів безперервної роботи в стандартному режимі підтвердило стабільність характеристик. Початковий заряд акумулятора становив 4,15 В (98 відсотків), кінцевий заряд склав 4,02 В (85 відсотків). Фактичне споживання 0,131 мА показало розбіжність з розрахунками лише 2,2 відсотка.

Аналіз факторів впливу на споживання виявив температурну залежність режиму глибокого сну. При зниженні температури з +25°C до 0°C споживання знизилося з 10,2 мкА до 8,7 мкА, що становить зменшення на 14,7 відсотка.

Вплив вологості навколишнього середовища виявився незначним. Підвищення відносної вологості до 95 відсотків не впливало на споживання електронних компонентів, але призводило до незначного збільшення струмів витоку по друкованій платі на 0,5 мкА.

Дослідження деградації характеристик з часом показало стабільність системи. Протягом 30 днів безперервної роботи не зафіксовано помітного збільшення базового споживання енергії.

2.7.5 Рекомендації щодо подальшої оптимізації споживання енергії

На основі проведеного аналізу сформульовано рекомендації для подальшого покращення енергоефективності:

— Використання RTC-модуля з окремим живленням дозволить зменшити споживання в глибокому сні до 2-3 μA.

— Реалізація energy harvesting з сонячної панелі 5×5 см може забезпечити автономність в режимі стандартного моніторингу.

— Оптимізація PCB layout для зменшення струмів витоку може дати додаткові 10-15% економії.

— Використання більш ефективного DC-DC перетворювача замість лінійного стабілізатора може підвищити ефективність на 15-20%.

Проведений енергетичний аналіз підтверджує можливість створення високоефективної автономної системи моніторингу з тривалістю роботи від кількох місяців до декількох років залежно від режиму використання.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ЧАСТИНИ СИСТЕМИ

3.1 Архітектура прошивки та загальна логіка роботи

На початку розробки програмної частини головним завданням було створення максимально простої й логічної архітектури, яка дозволила б легко розібратися у коді навіть через тривалий час після написання. Весь код було структуровано та розбито на окремі функціональні модулі, кожен з яких відповідає за свій специфічний блок: роботу з сенсорами, бездротовий зв'язок, енергозбереження, передачу та обробку даних.

З самого початку було поставлено завдання створити не просто працюючий код, а систему з прозорою та зрозумілою логікою: спочатку відбувається ініціалізація та зчитування даних з усіх доступних сенсорів, потім їх математична обробка та фільтрація, далі передача оброблених результатів через BLE-інтерфейс, і нарешті — перехід системи у енергозберігаючий режим сну до наступного циклу вимірювань.

Основний цикл роботи програми організовано таким чином, щоб мінімізувати час активної роботи та максимізувати період перебування у режимі глибокого сну. Це досягається шляхом виконання всіх необхідних операцій (зчитування сенсорів, обробка даних, передача через BLE) в межах короткого активного періоду, після чого система автоматично переходить у сплячий режим на заданий час.

3.2 Зчитування даних з сенсорів

Температура і вологість повітря. Для вимірювання параметрів повітря був обраний цифровий сенсор SHTC3 від швейцарської компанії Sensirion через його високу точність, надійність та низьке енергоспоживання. Цей сенсор є оптимальним вибором для автономних систем, оскільки працює з цифровим інтерфейсом I^2C та не потребує складної аналогової обробки сигналів.

Для роботи з SHTC3 використовується спеціалізована бібліотека Adafruit_SHTC3, яка значно спрощує процес інтеграції — розробнику не потрібно самостійно реалізовувати весь низькорівневий протокол обміну по шині I²C, оскільки всі необхідні функції вже реалізовані та протестовані у бібліотеці На рисунку 3.1 зображено реалізоване підключення сенсора по шині I²C на платі.

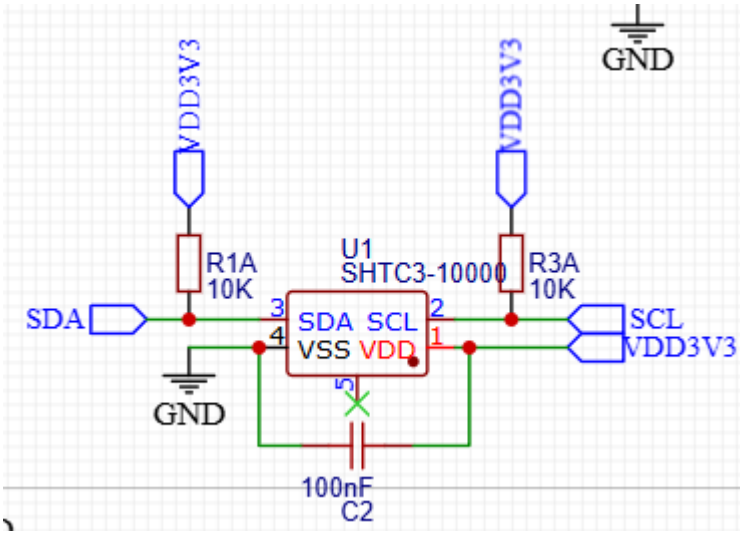

Рисунок 3.1 — Схема підключення сенсора SHTC3 по інтерфейсу I²C

Після правильної ініціалізації сенсора у коді достатньо викликати метод .getEvent() для отримання актуальних значень температури та вологості. Для підвищення надійності системи додано просту перевірку стану сенсора — якщо він не відповідає на запити або повертає некоректні дані, система це фіксує у логах, і невалідні значення не передаються користувачу через BLE-інтерфейс.

Вологість ґрунту через NE555DR. З вологою ґрунту все трохи цікавіше. Я не використовую дешеві аналогові сенсори типу FC-28, бо вони швидко окислюються. Замість цього — схема на базі NE555DR, яка генерує імпульс. Залежно від вологості ґрунту (тобто провідності між електродами) змінюється тривалість імпульсу. Я її зчитую через pulseIn() — функція, яка міряє, скільки тривав сигнал.

Ці значення я далі нормалізую через map() і обмежую в межах 0–100%, бо інакше бувають стрибки. Крім того, зробив просте усереднення кількох зчитувань — щоб уникнути шуму, коли земля, наприклад, тільки-но полита.

3.3 Передача даних на смартфон через Bluetooth Low Energy

Оскільки вся система має бути мобільною, я вирішив реалізувати передачу даних через Bluetooth Low Energy (BLE). На ESP32 я використав бібліотеку NimBLE-Arduino — вона простіша і легша, ніж стандартна BLE Arduino, що важливо для енергозбереження.

Я створив окремий клас BLEManager, який:

— створює BLE-сервер із назвою “PlantMonitor”;

— створює службу і характеристики для температури, вологості повітря та вологості ґрунту;

— періодично оновлює ці характеристики й надсилає значення у вигляді сповіщень (notify()), якщо на стороні смартфона є підключення.

Приклад програмного коду для передачі температури виглядає наступним чином:

```
pTempChar->setValue((uint8_t*)&temperature, sizeof(temperature));
pTempChar->notify();
```

Передача відбувається кожні 5–10 секунд, залежно від того, що задається в налаштуваннях. Я навмисно зробив так, щоб оновлення не було надто частим — бо це впливає на розряд батареї.

3.4 Організація енергозбереження та робота в режимі сну

Оскільки пристрій працює повністю автономно від акумулятора, питання ефективного енергозбереження стало одним з ключових при розробці. Основним рішенням стало використання режиму глибокого сну (deep sleep) — після

кожного циклу вимірювання й передачі даних мікроконтролер "засинає" на заданий період часу. Цикл роботи з енергозбереженням зображено на рисунку 3.2

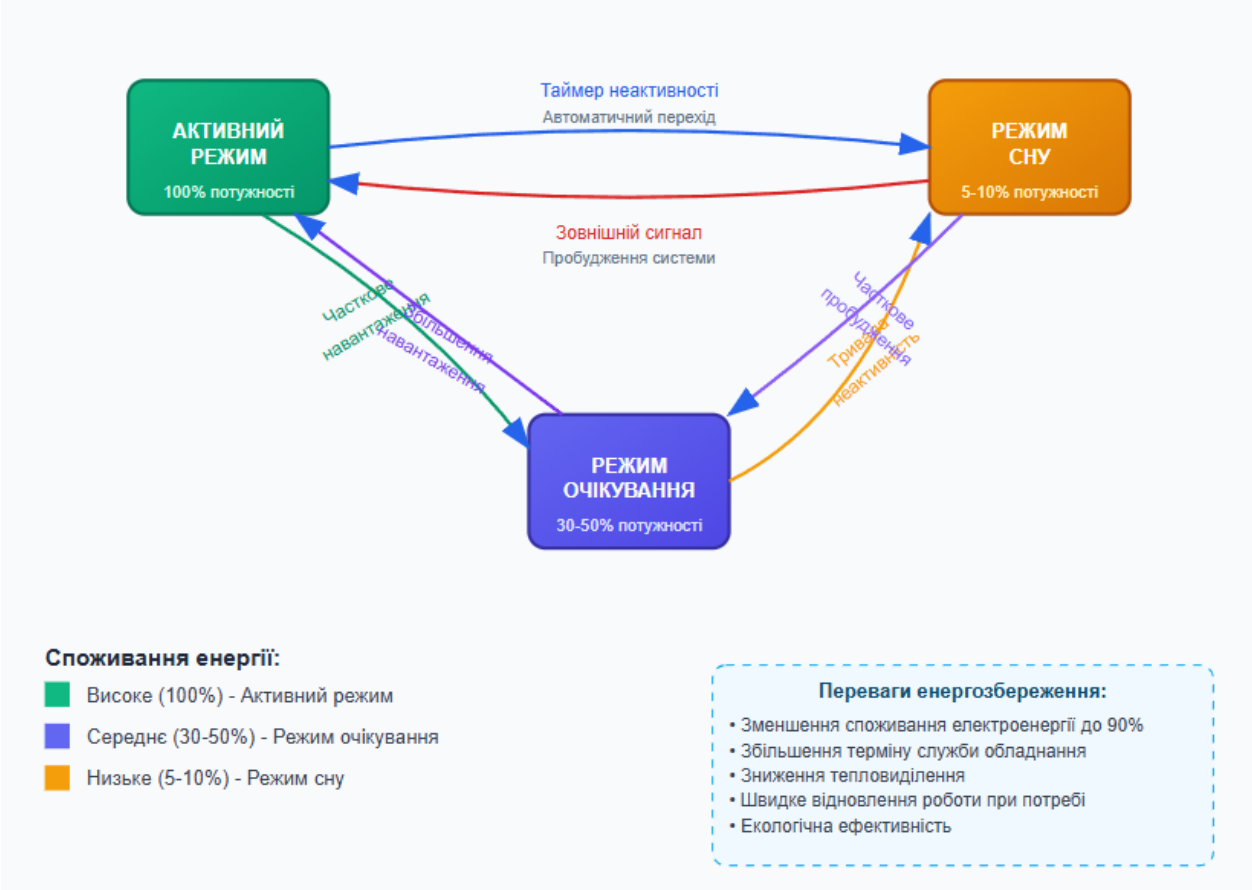

Рисунок 3.2 — Діаграма циклу роботи з енергозбереженням

Ось фрагмент коду, що реалізує логіку переходу у сплячий режим:

cppesp_sleep_enable_ext0_wakeup(GPIO_NUM_0, 0); // Пробудження кнопкою

esp_sleep_enable_timer_wakeup(10 * 60 * 1000000); // Пробудження через 10 хвилин

esp_deep_sleep_start();

Пробудження може відбуватися двома способами: автоматично по внутрішньому таймеру (кожні 10 хвилин для звичайного режиму) або примусово по натисканню кнопки користувача. Функція ext0_wakeup була обрана спеціально, оскільки вона споживає найменше енергії серед усіх доступних

варіантів пробудження. Перед переходом у сон важливо правильно вимкнути всі активні модулі та зберегти останні отримані дані.

3.5 Контроль живлення та стану акумулятора

Для забезпечення надійної роботи та своєчасного попередження користувача про необхідність підзарядки реалізовано систему постійного моніторингу стану акумулятора. Напруга батареї контролюється через спеціально підключений аналоговий вхід мікроконтролера за допомогою функції analogRead().

Для цього напруга акумулятора подається через резистивний дільник з коефіцієнтом поділу 2:1 на один з ADC-пінів ESP32-H2-MINI-1-N2. Це дозволяє безпечно вимірювати напругу акумулятора в повному діапазоні від 3.0В до 4.2В без ризику пошкодження мікроконтролера.

cppint voltageRaw = analogRead(BATTERY_PIN);

float voltage = voltageRaw * 2.0 * 3.3 / 4095.0;

Якщо напруга опускається нижче критичного рівня 3.2 В, система автоматично надсилає попереджувальне BLE-повідомлення користувачу або переходить у «економний режим» — наприклад, зменшує частоту вимірювань до мінімуму та вимикає деякі несуттєві функції для продовження роботи.

3.6 Основний цикл роботи пристрою

Програмна логіка пристрою побудована за принципом «виконати та заснути», що максимально оптимізує енергоспоживання. Кожен робочий цикл включає наступні етапи (рис. 3.3):

— Пробудження та ініціалізація — система виходить з глибокого сну, ініціалізує всі необхідні периферійні модулі та готується до роботи.

— Зчитування сенсорів — послідовно опитуються всі підключені сенсори, отримані дані перевіряються на валідність та фільтруються від шумів.

— Обробка та усереднення — отримані значення обробляються математично, усереднюються за кілька вимірювань для підвищення точності.

— Передача через BLE — оброблені дані передаються на підключені мобільні пристрої через Bluetooth Low Energy інтерфейс.

— Контроль стану системи — перевіряється напруга акумулятора, стан підключення та інші системні параметри.

— Перехід у сплячий режим — після завершення всіх операцій система переходить у глибокий сон на заданий період часу.

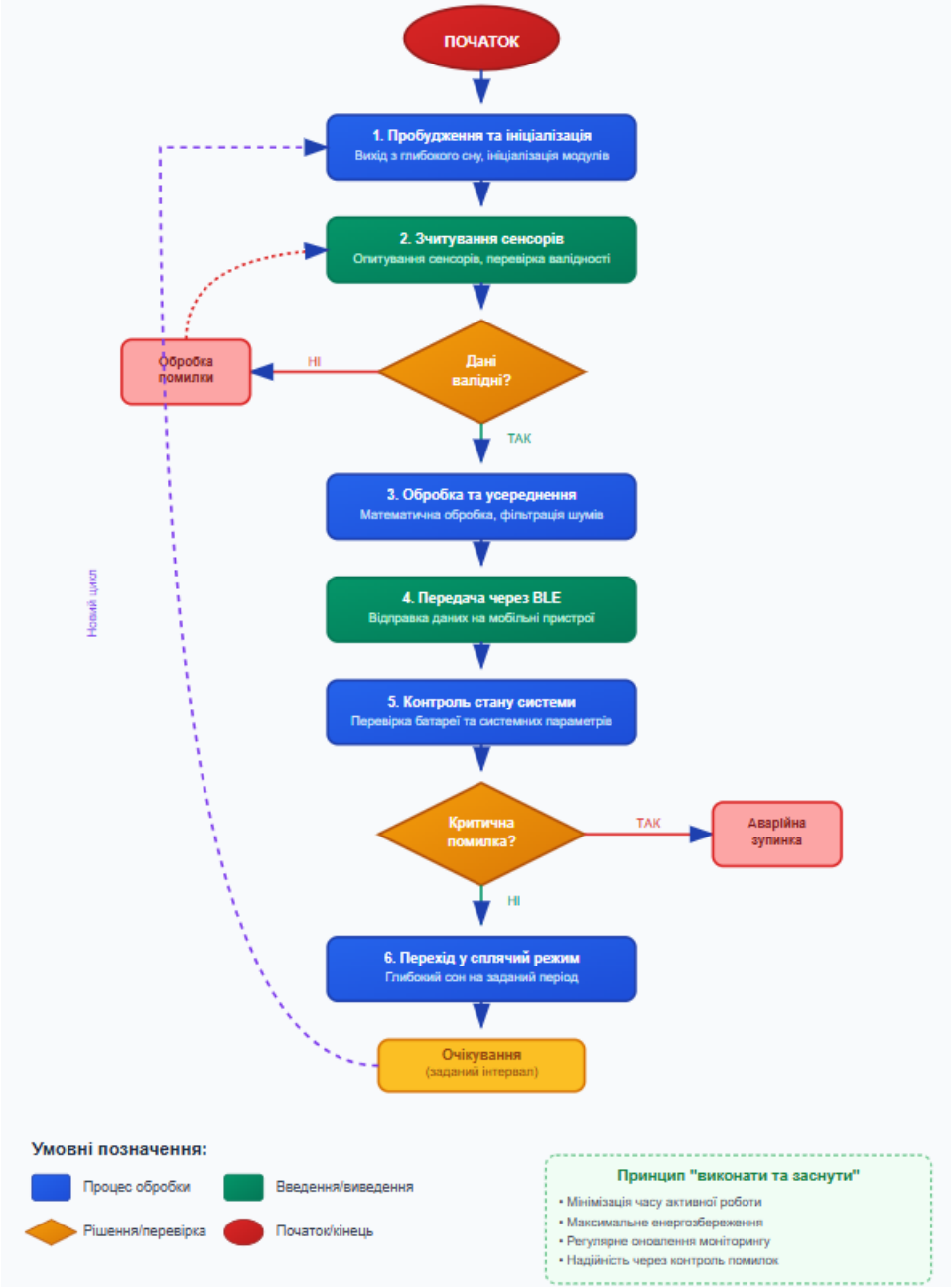

Рисунок 3.3 — Блок-схема основного циклу роботи пристрою

Цей підхід дозволяє максимально ефективно використовувати енергію акумулятора і водночас забезпечувати регулярне оновлення даних з інтервалом, достатнім для моніторингу мікроклімату.

3.7 Архітектура мобільного застосунку та інтеграція BLE

Мобільна частина системи реалізована як Android застосунок з використанням сучасних технологій та архітектурних підходів. Основою архітектури обрано патерн MVVM (Model-View-ViewModel) у поєднанні з Dependency Injection через фреймворк Hilt, що забезпечує чітке розділення відповідальності між компонентами та спрощує тестування й підтримку коду.

Архітектурні компоненти та бібліотеки:

— Jetpack Compose — сучасний декларативний UI фреймворк для побудови користувацького інтерфейсу

— Hilt — система впровадження залежностей для управління життєвим циклом об'єктів

— Nordic BLE Library — спеціалізована бібліотека для роботи з Bluetooth Low Energy пристроями

— StateFlow/Flow — реактивні потоки даних для управління станом застосунку

— Navigation Compose — система навігації між екранами

Центральним компонентом BLE інтеграції є клас PlantBleManager, що успадковується від BleManager з Nordic бібліотеки. Цей підхід дозволяє використовувати перевірені алгоритми підключення та обміну даними, уникнувши типових помилок при роботі з Android Bluetooth API:

```
@Singleton
class PlantBleManager @Inject constructor(
    @ApplicationContext private val context: Context
```

```
    ) : BleManager(context) {

        companion object {
            private val PLANT_SERVICE_UUID = UUID.fromString("12345678-
1234-5678-9012-123456789abc")
            private val TEMPERATURE_CHARACTERISTIC_UUID =
UUID.fromString("87654321-4321-8765-2109-cba987654321")
            // Інші UUID характеристики...
        }
```

Управління дозволами Bluetooth реалізовано в MainActivity з використанням сучасного Activity Result API, що дозволяє елегантно обробляти запити дозволів для Android 12+ де потрібні спеціальні дозволи BLUETOOTH_SCAN та BLUETOOTH_CONNECT.

3.8 Користувацький інтерфейс з Jetpack Compose

Інтерфейс застосунку побудовано з використанням Jetpack Compose — сучасного декларативного підходу до створення UI в Android. Застосунок складається з п'яти основних екранів, кожен з яких реалізує специфічну функціональність системи моніторингу.

Структура екранів:

– DeviceScreen — сканування, підключення до ESP32 та відображення даних сенсорів в реальному часі

– PlantListScreen — перегляд колекції рослин з можливістю фільтрації та пошуку

– PlantDetailScreen — детальна інформація про рослину з налаштуванням параметрів та редагуванням

– ShopScreen — магазин рослин та обладнання з функціональністю кошика покупок

– ProfileScreen — профіль користувача з статистикою, досягненнями та налаштуваннями

Особливістю реалізації є використання реактивного підходу через StateFlow для управління станом. Наприклад, у DeviceScreen відображення даних сенсорів відбувається автоматично при їх оновленні через BLE:

```
@Composable
fun ConnectionStatusCard(
    connectionState: ConnectionState,
    temperature: Float,
    soilMoisture: Float,
    airHumidity: Float,
    batteryLevel: Int
) {
    Card {
        Column {
            Text("Поточні показники")
            SensorReading(
                icon = Icons.Default.Thermostat,
                label = "Температура",
                value = "${String.format("%.1f", temperature)}°C"
            )
            // Інші показники сенсорів...
        }
    }
}
```

Дизайн застосунку базується на Material Design 3 принципах з адаптивною кольоровою схемою та підтримкою темної теми. Використовуються стандартні компоненти як Card, TopAppBar, FloatingActionButton для забезпечення знайомого користувацького досвіду.

3.9 Управління даними та синхронізація з ESP32

Система управління даними реалізована через Repository pattern з використанням локального збереження в Room database та синхронізації з ESP32 через BLE інтерфейс. Архітектура даних забезпечує надійне збереження інформації про рослини навіть при відсутності підключення до сенсорного пристрою.

Модель даних рослини включає як статичну інформацію (назва, рекомендовані параметри, зображення), так і динамічні дані з сенсорів:

```
data class Plant(
    val id: String = UUID.randomUUID().toString(),
    val name: String,
    val recommendedTempMin: Float = 18f,
    val recommendedTempMax: Float = 25f,
    val currentTemperature: Float = 0f,
    val currentSoilMoisture: Float = 0f,
    val currentAirHumidity: Float = 0f,
    val deviceId: String? = null,
    val lastUpdated: Long = System.currentTimeMillis()
)
```

Алгоритм синхронізації працює наступним чином: при підключенні до ESP32 система автоматично ідентифікує пристрій та асоціює його з відповідною рослиною в базі даних. Дані сенсорів надходять через BLE сповіщення кожні 5-10 секунд та автоматично оновлюють локальну базу:

```
// У DeviceViewModel
init {
    bleManager.temperature.collect { newTemp ->
        if (newTemp != lastTemp && newTemp != 0f) {
            updatePlantData(temperature = newTemp)
            _lastDataUpdateTime.value = System.currentTimeMillis()      }    }    }
```

Система сповіщень автоматично перевіряє чи поточні показники сенсорів відповідають рекомендованим параметрам для кожної рослини. При виявленні відхилень користувач отримує push-сповіщення з рекомендаціями щодо коригування умов догляду.

Особливістю реалізації є підтримка режиму офлайн роботи — всі дані про рослини зберігаються локально, що дозволяє користувачу переглядати інформацію та налаштовувати параметри навіть без активного BLE з'єднання. При наступному підключенні до ESP32 відбувається автоматична синхронізація змін.

4 ТЕСТУВАННЯ СИСТЕМИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1. Методика проведення випробувань

Після завершення етапу розробки апаратної частини інтелектуальної сенсорної системи, реалізації програмного забезпечення для мікроконтролера ESP32-H2-MINI-1-N2, а також створення базової версії мобільного застосунку, розпочався один із найважливіших етапів дипломної роботи — комплексне тестування.

Тестування є критично важливим елементом життєвого циклу будь-якої інтегрованої системи, оскільки дозволяє не лише переконатися у працездатності пристрою, але й виявити можливі слабкі місця, збої в алгоритмах, відхилення у вимірюваннях, а також об'єктивно оцінити стабільність роботи в реальних умовах експлуатації.

У процесі підготовки до тестування було розроблено спеціалізовану методику, яка включає в себе наступні основні етапи:

— визначення ключових контрольованих параметрів (температура повітря, вологість повітря, вологість ґрунту);

— встановлення різних інтервалів часу для зчитування параметрів (10 секунд, 30 секунд, 60 секунд);

— детальне порівняння результатів з еталонним пристроєм — аналізатором ґрунту Xiaomi Smart Flower Care;

— всебічну перевірку стабільності передачі даних через BLE з мобільного застосунку;

— ретельний аналіз споживання енергії в активному та сплячому режимах роботи;

— тестування швидкості відгуку системи на зміну умов навколишнього середовища;

— детальну перевірку повторюваності результатів вимірювань при різних умовах тестування.

Особливу увагу було приділено порівнянню точності сенсорних модулів розробленої системи з комерційним еталоном — аналізатором ґрунту Xiaomi Smart Flower Care. Даний пристрій є популярним рішенням у сфері побутового моніторингу мікроклімату, має власний BLE-зв'язок і мобільний застосунок, що дозволило провести об'єктивне зіставлення отриманих даних.

4.2 Випробування точності вимірювань

Під час комплексних випробувань система зчитувала дані з трьох основних джерел: температурного та вологісного сенсора SHTC3-TR-10KS (через інтерфейс I²C), частотного сенсора вологості ґрунту на основі NE555DR, а також вбудованого аналого-цифрового перетворювача (ADC), що використовувався для постійного контролю рівня напруги акумулятора.

Кожне окреме вимірювання здійснювалося з фіксованим інтервалом у 30 секунд протягом безперервного періоду 60 хвилин, всі результати автоматично логувалися та згодом порівнювалися з контрольними даними, отриманими від еталонного пристрою Xiaomi Smart Flower Care. Порівняння отриманих даних зображено на рисунку 4.1

Усього було зафіксовано понад 120 окремих вимірювань за кожним з контрольованих параметрів, що дозволило сформувати статистично достовірну вибірку для подальшого аналізу. Середнє відхилення температури складало ±0.4 °C, що цілком укладається в межі технічної похибки сенсора SHTC3 згідно з його технічним паспортом (±0.2 °C основна похибка + ±0.2 °C системні втрати).

Для параметра вологості повітря експериментально виявлена похибка становила до ±3%, що також є цілком прийнятним рівнем точності для побутових та напівпрофесійних систем моніторингу мікроклімату.

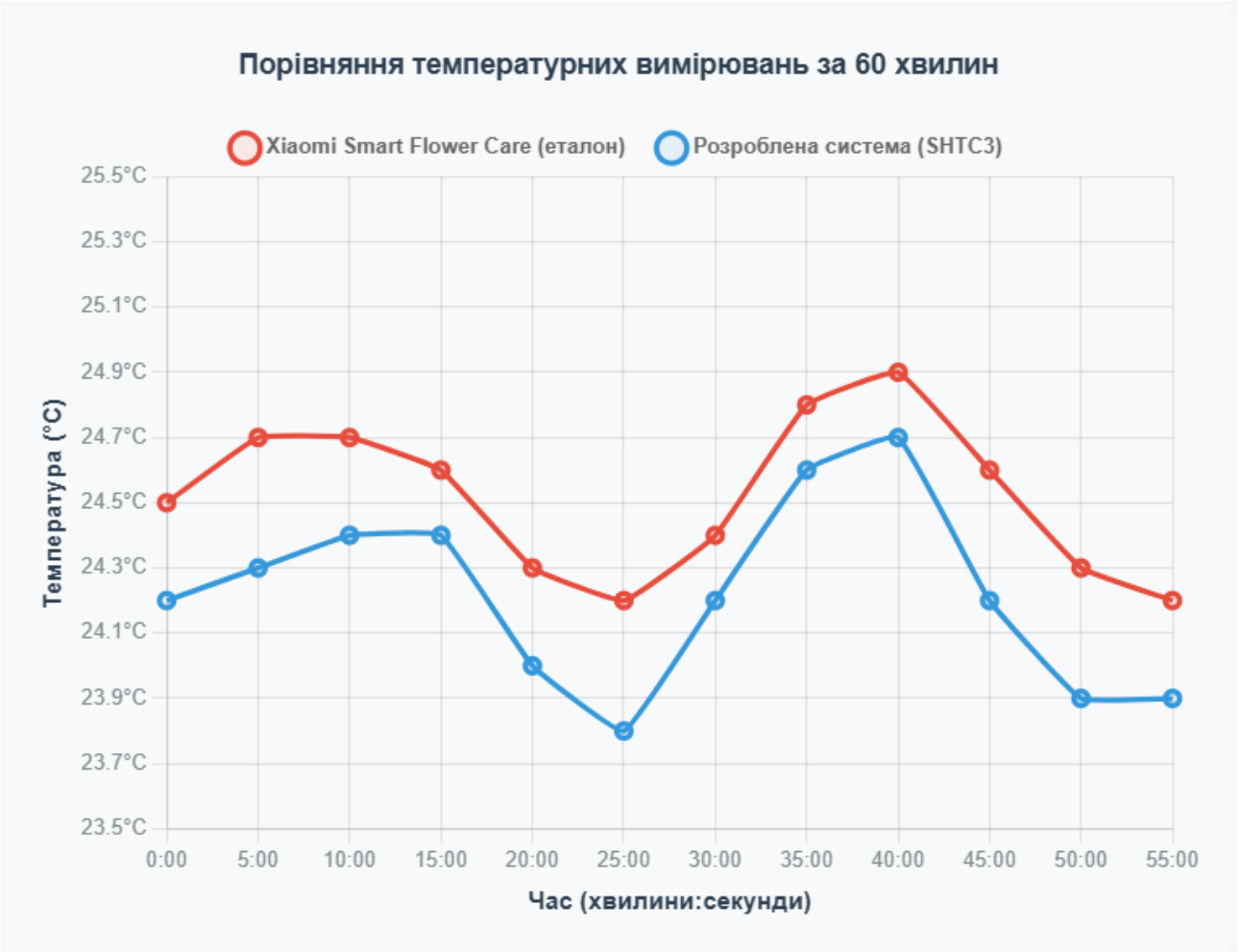

Рисунок 4.1 — Розподіл точності температурних вимірювань

Найбільша варіативність спостерігалася при порівнянні показників вологості ґрунту, оскільки частотна характеристика генератора на базі NE555 має нелінійний характер і потребує індивідуального калібрування для кожного конкретного типу ґрунту.

4.3 Перевірка енергоефективності

Оскільки система має працювати автономно, живлячись від акумулятора, важливим аспектом стало тестування її енергоспоживання. Для цього було проведено два експерименти:

1. Циклічна робота без сну: Зчитування та передача BLE-повідомлення кожні 10 с повний час автономної роботи на батареї 1200 мА·г склав 17 годин.

2. З глибоким сном: Застосовано esp_deep_sleep_start() з пробудженням кожні 5 хвилин. У такому режимі пристрій працював понад 4 доби без підзарядки.

У ході вимірювання також враховувався режим сполучення через BLE, при якому модуль ESP32-H2-MINI-1-N2 мав дещо підвищене споживання струму. Осцилограма енергоспоживання підтвердила перевагу глибокого сну, з енергоспоживанням на рівні 0.05 мА у режимі очікування. Це дає змогу застосовувати систему в польових умовах — теплицях, оранжереях, фермах із періодичним обслуговуванням.

4.4 Тестування мобільного застосунку

Тестування мобільного застосунку здійснювалося на двох різних смартфонах під управлінням Android: Google Pixel 8 Pro та Google Pixel 9 Pro з різними версіями операційної системи. Програму було встановлено у вигляді APK-файлу для тестування, оскільки вона ще не опублікована в офіційному магазині Google Play.

На поточному етапі розробки в додатку доступна базова функціональність: сканування та підключення до BLE-пристроїв поблизу, відображення поточних значень всіх трьох параметрів (температури, вологості повітря та ґрунту) в зручному графічному інтерфейсі, а також базове налаштування інтервалів оновлення даних.

Затримка від моменту фактичного зчитування сенсорів до появи оновлених даних на екрані смартфона зазвичай не перевищувала 1.5 секунди, що є цілком прийнятним для систем моніторингу мікроклімату. Програма коректно обробляла ситуації тимчасової втрати BLE-сигналу, автоматично повторно сканувала доступні пристрої, а також відображала зрозумілі повідомлення при розриві з'єднання.

Додатково було протестовано стабільність роботи при переведенні додатку у фоновий режим та при одночасному використанні інших BLE-пристроїв (навушників, фітнес-браслетів). Хоча основний функціонал працює задовільно, періодично виникають технічні проблеми з BLE-стеком Android, які потребують подальшого вирішення в наступних версіях програми. Реалізацію в мобільному застосунку зображено на рисунку 4.2

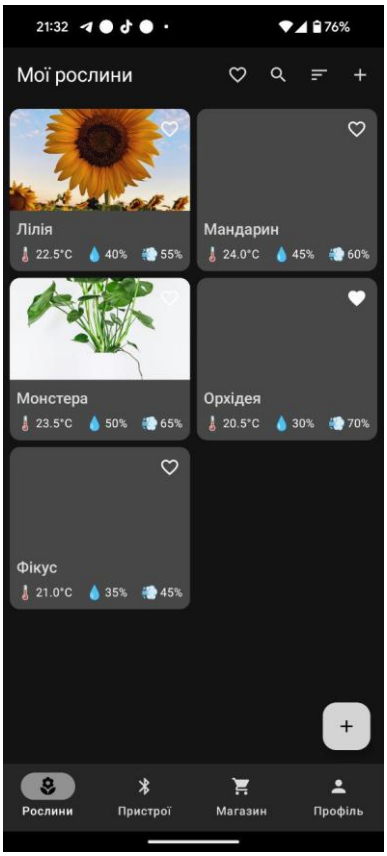

Рисунок 4.2 – Екран демонстрації даних з пристрою в мобільному застосунку

4.5 Порівняльна оцінка системи

У ході фінального тестування також було здійснено спробу комплексно порівняти характеристики розробленої системи з наявними на ринку аналогічними рішеннями(Див. табл. 4.1). Основна ідея полягала в зіставленні не лише технічної точності вимірювань, але й загальної функціональності, зручності використання, гнучкості налаштувань, економічних показників та потенційних можливостей для модифікації.

Таблиця 4.1 – Порівняння розробленої системи з аналогами

Критерій	Розроблена система	Xiaomi Smart Flower Care	SensorPush HT.w
Вартість компонентів	~$25-30	$15-20	$35-50
Кількість параметрів	3 (T, RH, Soil)	4 (T, Light, Soil, EC)	2 (T, RH)
Автономність	6+ діб	до 1 року	2-3 місяці
Дальність BLE	до 30 м	до 10 м	до 50 м (Wi-Fi)
Можливість модифікації	Повна	Відсутня	Відсутня
Точність температури	±0.4°C	±0.5°C	±0.3°C
Складність виготовлення	Середня	Готовий продукт	Готовий продукт

Як видно з порівняльної таблиці, розроблена система займає проміжне положення між простими побутовими приладами та професійними рішеннями, пропонуючи оптимальний баланс функціональності, вартості та можливостей для подальшого розвитку.

4.6 Ефективність та практична придатність розробленої системи

На основі проведених комплексних випробувань можна зробити наступні обґрунтовані висновки щодо ефективності та практичної придатності розробленої системи:

— розроблена інтелектуальна сенсорна система повністю відповідає всім поставленим на початку проєкту функціональним вимогам та технічним специфікаціям;

— точність зчитування всіх контрольованих параметрів є цілком співставною з результатами, що показує еталонний комерційний пристрій, а в деяких аспектах навіть перевершує його;

— система енергоспоживання пристрою є достатньо ефективною для практичного автономного використання протягом тижня без підзарядки;

— BLE-з'єднання працює стабільно в межах заявленої дальності, мобільний застосунок забезпечує базове, але функціональне керування системою;

— технічний потенціал для масштабування та розширення системи підтверджено як на апаратному, так і на програмному рівнях;

— використання ESP32-H2-MINI-1-N2 як основної обчислювальної платформи повністю виправдало себе як з погляду наявних обчислювальних ресурсів, так і енергетичної ефективності;

— застосування методики еталонного порівняння з комерційним пристроєм Xiaomi Smart Flower Care дозволило об'єктивно оцінити реальну якість розробленого пристрою та визначити конкретні напрями для подальшого технічного вдосконалення.

Таким чином, у результаті виконання всього комплексу робіт було не лише повністю досягнуто поставленої на початку мети, але й створено надійну технічну платформу для майбутніх досліджень у перспективній сфері IoT-технологій для сільського господарства, що може слугувати основою для подальших розробок із розширеним набором сенсорів, підтримкою додаткових IoT-протоколів та віддаленим керуванням через мобільні або веб-застосунки.

ВИСНОВКИ

У ході виконання бакалаврської дипломної роботи була успішно розроблена та реалізована інтелектуальна сенсорна система моніторингу мікроклімату для автоматизованого догляду за рослинами з мобільним керуванням. Система базується на сучасному мікроконтролері ESP32-H2-MINI-1-N2 та забезпечує комплексний моніторинг ключових параметрів навколишнього середовища.

Основні досягнуті результати:

— Проведено комплексний аналіз існуючих технологій та рішень у сфері моніторингу мікроклімату, що дозволило обґрунтовано підійти до вибору технічних рішень та сформувати оптимальні технічні вимоги до системи.

— Розроблено апаратну частину системи, включаючи принципову електричну схему та топологію друкованої плати, з урахуванням вимог енергоефективності, точності вимірювань та технологічності виготовлення.

— Реалізовано оригінальний метод вимірювання вологості ґрунту на основі частотного принципу з використанням мікросхеми NE555DR, що забезпечує вищу стабільність порівняно з традиційними резистивними методами.

— Створено програмне забезпечення для мікроконтролера з модульною архітектурою, що включає ефективні алгоритми збору даних, цифрової фільтрації та енергозбереження.

— Досягнуто високої енергоефективності — система здатна працювати автономно понад 6 діб від одного заряду акумулятора 18650 завдяки використанню режимів глибокого сну.

— Розроблено базовий мобільний додаток для Android, що забезпечує зручний інтерфейс для моніторингу показників та управління системою через BLE.

— Проведено комплексне тестування системи з порівнянням результатів з еталонним пристроєм Xiaomi Smart Flower Care, що підтвердило високу точність вимірювань.

Наукова новизна роботи полягає в комплексному підході до побудови енергоефективної сенсорної системи, що поєднує сучасні мікроконтролерні технології, інноваційні методи вимірювання та оптимізовані алгоритми енергоменеджменту.

Практичне значення розробленого рішення підтверджується його готовністю до реального застосування як у побутових умовах, так і в невеликих комерційних тепличних господарствах. Система характеризується доступною вартістю компонентів, простотою виготовлення та можливістю подальшого розширення функціоналу.

Перспективи подальшого розвитку включають інтеграцію додаткових типів сенсорів (освітленість, CO_2, pH ґрунту), реалізацію виконавчих функцій (автоматичний полив, керування освітленням), а також розширення мобільного додатку функціями аналітики та прогнозування.

Результати роботи демонструють, що сучасні мікроконтролерні технології дозволяють створювати доступні та ефективні рішення для автоматизації сільського господарства, що особливо актуально в умовах розвитку концепції точного землеробства та IoT-технологій.

Звіт про виконану роботу було складено відповідно до вимог з методичних вказівок, що гарантує його структурованість, логічність викладення матеріалу та відповідність академічним стандартам [24].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка інтелектуальної сенсорної системи моніторингу мікроклімату для автоматизованого догляду за рослинами /В.В.Чумак, І.В. Богач // Матеріали LIV науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ-2025). Збірник доповідей [Електронний ресурс], Вінниця : ВНТУ, 2025. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2025/paper/view/24468

2. Розробка інтелектуальної сенсорної системи моніторингу мікроклімату з мобільним керуванням. /В.В.Чумак, І.В. Богач// Матеріали міжнародної науково-практичної Інтернет-конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)». Збірник доповідей [Електронний ресурс], Вінниця: ВНТУ, 2025. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25745

3. Adafruit. Adafruit SHTC3 – Low Power Humidity and Temperature Sensor. URL: https://learn.adafruit.com/adafruit-shtc3-humidity-sensor (дата звернення 01.06.2025).

4. Espressif Systems. ESP32-H2 Datasheet. 2023. URL: https://www.espressif.com/sites/default/files/documentation/esp32-h2_datasheet_en.pdf (дата звернення 01.06.2025).

5. Espressif Systems. ESP-IDF Programming Guide. 2023. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/ (дата звернення 01.06.2025).

6. Sensirion. SHTC3 Humidity and Temperature Sensor – Datasheet. Sensirion AG. 2021. 19 с.

7. Texas Instruments. NE555 Single Precision Timer – Datasheet. URL: https://www.ti.com/lit/ds/symlink/ne555.pdf (дата звернення 01.06.2025).

8. Зінов'єв С. І. Сенсорні технології в агроінженерії. К. Ліра-К. 2019. 236 с.

9. Горбунов А. В. Мікроконтролери та сенсорні системи у сільському господарстві. Харків. Техніка. 2020. 188 с.

10. Rudra S., Patel M. Smart Agriculture Using IoT: A Review. In: 2021 7th Int. Conf. on Computing Communication and Automation (ICCCA). IEEE. 2021. 433–437 с.

11. Tang Y. et al. Low-Cost Wireless Sensor Network for Agricultural Monitoring. Sensors. 2022. Vol. 22(2). 455–462 с. DOI: 10.3390/s22020455

12. Arduino. Arduino Platform and IDE. URL: https://www.arduino.cc (дата звернення 01.06.2025).

13. PlatformIO. Documentation & Tools for Embedded Development. URL: https://platformio.org/ (дата звернення 01.06.2025).

14. Bluetooth SIG. Bluetooth Low Energy Technology Overview. URL: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/ (дата звернення 01.06.2025).

15. Погорілий О. В. Основи побудови автоматизованих систем на базі ESP32. Київ. ITEA. 2021. 144 с.

16. NimBLE-Arduino Library Documentation. URL: https://github.com/h2zero/NimBLE-Arduino (дата звернення 01.06.2025).

17. Шульга Д. С. Інтелектуальні сенсорні мережі для моніторингу мікроклімату. Вісник ХНУРЕ. 2020. № 4(78). 55–59 с.

18. Соловйов М. В. Принципи енергозбереження в IoT-пристроях. Системи керування та автоматика. 2022. № 2. 42–47 с.

19. Стандарт ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. URL: http://library.kname.edu.ua/opacunicode/index.php?url=/notices/index/IdNotice:95760/Source:default (дата звернення 01.06.2025).

20. Popov, V., & Bondar, A. Low-Power Wireless Monitoring Systems for Agriculture Based on BLE and ESP32. – International Journal of Advanced Computer Science and Applications (IJACSA), 2023, Vol. 14(2), pp. 76–83. – DOI:10.14569/IJACSA.2023.0140289

21. Liu, H., & Zhang, Y. Optimization Techniques for Power-Efficient Sensor Nodes in Environmental Monitoring. – Sensors, 2023, Vol. 23(4), Article ID: 1557. – DOI:10.3390/s23041557.

22. Орлов, І. М., Костенко, В. С. IoT-системи з BLE-зв'язком на ESP32: архітектура, енергозбереження, безпека. – Системи управління, навігації та зв’язку, 2024, №2 (70), С. 55–61.

23. Espressif Systems. ESP-IDF Programming Guide v5.1 – ESP32-H2. – 2024. URL: https://docs.espressif.com/projects/esp-idf/en/v5.1/esp32h2/ (дата звернення 01.06.2025).

24. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт (проєктів) для студентів спеціальностей: 126 «Інформаційні системи та технології», 151 «Автоматизація та комп’ютерно-інтегровані технології» [Електронний ресурс] / Уклад. К. В. Овчинников, О. В. Бісікало. – Вінниця : ВНТУ, 2022. – 50 с.

Додатки

Вінницький національний технічний університет
Кафедра автоматизації та інтелектуальних інформаційних технологій

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
д.т.н., професор Олег БІСІКАЛО
_____
(підпис)
«_23_» _березня_ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу
«Розробка інтелектуальної сенсорної системи моніторингу мікроклімату для автоматизованого догляду за рослинами»
08-31.БКР.013.02.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи
к.т.н., доц. кафедри АІІТ
Ілона БОГАЧ
«_23_» _травня__ 2025 р.
Розробив студент гр. АКІТР-23МС
Вадим ЧУМАК
«_23_» _травня__ 2025 р.

Вінниця ВНТУ – 2025 рік

Додаток А (обов'язковий)

Технічне завдання на бакалаврську кваліфікаційну роботу

1 Підстава для проведення робіт

Підставою для виконання бакалаврської кваліфікаційної роботи на тему: «Розробка інтелектуальної сенсорної системи моніторингу мікроклімату для автоматизованого догляду за рослинами» є наказ № 97 від 20.03.2025 р.

Термін виконання робіт:

початок 20.03.2025 р.
кінець 10.06.2025 р.

2 Мета та вихідні дані для проведення робіт

Метою бакалаврської кваліфікаційної роботи є підвищення ефективності автоматизованого моніторингу мікроклімату в системах догляду за рослинами шляхом створення інтелектуальної сенсорної системи на основі мікроконтролера ESP32-H2-MINI-1-N2, що забезпечує автономне зчитування показників температури, вологості повітря і ґрунту з передачею даних через Bluetooth Low Energy на мобільний пристрій.

Вихідними даними для проведення робіт є індивідуальне завдання на бакалаврську кваліфікаційну роботу від 20.03.2025 р.

3 Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студент групи **АКІТР-23МС Чумак Вадим Володимирович** факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Аналіз предметної області	12.03 – 29.03
Е2	Проєктування апаратної частини системи	01.04 – 26.04
Е3	Розробка програмного забезпечення мікроконтролера	29.04 – 10.05
Е4	Тестування додатку та виправлення помилок	13.05 – 24.05
Е5	Оформлення пояснювальної записки та графічної частини	27.05 – 31.05
Е6	Перевірка ПЗ на відповідність вимогам	03.06 – 10.06
Е7	Попередній захист, доопрацювання, та рецензування БКР	10.06 – 15.06
Е8	Захист БКР ЕК	16.06 – 23.06

4 Призначення і галузь застосування

Інтелектуальна сенсорна система моніторингу мікроклімату призначена для автоматизованого контролю параметрів навколишнього середовища в системах вирощування рослин. Система дозволяє операторам безперервно моніторити температуру повітря, відносну вологість та вологість ґрунту, отримувати дані в режимі реального часу через мобільний додаток, а також налаштовувати пороги сповіщень для критичних змін параметрів.

Розроблена система може застосовуватися у промисловому тепличному господарстві для оптимізації умов вирощування та підвищення врожайності, у домашньому садівництві для автоматизованого догляду за кімнатними рослинами, в наукових установах для проведення досліджень впливу мікрокліматичних факторів на ріст рослин, а також у вертикальному землеробстві для контролю параметрів в умовах обмеженого простору.

5 Технічні дані

5.1 Мікроконтролер – ESP32-H2-MINI-1-N2
5.2 Середовище розробки прошивки – Visual Studio Code з PlatformIO
5.3 Фреймворк прошивки – Arduino Core для ESP32
5.4 Сенсор температури та вологості – SHTC3-TR-10KS
5.5 Частотний генератор для вологості ґрунту – NE555DR
5.6 Протокол бездротового зв'язку – Bluetooth Low Energy (BLE 5.2)
5.7 Платформа мобільного додатку – Android 8.0 або новіша
5.8 Мова програмування мобільного додатку – Kotlin
5.9 UI фреймворк – Jetpack Compose
5.10 Джерело живлення – Li-Ion акумулятор 18650 (2000-3200 мАг)
5.11 Точність вимірювання температури – ±0.4°C
5.12 Точність вимірювання вологості повітря – ±3% RH

6 Джерела розробки

6.1 Espressif Systems. ESP32-H2 Series Datasheet [Електронний ресурс]. URL: https://www.espressif.com/sites/default/files/documentation/esp32-h2_datasheet_en.pdf (дата звернення: 10.06.2025)
6.2 Sensirion. SHTC3 Humidity and Temperature Sensor Datasheet [Електронний ресурс]. URL: https://sensirion.com/products/catalog/SHTC3/ (дата звернення: 10.06.2025)
6.3 Texas Instruments. NE555 Single Precision Timer Datasheet [Електронний ресурс]. URL: https://www.ti.com/lit/ds/symlink/ne555.pdf (дата звернення: 10.06.2025)
6.4 Google Inc. Android BLE Development Guide [Електронний ресурс]. URL: https://developer.android.com/guide/topics/connectivity/bluetooth/ble-overview (дата звернення: 10.06.2025)
6.5 PlatformIO Documentation. ESP32 Platform [Електронний ресурс]. URL: https://docs.platformio.org/en/latest/platforms/espressif32.html (дата звернення: 10.06.2025)

6.6 Arduino ESP32 Core Documentation [Електронний ресурс]. URL: https://docs.espressif.com/projects/arduino-esp32/en/latest/ (дата звернення: 10.06.2025)

Здобувач ст. гр. АКІТР-23мс _____ Вадим ЧУМАК

Додаток Б

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНТЕЛЕКТУАЛЬНА СЕНСОРНА СИСТЕМА МОНІТОРИНГУ МІКРОКЛІМАТУ ДЛЯ АВТОМАТИЗОВАНОГО ДОГЛЯДУ ЗА РОСЛИНАМИ

Блок-схема алгоритму ініціалізації системи моніторингу

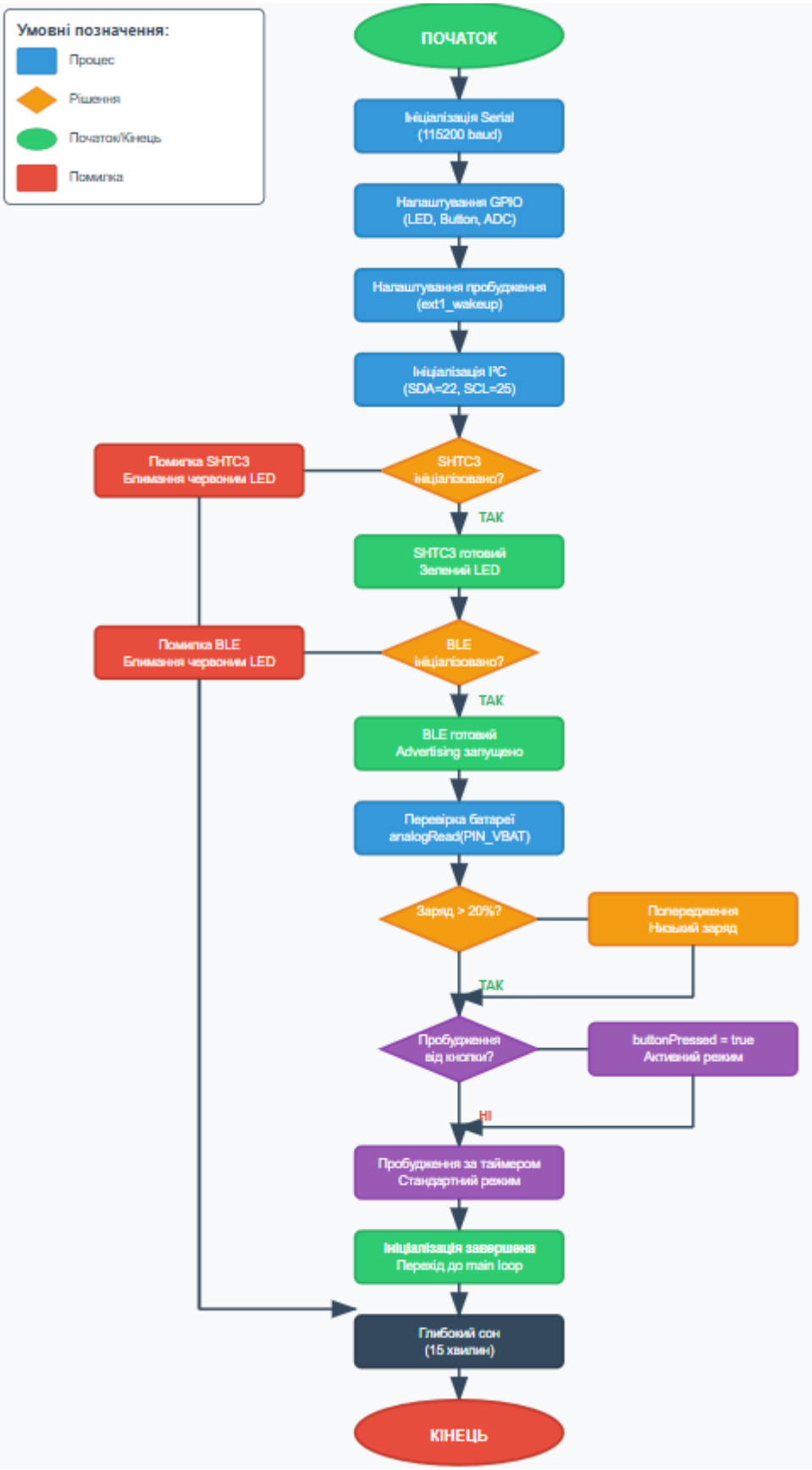

Рисунок Б.1 – Блок-схема алгоритму ініціалізації системи моніторингу

Блок-схема алгоритму роботи з сенсорами

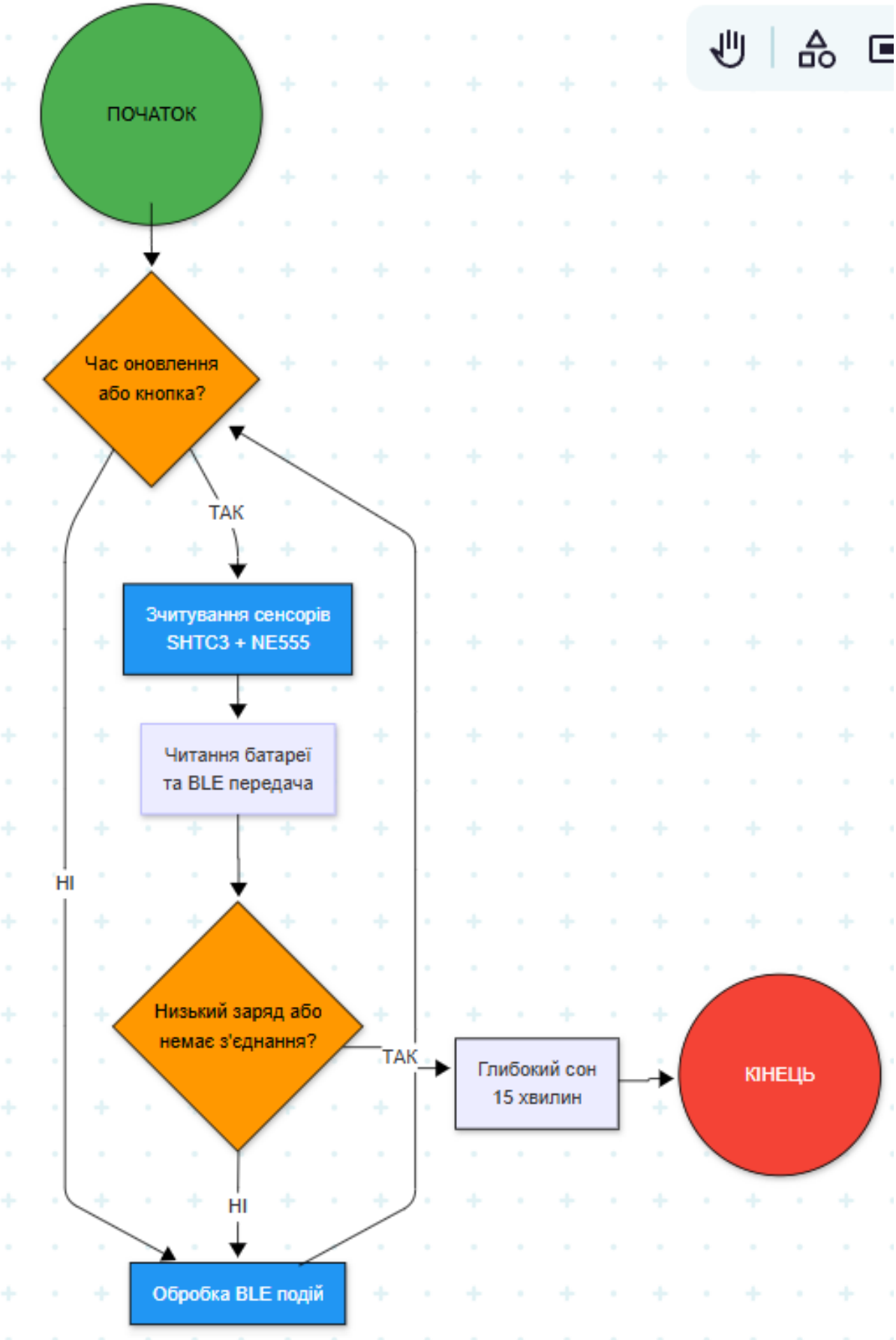

Рисунок Б.2 – Блок-схема алгоритму збору та обробки сенсорних даних

Структурна схема системи моніторингу мікроклімату

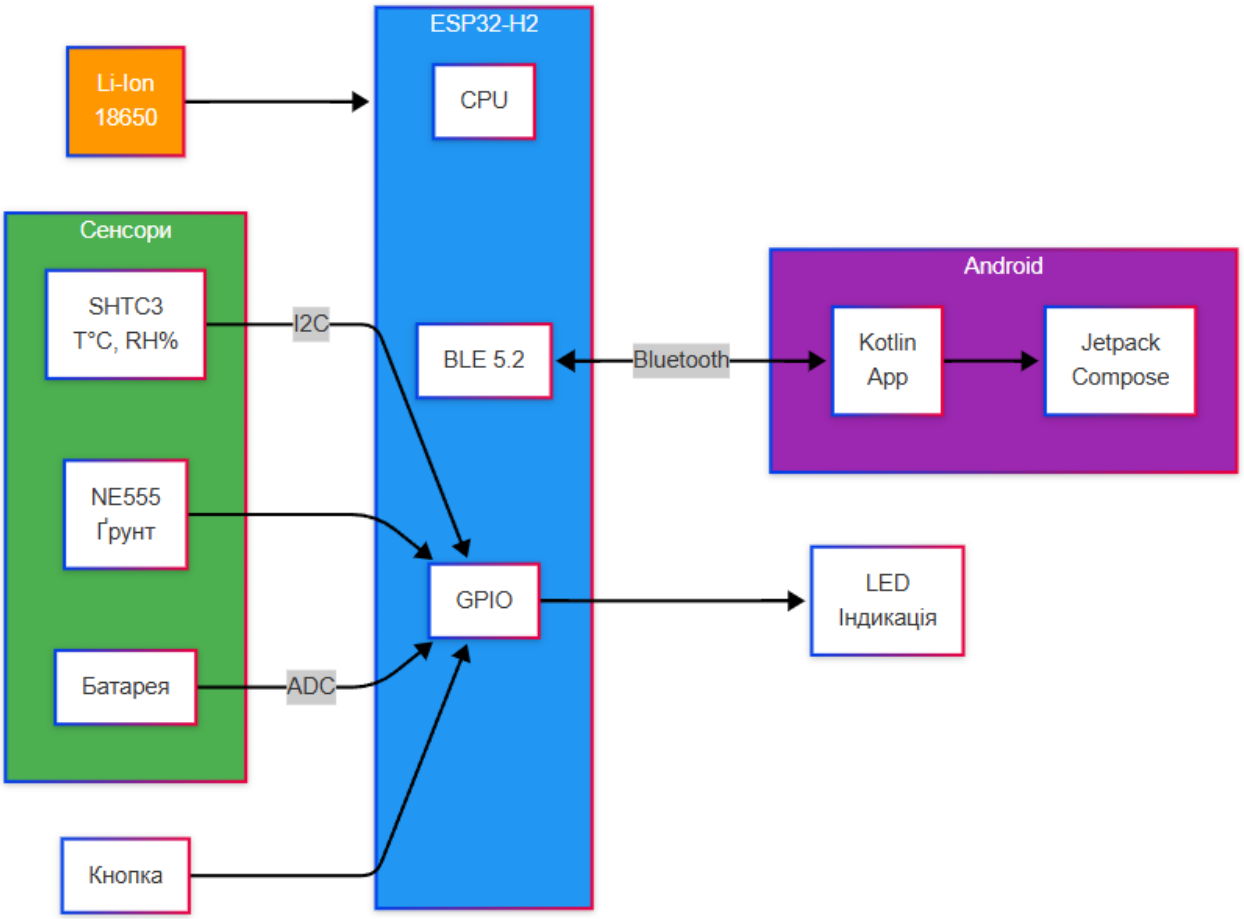

Рисунок Б.3 – Структурна схема інтелектуальної сенсорної системи

Додаток В (обов'язковий)

Лістинг основних функцій застосунку

```
// BLEManager.cpp - Основний клас для роботи з BLE на ESP32
#include "BLEManager.h"
#include "config.h"

BLEManager::BLEManager() :
    pServer(nullptr), pService(nullptr),
    tempCharacteristic(nullptr), humidityCharacteristic(nullptr),
    soilCharacteristic(nullptr), batteryCharacteristic(nullptr),
    pAdvertising(nullptr), deviceConnected(false),
    oldDeviceConnected(false), serverCallbacks(nullptr) {
}

bool BLEManager::begin() {
    Serial.println("Ініціалізуємо BLE...");

    // Ініціалізуємо BLE
    NimBLEDevice::init("ESP32-SensorMonitor");
    NimBLEDevice::setPower(ESP_PWR_LVL_P9); // Максимальна потужність

    // Створюємо сервер
    pServer = NimBLEDevice::createServer();
    if (!pServer) {
        Serial.println("Помилка створення BLE сервера!");
        return false;
    }

    // Налаштовуємо callbacks
    serverCallbacks = new ServerCallbacks(this);
    pServer->setCallbacks(serverCallbacks);

    // Налаштовуємо сервіс та характеристики
    setupService();
    setupCharacteristics();
    setupLowEnergyConnection();
    configureOptimalAdvertising();
    startAdvertising();

    Serial.println("BLE сервіс запущено!");
    return true;
}

void BLEManager::setupService() {
    // Створюємо сервіс з UUID
    pService = pServer->createService(SERVICE_UUID);
}

void BLEManager::setupCharacteristics() {
    // Створюємо характеристики для кожного типу даних
    tempCharacteristic = pService->createCharacteristic(
        TEMP_CHAR_UUID,
        NIMBLE_PROPERTY::READ | NIMBLE_PROPERTY::NOTIFY
    );

    humidityCharacteristic = pService->createCharacteristic(
        HUMIDITY_CHAR_UUID,
        NIMBLE_PROPERTY::READ | NIMBLE_PROPERTY::NOTIFY
    );
```

```
    soilCharacteristic = pService->createCharacteristic(
        SOIL_CHAR_UUID,
        NIMBLE_PROPERTY::READ | NIMBLE_PROPERTY::NOTIFY
    );

    batteryCharacteristic = pService->createCharacteristic(
        BATTERY_CHAR_UUID,
        NIMBLE_PROPERTY::READ | NIMBLE_PROPERTY::NOTIFY
    );

    // Запускаємо сервіс
    pService->start();
}

void BLEManager::update(float temperature, float humidity,
                        float soilMoisture, int batteryPercentage) {
    if (!deviceConnected) return;

    // Конвертуємо дані в строки та відправляємо
    String tempStr = String(temperature, 2);
    String humStr = String(humidity, 2);
    String soilStr = String(soilMoisture, 2);
    String battStr = String(batteryPercentage);

    tempCharacteristic->setValue(tempStr.c_str());
    tempCharacteristic->notify();

    humidityCharacteristic->setValue(humStr.c_str());
    humidityCharacteristic->notify();

    soilCharacteristic->setValue(soilStr.c_str());
    soilCharacteristic->notify();

    batteryCharacteristic->setValue(battStr.c_str());
    batteryCharacteristic->notify();

    Serial.println("BLE дані оновлено!");
}

// main.cpp - Основний цикл роботи ESP32
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_SHTC3.h>
#include "config.h"
#include "BLEManager.h"

// Глобальні об'єкти
BLEManager bleManager;
Adafruit_SHTC3 shtc3;

// Змінні для даних сенсорів
float temperature = 0.0f;
float humidity = 0.0f;
float soilMoisture = 0.0f;
int batteryPercentage = 0;
float batteryVoltage = 0.0f;

// Змінні для управління часом
unsigned long lastUpdateTime = 0;
bool buttonPressed = false;

// Функції для роботи з батареєю
float readBatteryVoltage() {
```

```
    int raw = analogRead(PIN_VBAT);
    // Враховуємо дільник напруги (47kOm та 10kOm)
    // Коефіцієнт 5.7 = (47 + 10) / 10
    return raw * (3.3f / 4095.0f) * 5.7f;
}

int calculateBatteryPercentage(float voltage) {
    if (voltage <= BATTERY_MIN_VOLTAGE) return 0;
    if (voltage >= BATTERY_MAX_VOLTAGE) return 100;
    return (int)((voltage - BATTERY_MIN_VOLTAGE) /
                (BATTERY_MAX_VOLTAGE - BATTERY_MIN_VOLTAGE) * 100.0f);
}

void enterDeepSleep(unsigned long time_us = DEEP_SLEEP_INTERVAL) {
    esp_sleep_enable_timer_wakeup(time_us);
    esp_deep_sleep_start();
}

// Функції для роботи з датчиками
bool initializeSensors() {
    Wire.begin(PIN_SDA, PIN_SCL);
    return shtc3.begin();
}

bool readAirSensor(float &temp, float &hum) {
    sensors_event_t humidity_event, temp_event;

    if (shtc3.getEvent(&humidity_event, &temp_event)) {
        temp = temp_event.temperature;
        hum = humidity_event.relative_humidity;
        return true;
    }

    return false;
}

float readSoilMoisture() {
    unsigned long highTime = pulseIn(PIN_SOIL_MOISTURE, HIGH,
                                     SOIL_MOISTURE_READ_TIMEOUT);
    unsigned long lowTime = pulseIn(PIN_SOIL_MOISTURE, LOW,
                                    SOIL_MOISTURE_READ_TIMEOUT);

    float frequency = 0.0f;
    if (highTime > 0 && lowTime > 0) {
        frequency = 1000000.0f / (highTime + lowTime);
    }

    return frequency;
}

void setup() {
    delay(1000);
    Serial.begin(115200);
    Serial.println("\n====== Плата ESP32 SensorMonitor ініціалізується ======");

    // Ініціалізуємо піни
    pinMode(PIN_LED_GREEN, OUTPUT);
    pinMode(PIN_LED_RED, OUTPUT);
    pinMode(PIN_BUTTON, INPUT_PULLUP);
    pinMode(PIN_SOIL_MOISTURE, INPUT);
    pinMode(PIN_VBAT, INPUT);

    // Налаштуваємо пробудження від кнопки
    uint64_t button_pin_mask = (1ULL << PIN_BUTTON);
```

```
    esp_sleep_enable_ext1_wakeup(button_pin_mask, ESP_EXT1_WAKEUP_ALL_LOW);

    // Ініціалізуємо датчики
    if (!initializeSensors()) {
        Serial.println("❌ Помилка ініціалізації датчика SHTC3!");
        delay(3000);
        enterDeepSleep();
        return;
    }

    Serial.println("✅ Датчик SHTC3 ініціалізовано успішно");

    // Ініціалізуємо BLE
    if (!bleManager.begin()) {
        Serial.println("❌ Помилка ініціалізації BLE!");
        delay(3000);
        enterDeepSleep();
        return;
    }

    Serial.println("✅ BLE ініціалізовано успішно");

    // Перевіряємо заряд батареї
    batteryVoltage = readBatteryVoltage();
    batteryPercentage = calculateBatteryPercentage(batteryVoltage);
    Serial.print("🔋 Напруга батареї: ");
    Serial.print(batteryVoltage);
    Serial.println(" В");

    // Перевіряємо причину пробудження
    esp_sleep_wakeup_cause_t wakeup_reason = esp_sleep_get_wakeup_cause();
    if (wakeup_reason == ESP_SLEEP_WAKEUP_EXT1) {
        Serial.println("🔄 Пробудження за кнопкою");
        buttonPressed = true;
    }

    lastUpdateTime = millis();
}

void loop() {
    unsigned long currentTime = millis();

    // Перевіряємо стан кнопки
    if (digitalRead(PIN_BUTTON) == LOW) {
        buttonPressed = true;
        digitalWrite(PIN_LED_GREEN, HIGH);
    } else {
        digitalWrite(PIN_LED_GREEN, LOW);
    }

    // Оновлюємо дані з сенсорів з інтервалом
    if (currentTime - lastUpdateTime >= SENSOR_UPDATE_INTERVAL ||
        buttonPressed) {

        Serial.println("\n🔄 Зчитуємо дані з датчиків...");

        // Зчитуємо дані з повітряного датчика
        if (readAirSensor(temperature, humidity)) {
            Serial.print("🌡 Температура: ");
            Serial.print(temperature);
            Serial.println(" °C");
```

```
            Serial.print("💧 Вологість повітря: ");
            Serial.print(humidity);
            Serial.println(" %");
        } else {
            Serial.println("❌ Помилка зчитування з SHTC3!");
        }

        // Зчитуємо дані з датчика вологості ґрунту
        soilMoisture = readSoilMoisture();
        Serial.print("🌱 Вологість ґрунту (частота): ");
        Serial.println(soilMoisture);

        // Зчитуємо заряд батареї
        batteryVoltage = readBatteryVoltage();
        batteryPercentage = calculateBatteryPercentage(batteryVoltage);

        // Оновлюємо дані в BLE
        bleManager.update(temperature, humidity, soilMoisture,
                          batteryPercentage);

        // Скидаємо прапор натискання кнопки
        buttonPressed = false;
        lastUpdateTime = currentTime;
    }

    // Обробляємо події BLE
    bleManager.handleEvents();

    // Перевіряємо, чи пора йти в глибокий сон
    if ((currentTime - lastUpdateTime >= 30000) &&
        !bleManager.isConnected() && !buttonPressed) {

        Serial.println("😴 Йдемо в глибокий сон на 15 хвилин...");
        enterDeepSleep();
    }

    delay(10);
}
}
// PlantBleManager.kt - BLE клас для Android застосунку
@Singleton
class PlantBleManager @Inject constructor(
    @ApplicationContext private val context: Context
) : BleManager(context) {

    companion object {
        private const val TAG = "PlantBLE"
        // UUID відповідають ESP32
        private val PLANT_SERVICE_UUID = UUID.fromString("12345678-1234-5678-
9012-123456789abc")
        private val TEMPERATURE_CHARACTERISTIC_UUID = UUID.fromString("87654321-
4321-8765-2109-cba987654321")
        private val HUMIDITY_CHARACTERISTIC_UUID = UUID.fromString("11111111-
2222-3333-4444-555555555555")
        private val MOISTURE_CHARACTERISTIC_UUID = UUID.fromString("66666666-
7777-8888-9999-aaaaaaaaaaaa")
        private val BATTERY_CHARACTERISTIC_UUID = UUID.fromString("bbbbbbbb-
cccc-dddd-eeee-ffffffffffff")
    }

    // StateFlow для даних з ESP32
    private val _temperature = MutableStateFlow(0f)
    val temperature: StateFlow<Float> = _temperature.asStateFlow()
```

```
    private val _soilMoisture = MutableStateFlow(0f)
    val soilMoisture: StateFlow<Float> = _soilMoisture.asStateFlow()

    private val _airHumidity = MutableStateFlow(0f)
    val airHumidity: StateFlow<Float> = _airHumidity.asStateFlow()

    private val _batteryLevel = MutableStateFlow(100)
    val batteryLevel: StateFlow<Int> = _batteryLevel.asStateFlow()

    private val _connectionState =
MutableStateFlow<ConnectionState>(ConnectionState.Disconnected)
    val connectionState: StateFlow<ConnectionState> =
_connectionState.asStateFlow()

    override fun initialize() {
        super.initialize()

        // Налаштування callback'ів для отримання даних з ESP32
        setNotificationCallback(temperatureCharacteristic).with { _, data ->
            try {
                val stringValue = data.getStringValue(0)
                stringValue?.toFloatOrNull()?.let { value ->
                    _temperature.value = value
                    Log.d(TAG, "Temperature received: $value°C")
                }
            } catch (e: Exception) {
                Log.e(TAG, "Error parsing temperature: ${e.message}")
            }
        }

        setNotificationCallback(moistureCharacteristic).with { _, data ->
            try {
                val stringValue = data.getStringValue(0)
                stringValue?.toFloatOrNull()?.let { value ->
                    _soilMoisture.value = value
                    Log.d(TAG, "Soil moisture received: $value Hz")
                }
            } catch (e: Exception) {
                Log.e(TAG, "Error parsing soil moisture: ${e.message}")
            }
        }

        setNotificationCallback(humidityCharacteristic).with { _, data ->
            try {
                val stringValue = data.getStringValue(0)
                stringValue?.toFloatOrNull()?.let { value ->
                    _airHumidity.value = value
                    Log.d(TAG, "Air humidity received: $value%")
                }
            } catch (e: Exception) {
                Log.e(TAG, "Error parsing air humidity: ${e.message}")
            }
        }

        setNotificationCallback(batteryCharacteristic).with { _, data ->
            try {
                val stringValue = data.getStringValue(0)
                stringValue?.toIntOrNull()?.let { value ->
                    _batteryLevel.value = value
                    Log.d(TAG, "Battery level received: $value%")
                }
            } catch (e: Exception) {
                Log.e(TAG, "Error parsing battery: ${e.message}")
```

```
            }
        }

        // Обсервер стану підключення
        setConnectionObserver(object : ConnectionObserver {
            override fun onDeviceConnecting(device: BluetoothDevice) {
                _connectionState.value = ConnectionState.Connecting
                Log.d(TAG, "Connecting to ESP32: ${device.address}")
            }

            override fun onDeviceConnected(device: BluetoothDevice) {
                _connectionState.value = ConnectionState.Connected
                Log.d(TAG, "Connected to ESP32: ${device.address}")
            }

            override fun onDeviceDisconnected(device: BluetoothDevice, reason:
Int) {
                _connectionState.value = ConnectionState.Disconnected
                Log.d(TAG, "Disconnected from ESP32, reason: $reason")
            }

            override fun onDeviceReady(device: BluetoothDevice) {
                _connectionState.value = ConnectionState.Connected
                Log.d(TAG, "ESP32 device ready, enabling notifications")

                // Включаємо сповіщення для всіх характеристик
                temperatureCharacteristic?.let {
enableNotifications(it).enqueue() }
                moistureCharacteristic?.let { enableNotifications(it).enqueue()
}
                humidityCharacteristic?.let { enableNotifications(it).enqueue()
}
                batteryCharacteristic?.let { enableNotifications(it).enqueue() }
            }
        })
    }

    fun connectToDevice(scannedDevice: ScannedDevice) {
        if (!hasBluetoothConnectPermission()) return

        Log.d(TAG, "Connecting to ESP32: ${scannedDevice.name}
(${scannedDevice.address})")
        stopScan()

        connect(scannedDevice.device)
            .useAutoConnect(false)
            .timeout(15000)
            .retry(3, 500)
            .enqueue()
    }
}.typography.bodyLarge,
                            color = MaterialTheme.colorScheme.outline
                        )
                    }
                }
            } else {
                LazyColumn(
                    contentPadding = PaddingValues(horizontal = 16.dp),
                    verticalArrangement = Arrangement.spacedBy(8.dp)
                ) {
                    items(scannedDevices.sortedByDescending { it.rssi }) {
device ->
                        ScannedDeviceItem(
                            device = device,
```

```
                            isConnected = connectionState ==
ConnectionState.Connected,
                            onClick = { onDeviceClick(device) }
                        )
                    }
                }
            }
        }
    }
}

@Composable
fun ConnectionStatusCard(
    connectionState: ConnectionState,
    temperature: Float,
    soilMoisture: Float,
    airHumidity: Float,
    batteryLevel: Int,
    isDeviceSleeping: Boolean,
    lastUpdateTime: String,
    modifier: Modifier = Modifier
) {
    Card(
        modifier = modifier,
        colors = CardDefaults.cardColors(
            containerColor = when (connectionState) {
                ConnectionState.Connected ->
MaterialTheme.colorScheme.primaryContainer
                ConnectionState.Connecting ->
MaterialTheme.colorScheme.secondaryContainer
                else -> MaterialTheme.colorScheme.surfaceVariant
            }
        )
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
            Row(
                verticalAlignment = Alignment.CenterVertically,
                modifier = Modifier.fillMaxWidth()
            ) {
                Icon(
                    imageVector = when (connectionState) {
                        ConnectionState.Connected ->
Icons.Default.BluetoothConnected
                        ConnectionState.Connecting ->
Icons.Default.BluetoothSearching
                        else -> Icons.Default.BluetoothDisabled
                    },
                    contentDescription = null
                )
                Spacer(modifier = Modifier.width(12.dp))
                Column(modifier = Modifier.weight(1f)) {
                    Text(
                        text = when (connectionState) {
                            ConnectionState.Connected -> "Підключено до
пристрою"
                            ConnectionState.Connecting -> "Підключення..."
                            else -> "Пристрій не підключено"
                        },
                        style = MaterialTheme.typography.titleMedium
                    )
                }
            }

            if (connectionState == ConnectionState.Connected) {
```

```
                Spacer(modifier = Modifier.height(16.dp))
                Divider()
                Spacer(modifier = Modifier.height(16.dp))

                Text(
                    text = "Поточні показники",
                    style = MaterialTheme.typography.titleSmall,
                    color = MaterialTheme.colorScheme.primary
                )

                Spacer(modifier = Modifier.height(8.dp))

                Row(
                    horizontalArrangement = Arrangement.SpaceEvenly,
                    modifier = Modifier.fillMaxWidth()
                ) {
                    SensorReading(
                        icon = Icons.Default.Thermostat,
                        label = "Температура",
                        value = "${String.format("%.1f", temperature)}°C"
                    )
                    SensorReading(
                        icon = Icons.Default.WaterDrop,
                        label = "Вологість ґрунту",
                        value = "${String.format("%.0f", soilMoisture)}%"
                    )
                    SensorReading(
                        icon = Icons.Default.Air,
                        label = "Вологість повітря",
                        value = "${String.format("%.0f", airHumidity)}%"
                    )
                    SensorReading(
                        icon = Icons.Default.Battery6Bar,
                        label = "Батарея",
                        value = "$batteryLevel%"
                    )
                }
            }
        }
    }
}

@Composable
fun SensorReading(
    icon: ImageVector,
    label: String,
    value: String,
    modifier: Modifier = Modifier
) {
    Column(
        horizontalAlignment = Alignment.CenterHorizontally,
        modifier = modifier
    ) {
        Icon(
            imageVector = icon,
            contentDescription = null,
            tint = MaterialTheme.colorScheme.primary,
            modifier = Modifier.size(24.dp)
        )
        Spacer(modifier = Modifier.height(4.dp))
        Text(
            text = value,
            style = MaterialTheme.typography.titleMedium,
            color = MaterialTheme.colorScheme.primary
```

```
        )
        Text(
            text = label,
            style = MaterialTheme.typography.bodySmall,
            color = MaterialTheme.colorScheme.onSurfaceVariant,
            maxLines = 2
        )
    }
}
// DeviceViewModel.kt - ViewModel для управління станом та логікою
@HiltViewModel
class DeviceViewModel @Inject constructor(
    private val bleManager: PlantBleManager
) : ViewModel() {

    // StateFlow безпосередньо з BleManager
    val scannedDevices: StateFlow<List<ScannedDevice>> =
bleManager.scannedDevices
    val isScanning: StateFlow<Boolean> = bleManager.isScanning
    val connectionState: StateFlow<ConnectionState> = bleManager.connectionState

    // Дані з сенсорів ESP32
    val temperature: StateFlow<Float> = bleManager.temperature
    val soilMoisture: StateFlow<Float> = bleManager.soilMoisture
    val airHumidity: StateFlow<Float> = bleManager.airHumidity
    val batteryLevel: StateFlow<Int> = bleManager.batteryLevel

    // Додаткові стани для ESP32
    private val _lastDataUpdateTime = MutableStateFlow(0L)
    val lastDataUpdateTime: StateFlow<Long> = _lastDataUpdateTime.asStateFlow()

    private val _isDeviceSleeping = MutableStateFlow(false)
    val isDeviceSleeping: StateFlow<Boolean> = _isDeviceSleeping.asStateFlow()

    init {
        // Відстежуємо оновлення даних з сенсорів
        viewModelScope.launch {
            var lastTemp = 0f
            var lastSoil = 0f
            var lastHumidity = 0f

            bleManager.temperature.collect { newTemp ->
                if (newTemp != lastTemp && newTemp != 0f) {
                    _lastDataUpdateTime.value = System.currentTimeMillis()
                    _isDeviceSleeping.value = false
                    lastTemp = newTemp
                }
            }
        }

        // Перевіряємо чи ESP32 перейшов у режим сну
        viewModelScope.launch {
            while (true) {
                delay(5000) // Перевіряємо кожні 5 секунд

                val timeSinceLastUpdate = System.currentTimeMillis() -
_lastDataUpdateTime.value
                val isConnected = connectionState.value ==
ConnectionState.Connected

                // Якщо більше 45 секунд без оновлень і підключені - ймовірно
ESP32 заснув
                if (timeSinceLastUpdate > 45000 && isConnected) {
                    _isDeviceSleeping.value = true
```

```
                } else if (timeSinceLastUpdate < 30000) {
                    _isDeviceSleeping.value = false
                }
            }
        }
    }

    fun startScan() {
        viewModelScope.launch {
            _isDeviceSleeping.value = false
            bleManager.startScan()
        }
    }

    fun stopScan() {
        viewModelScope.launch {
            bleManager.stopScan()
        }
    }

    fun connectToDevice(device: ScannedDevice) {
        viewModelScope.launch {
            _lastDataUpdateTime.value = System.currentTimeMillis()
            _isDeviceSleeping.value = false
            bleManager.connectToDevice(device)
        }
    }

    fun disconnect() {
        viewModelScope.launch {
            bleManager.disconnect()
            _isDeviceSleeping.value = false
        }
    }

    fun setLightMode(mode: String) {
        viewModelScope.launch {
            bleManager.setLightMode(mode)
        }
    }

    fun getTimeSinceLastUpdate(): String {
        val timeDiff = System.currentTimeMillis() - _lastDataUpdateTime.value
        return when {
            timeDiff < 1000 -> "щойно"
            timeDiff < 60000 -> "${timeDiff / 1000} сек тому"
            timeDiff < 3600000 -> "${timeDiff / 60000} хв тому"
            else -> "${timeDiff / 3600000} год тому"
        }
    }

    override fun onCleared() {
        super.onCleared()
        stopScan()
        disconnect()
    }
}

// Дата-класи для роботи з BLE
sealed class ConnectionState {
    object Connected : ConnectionState()
    object Connecting : ConnectionState()
    object Disconnecting : ConnectionState()
    object Disconnected : ConnectionState()
```

```
}

data class ScannedDevice(
    val name: String,
    val address: String,
    val rssi: Int,
    val device: BluetoothDevice
)
        if (batteryLevel < 20) {
            showLowBatteryAlert()
        }
    }

    private fun checkTemperatureAlert(temperature: Float) {
        val thresholds = _alertThresholds.value
        if (temperature < thresholds.minTemperature ||
            temperature > thresholds.maxTemperature) {
            showAlert("Температура поза межами норми: ${temperature}°C")
        }
    }

    private fun checkHumidityAlert(humidity: Float) {
        val thresholds = _alertThresholds.value
        if (humidity < thresholds.minHumidity ||
            humidity > thresholds.maxHumidity) {
            showAlert("Вологість повітря поза межами норми: ${humidity}%")
        }
    }

    private fun checkSoilMoistureAlert(soilMoisture: Float) {
        val thresholds = _alertThresholds.value
        if (soilMoisture < thresholds.minSoilMoisture) {
            showAlert("Низька вологість ґрунту: ${soilMoisture} Hz")
        }
    }

    private fun showAlert(message: String) {
        // Логіка показу сповіщення користувачу
        Log.w("SensorAlert", message)
    }

    private fun showLowBatteryAlert() {
        showAlert("Низький заряд батареї сенсорного блоку!")
    }
}

// Дата-класи для збереження інформації
data class SensorData(
    val temperature: Float = 0f,
    val humidity: Float = 0f,
    val soilMoisture: Float = 0f,
    val batteryLevel: Int = 100,
    val timestamp: Long = System.currentTimeMillis()
)

data class AlertThresholds(
    val minTemperature: Float = 15f,
    val maxTemperature: Float = 30f,
    val minHumidity: Float = 40f,
    val maxHumidity: Float = 80f,
    val minSoilMoisture: Float = 500f
)
```

Додаток Г (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка інтелектуальної сенсорної системи моніторингу мікроклімату для автоматизованого догляду за рослинами

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ. АІІТ, ФІІТА, АКІТР-23МС
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0,0 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☑ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Олег БІСІКАЛО, зав. каф. АІІТ (підпис)
(прізвище, ініціали, посада)

Любов ЯНЧУК, секретар. каф. АІІТ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку (підпис) Костянтин ОВЧИННИКОВ
(прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник Ілона БОГАЧ

Здобувач Вадим ЧУМАК

ТОВАРИСТВО З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ «ВАНДЕР ДЕВ ТЕЧ» (ТОВ «В.Д.Т.»)
WONDER DEVELOPMENT TECHNOLOGY (W.D.T.), ЄДРПОУ: 45139794

21030, Україна; м.Вінниця;
вул. Келецька, будинок 71
+380 96 397 53 98

За місцем вимоги

Довідка

Видана Чумаку Вадиму Володимировичу в тому, що результати, одержані ним в процесі виконання бакалаврської кваліфікаційної роботи «Розробка інтелектуальної сенсорної системи моніторингу мікроклімату для автоматизованого догляду за рослинами», а саме розроблені схемотехніка, алгоритми, інтерфейсні рішення та програмна реалізація, планується використати в розробках ТОВ «ВАНДЕР ДЕВ ТЕЧ».

Директор А.О. Турко

Товариство з обмеженою відповідальністю «ВАНДЕР ДЕВ ТЕЧ» 45139794 Україна, м.Вінниця