

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

Факультет програмних систем та прикладної математики

**Кафедра системного програмування і спеціалізованих
комп'ютерних систем**

До захисту допущено:

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___» _____ червня _____ 2026 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою

«Системне програмування та спеціалізовані комп'ютерні системи»

спеціальності 123 «Комп'ютерна інженерія»

**на тему: «Апаратно-програмний комплекс для моніторингу мікроклімату
рослин на базі мікроконтролера»**

Виконала:

студентка IV курсу, групи КВ-22

Лукашук Юлія Андріївна _____

Керівник:

доцент кафедри СПіСКС, д.т.н., доцент

Романкевич Олексій Михайлович _____

Консультант з нормконтролю:

доцент кафедри СПіСКС, к.т.н. Клятченко Я.М. _____

Рецензент:

Професор кафедри ОТ ФІОТ, д.т.н., проф.

Сергієнко Анатолій Михайлович _____

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

Студент _____

Київ – 2026 року

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

Факультет програмних систем та прикладної математики

**Кафедра системного програмування і
спеціалізованих комп'ютерних систем**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування та спеціалізовані комп'ютерні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«__» червня 2026р.

ЗАВДАННЯ

на дипломний проєкт студента

Лукашук Юлії Андріївни

1. Тема проєкту «Апаратно-програмна система фонового підсвічування для мультимедіа» керівник проєкту доц.каф. СПіСКС, д.т.н., Романкевич Олексій Михайлович, затверджені наказом по університету від 29 травня 2026 р. №1808-С..

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту:

див. ТЗ

4. Зміст пояснювальної записки

- 1) вступ;
- 2) аналіз існуючих рішень;
- 3) вибір і обґрунтування засобів реалізації;
- 4) проектування пристрою та розробка програмного забезпечення.

5. Перелік графічного матеріалу(із зазначенням обов’язкових креслеників, плакатів, презентацій тощо):

- 1) Апаратно-програмний комплекс для моніторингу мікроклімату рослин;
- 2) Програмне забезпечення мікроконтролера;
- 3) Комунікація між застосунком та девайсом;
- 4) Алгоритм навігації по меню OLED дисплея.

6. Консультанти розділів проєкту

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

7. Дата видачі завдання: 20.12.2025

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Видача завдання на дипломне проєктування	20.12.2025	Виконано
2.	Вивчення літератури за тематикою проєкту	07.01.2026	Виконано
3.	Розроблення та узгодження технічного завдання	17.02.2026	Виконано
4.	Аналіз існуючих рішень	02.03.2026	Виконано
5.	Підготовка матеріалів першого розділу дипломного проєкту	28.03.2026	Виконано
6.	Підготовка матеріалів другого та третього розділів дипломного проєкту	20.04.2026	Виконано
7.	Підготовка графічної частини дипломного проєкту	25.05.2026	Виконано
8.	Оформлення документації дипломного проєкту	03.06.2026	Виконано
9.	Попередній огляд матеріалів проєкту на кафедрі	04.06.2026	Виконано

Студент

Юлія ЛУКАЩУК

Керівник

Олексій РОМАНКЕВИЧ

АНОТАЦІЯ

Бакалаврський дипломний проєкт включає пояснювальну записку (56 стор., 15 рис., 2 табл., список використаної літератури з 14 найменувань, 3 додатків)

Об'єкт розробки – апаратно-програмний комплекс для моніторингу мікроклімату рослин на базі мікроконтролера. Комп'ютерна система дозволяє:

- здійснювати безперервний збір кліматичних показників навколишнього середовища та багатоканальний моніторинг вологості ґрунту;
- виводити поточні телеметричні дані на локальний OLED-дисплей у режимі реального часу;
- забезпечувати асинхронний мережевий обмін за протоколом WebSocket із інтегрованими механізмами сторожового таймера та автоматичного перепідключення після втрати сигналу;
- відображати аналітичні векторні графіки динаміки показників із каскадною системою історії за останні 24 години та логікою адаптивного масштабування осей під діапазон вхідних значень;
- динамічно конфігурувати перелік активних вимірювальних зондів, змінювати їхні назви та гарантовано зберігати налаштування й накопичений кеш історії у локальну пам'ять при зміні стану програми чи її закритті.

У процесі розробки були використані мови програмування C++ та Dart, кросплатформний фреймворк Flutter, асинхронний вебсервер ESPAsyncWebServer, протокол мережевого обміну WebSocket, інтерфейс I2C.

В ході розробки:

- проведено аналіз сучасних методів автоматизованого контролю параметрів життєдіяльності рослин та засобів їх технічної реалізації;
- сформульовано технічні вимоги до точності вимірювань, швидкодії та

інтерфейсу комплексу;

- розроблено десктопний застосунок з адаптивним графічним інтерфейсом;
- реалізовано алгоритми усереднення телеметрії, каскадної буферизації та фонового збереження історії з механізмом тротлінгу для зниження зносу накопичувача;
- розроблено мікропрограмне забезпечення для двоядерного мікроконтролера архітектури ESP32 з розділенням логіки інтерфейсу та мережевого стеку;

Упровадження цієї системи дозволить автоматизувати процес контролю умов вирощування кімнатних рослин, мінімізувати вплив людського фактора на догляд за ними та забезпечити точний моніторинг параметрів мікроклімату в реальному часі.

Ключові слова: МІКРОКЛІМАТ, ВОЛОГІСТЬ ҐРУНТУ, ESP32, FLUTTER, WEBSOCKET, АСИНХРОННИЙ ВЕБСЕРВЕР, ШИНА I2C, ТЕЛЕМЕТРІЯ, АПАРАТНО-ПРОГРАМНИЙ КОМПЛЕКС, СЕНСОРИ.

ABSTRACT

The bachelor's thesis project includes an explanatory note (56 pages, 15 figures, 2 tables, a list of references with 14 items, 3 appendices) The object of development is a hardware-software complex for monitoring the plant microclimate based on a microcontroller.

The computer system allows:

- to carry out continuous collection of environmental climate indicators and multichannel monitoring of soil moisture;
- to display current telemetry data on a local OLED display in real time;
- to provide asynchronous network exchange via the WebSocket protocol with integrated watchdog timer mechanisms and automatic reconnection after signal loss;
- to display analytical vector graphs of indicator dynamics with a tiered history system for the last 24 hours and adaptive axis scaling logic for the range of input values;
- to dynamically configure the list of active measuring probes, change their names, and guarantee the saving of settings and accumulated history cache into local memory upon program state changes or its closure.

In the process of development, the C++ and Dart programming languages, the Flutter cross-platform framework, the ESPAsyncWebServer asynchronous web server, the WebSocket network exchange protocol, and the I2C interface were used.

In the course of development:

- an analysis of modern methods of automated control of plant vital parameters and means of their technical implementation was conducted;
- technical requirements for measurement accuracy, speed, and the complex interface were formulated;
- a desktop application with an adaptive graphical user interface was developed;
- telemetry averaging, tiered buffering, and background history saving algorithms with a throttling mechanism to reduce drive wear were implemented;
- firmware for the dual-core microcontroller of the ESP32 architecture was

developed with the separation of interface logic and network stack;

The implementation of this system will allow automating the process of controlling indoor plant growing conditions, minimizing the impact of the human factor on their care, and ensuring accurate real-time monitoring of microclimate parameters.

Keywords: MICROCLIMATE, SOIL MOISTURE, ESP32, FLUTTER, WEBSOCKET, ASYNCHRONOUS WEB SERVER, I2C BUS, TELEMETRY, HARDWARE-SOFTWARE COMPLEX, SENSORS.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	A4	ІАЛЦ.467200.002 ТЗ	Апаратно-програмний комплекс для моніторингу мікроклімату рослин на базі мікроконтролера Технічне завдання	4		
	A4	ІАЛЦ.467200.003 ТП	Апаратно-програмний комплекс для моніторингу мікроклімату рослин на базі мікроконтролера Відомість технічного проекту	2		
	A4	ІАЛЦ.467200.004 ПЗ	Апаратно-програмний комплекс для моніторингу мікроклімату рослин на базі мікроконтролера Пояснювальна записка	56		

[illegible]

[illegible]

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ.....	2
2. ПІДСТАВА ДЛЯ РОЗРОБКИ.....	2
3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ	2
4. ДЖЕРЕЛА РОЗРОБКИ.....	2
5. ТЕХНІЧНІ ВИМОГИ.....	3
5.1 Вимоги до програмного продукту, що розробляється.....	3
5.2 Вимоги апаратного забезпечення	3
5.3 Вимоги до програмного та апаратного забезпечення користувача	3
6. ЕТАПИ РОЗРОБКИ	4

					ІАЛЦ. 467200.002 ТЗ			
Змін	Арк.	№ докум.	Підпис	Дата				
Розробив		Лукашук Ю.А.			Апаратно-програмний комплекс для моніторингу мікроклімату рослин на базі мікроконтролера Технічне завдання			
Перевірив		Романкевич О.М						
Н. контроль		Клятченко Я.М.						
Затвердив		Романкевич В.О.						
					Літ.	Аркуш	Аркушів	
							1	4
					КПІ ім. Ігоря Сікорського, ФПСІМ КВ-22			

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Апаратно-програмний комплекс для моніторингу мікроклімату рослин на базі мікроконтролера».

Галузь застосування: системи автоматизації та інтернет речей (IoT), розумне сільське господарство (Smart Agriculture), системи «розумний дім» (Smart Home), кімнатне квітникарство та екологічний моніторинг.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

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

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою розробки є створення точної, автономної та високошвидкісної системи для багатоканального контролю умов життєдіяльності рослин. Призначення роботи полягає у забезпеченні безперервного збору показників навколишнього середовища та вологості ґрунту, локальній візуалізації даних, а також їх миттєвій бездротовій передачі в кросплатформний клієнтський застосунок за допомогою асинхронного мережевого обміну (WebSockets). Комплекс призначений для автоматизації догляду за рослинами та мінімізації впливу людського фактора завдяки безперервному моніторингу телеметрії у реальному часі.

					ІАЛЦ.467200.002 ТЗ	Арк.
						2
Змін.	Арк.	№ докум.	Підпис	Дата		

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є науково-технічна література з побудови сенсорних мереж та IoT-рішень, офіційна технічна документація до мікроконтролера ESP32, специфікації протоколу WebSocket та інтерфейсу I2C, технічні описи (datasheets) до цифрових і аналогових датчиків вимірювання, документація фреймворку Flutter та мови Dart, а також результати власних тестувань апаратної стабільності шини даних та фільтрації електромагнітних завад живлення.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- безперервне опитування аналогових (вологість ґрунту) та цифрових (клімат) сенсорів із застосуванням алгоритму ковзного середнього для фільтрації апаратного шуму;
- локальне відображення поточних показників та системних статусів на графічному OLED-дисплеї;
- реалізація асинхронного вебсервера на базі мікроконтролера для підтримки постійного двостороннього з'єднання (WebSocket) з клієнтами, інтегрованими механізмами сторожового таймера та автоматичного перепідключення;
- створення десктопного клієнтського застосунку з адаптивним інтерфейсом (Sliver-архітектура) для візуалізації телеметрії у реальному часі та реактивною системою сповіщень про критичні стани ґрунту чи клімату;
- реалізація багаторівневої каскадної системи збереження історії вимірювань (від 1 хвилини до 24 годин) з відображенням на оптимізованих векторних графіках та логікою динамічного масштабування осей;
- інтеграція застосунку з операційною системою для забезпечення

					ІАЛЦ.467200.002 ТЗ	Арк.
						3
Змін.	Арк.	№ докум.	Підпис	Дата		

фоновій роботи через системний трей із динамічним контекстним меню моніторингу;

- наявність інтерфейсу для конфігурації мережевої адреси, індивідуального найменування робочих зон та програмного маскування непідключених датчиків зі збереженням налаштувань і кешу історії у локальній пам'яті при зміні стану вікна;

5.2 Вимоги апаратного забезпечення

- мікроконтролер архітектури ESP32 з підтримкою Wi-Fi;
- блок цифрових датчиків для вимірювання температури, вологості, атмосферного тиску та рівня освітленості;
- блок аналогових ємнісних датчиків вологості ґрунту;
- OLED-дисплей для локального виводу інформації;
- апаратна підсистема фільтрації живлення для стабілізації роботи цифрових датчиків під час активності радіомодуля;
- підключення до локальної Wi-Fi мережі.

5.3 Вимоги до програмного та апаратного забезпечення користувача

- персональний комп'ютер (ОС Windows/macOS/Linux) або мобільний пристрій, здатний запускати застосунки, скомпільовані за допомогою фреймворку Flutter;
- локальна Wi-Fi мережа для забезпечення безперешкодного обміну даними між апаратною частиною та пристроєм користувача.

					ІАЛЦ.467200.002 ТЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		4

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
1.	Пошук та ознайомлення з літературою	07.01.2026
2.	Розробка та узгодження технічного завдання	17.02.2026
3.	Пошук та аналіз існуючих рішень	02.03.2026
4.	Підготовка матеріалів до першого розділу дипломного проєкту	28.03.2026
5.	Підготовка матеріалів до другого розділу дипломного проєкту	20.04.2026
6.	Підготовка графічної частини дипломного проєкту	25.05.2026
7.	Оформлення звіту та документації дипломного проєкту	03.06.2026
8.	Попередній огляд матеріалів дипломної роботи на кафедрі	04.06.2026

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	№ прим.	Примітки	
	A4	ІАЛЦ.467200.004 ПЗ	Апаратно-програмний комплекс для моніторингу мікроклімату рослин на базі мікроконтролера.	56			
			Пояснювальна записка				
	A4	ІАЛЦ.467200.005 Е1	Взаємодія компонентів АПК.	1			
			Схема електрична структурна				
	A4	ІАЛЦ.467200.006 Д2	Алгоритм калібрування датчика вологості ґрунту.	1			
			Схема алгоритму				
	A4	ІАЛЦ.467200.007 Д3	Архітектура програмного забезпечення комплексу.	1			
			Схема структурна				
	A4	ІАЛЦ.467200.008 Д4	Алгоритм навігації по меню OLED дисплея.	1			
			Схема алгоритму				
			ІАЛЦ.467200.003 ТП				
Змін.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмний комплекс для моніторингу мікроклімату рослин на базі мікроконтролера Відомість технічного проєкту		
Розробив	Лукашук Ю.А.						
Перевірив	Романкевич О.М.						
Консульт.							
Н. контроль	Клятченко Я.М.						
Зав. каф.	Романкевич В.О.						
					Літ.	Аркуш	Аркушів
						1	2
					КПІ ім. Ігоря Сікорського, ФПСІМ КВ-22		

[illegible]

Пояснювальна записка до дипломного проєкту

на тему: Апаратно-програмний комплекс для моніторингу мікроклімату
рослин на базі мікроконтролера

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ.....	5
ВСТУП.....	6
1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ	7
1.1 Загальна характеристика задачі моніторингу мікроклімату та вологості ґрунту рослин.....	7
1.2 Аналіз існуючих комерційних та відкритих систем.....	8
1.2.1 Комерційні пристрої закритого типу	8
1.2.2 Відкриті DIY-комплекси на базі класичних мікроконтролерів	9
1.2.3 Системи на базі зовнішніх IoT-серверів	9
1.3. Порівняльна характеристика мережевих та апаратних показників.....	10
1.4 Узагальнення результатів аналізу та обґрунтування теми дипломного проєкту	11
2. ВИБІР І ОБҐРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ	13
2.1 Мікроконтролер ESP32	13
2.1.1 Використання ESP32 для високошвидкісних мережевих IoT-систем..	13
2.1.2 Порівняння з альтернативними апаратними платформами.....	14
2.2 Вимірювальні сенсори та інтерфейси зв'язку	16
2.2.1 Датчики вологості ґрунту.....	16
2.2.2 Цифрові сенсори мікроклімату.....	18
2.3 Засоби локальної візуалізації даних	20

					ІАЛЦ.467200.004 ПЗ		
Зм	Лист	№ докум.	Підп.	Дата			
Розроб.		Лукашук Ю.А.			Апаратно-програмний комплекс для моніторингу мікроклімату рослин на базі мікроконтролера Відомість технічного проєкту	Літ.	Лист
Перев.		Романкевич О.М.					2
							56
Н. контр.		Клятченко Я. М.				КПІ ім. Ігоря Сікорського, ФПСМ КВ-22	
Затв.		Романкевич В.О					

2.4 Джерело живлення та підсистеми фільтрації завад.....	22
2.4.1 Вибір джерела живлення.....	22
2.4.2 Підсистема апаратної фільтрації завад та захисту шин даних.....	22
2.5 Технології мережевого обміну	23
2.6 Середовища розробки.....	25
2.6.1 Середовище розробки для мікроконтролера ESP32.....	25
2.6.2 Середовище розробки кросплатформного клієнтського застосунку....	26
2.7 Бібліотеки та фреймворки	27
2.7.1 Бібліотеки для мікроконтролера ESP32.....	27
2.7.2 Інструментарій клієнтського застосунку та фреймворк Flutter	28
3. ПРОЄКТУВАННЯ ПРИСТРОЮ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	31
3.1 Апаратна реалізація та підключення компонентів	31
3.1.1 Схема підключення мікроконтролера ESP32, цифрової шини I2C та аналогових зондів.....	31
3.1.2 Організація живлення та апаратний захист від імпульсних завад радіомодуля	32
3.2 Розробка мікропрограмного забезпечення ESP32	33
3.2.1 Організація багатозадачності та скінченний автомат локального інтерфейсу.....	34
3.2.2 Алгоритми цифрової фільтрації та стабілізація аналогових показань..	36
3.2.3 Асинхронний мережевий стек та динамічна серіалізація телеметрії...	38
3.2.4 Інтерактивна система локального калібрування та енергонезалежна пам'ять	39
3.3 Розробка клієнтського десктопного застосунку	40

3.3.1 Архітектурна структура застосунку на базі фреймворку Flutter	40
3.3.2 Реалізація служби безперервного бездротового зв'язку	43
3.3.3 Алгоритми обробки телеметрії та багаторівнева система буферизації даних.....	44
3.3.4 Розробка адаптивного графічного інтерфейсу користувача	46
3.3.5 Аналітична візуалізація даних та оптимізація рендерингу	48
3.4 Тестування та результати роботи програмно-апаратного комплексу	49
3.4.1 Робота апаратної частини та інтеграція датчиків	49
3.4.2 Результати роботи десктопного застосунку та візуалізація	50
ВИСНОВКИ.....	53
СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....	55

ДОДАТКИ

Додаток А. Копії графічного матеріалу

ІАЛЦ.467200.005 Е1. Взаємодія компонентів АПК. Схема електрична структурна

ІАЛЦ.467200.006 Д2. Алгоритм калібрування датчика вологості ґрунту. Схема алгоритму

ІАЛЦ.467200.007 Д3. Архітектура програмного забезпечення комплексу. Схема структурна

ІАЛЦ.467200.008 Д4. Алгоритм навігації по меню OLED дисплея Схема алгоритму

Додаток Б. Фрагменти коду

Додаток В. Презентація

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		3

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

АЦП – аналого-цифровий перетворювач.

API – інтерфейс прикладного програмування (Application Programming Interface).

BLE – технологія бездротового зв'язку з низьким енергоспоживанням (Bluetooth Low Energy).

DIY – напрямок самостійного конструювання (Do It Yourself).

GATT – загальний профіль атрибутів (Generic Attribute Profile).

GPIO – інтерфейс вводу/виводу загального призначення (General-Purpose Input/Output).

HTTP – протокол передачі гіпертексту (HyperText Transfer Protocol).

I2C – послідовна шина даних для зв'язку інтегральних схем (Inter-Integrated Circuit).

IDE – інтегроване середовище розробки (Integrated Development Environment).

IoT – інтернет речей (Internet of Things).

IP – міжмережевий протокол (Internet Protocol).

JSON – текстовий формат обміну даними (JavaScript Object Notation).

MQTT – протокол обміну повідомленнями (Message Queuing Telemetry Transport).

OLED – органічний світлодіод (Organic Light-Emitting Diode).

SCL – лінія тактування шини (Serial Clock).

SDA – лінія даних шини (Serial Data).

TCP – протокол керування передачею (Transmission Control Protocol).

Wi-Fi – технологія бездротової локальної мережі (Wireless Fidelity).

ВСТУП

Забезпечення оптимальних умов життєдіяльності кімнатних та декоративних рослин вимагає безперервного контролю параметрів навколишнього середовища. Відхилення показників вологості ґрунту, температури, атмосферного тиску та освітленості від норми призводять до порушення фізіологічних процесів та загибелі рослин. Традиційний візуальний огляд та ручне вимірювання не забезпечують необхідної точності та своєчасності отримання даних. Застосування апаратно-програмних комплексів автоматизованого моніторингу вирішує цю проблему шляхом безперервного збору об'єктивних телеметричних показників.

Технічна реалізація багатоканальних систем збору даних пов'язана з низкою апаратних та алгоритмічних проблем. Інтеграція чутливих аналогових зондів вологості та цифрових сенсорів на шині I2C вимагає забезпечення стабільного живлення та захисту від електромагнітних завад. Багато існуючих Internet of Things (IoT) рішень використовують синхронні алгоритми мережевого обміну на базі HyperText Transfer Protocol (HTTP) запитів. Такий підхід створює надмірне навантаження на процесорне ядро, викликає блокування основного циклу опитування сенсорів та призводить до суттєвих затримок у відображенні інформації на стороні клієнта.

Спроба мінімізувати навантаження на мікроконтролер з одночасним збереженням високої частоти оновлення даних вимагає переходу від класичної архітектури запит-відповідь до використання протоколів повнодуплексного асинхронного зв'язку. Застосування мікроконтролерів із підтримкою технології WebSocket та асинхронних вебсерверів дає можливість обробляти мережеві події у фоновому режимі без впливу на швидкість зчитування периферії. Комбінація цих підходів забезпечує високу стабільність системи, захист від апаратних зависань та миттєву реакцію інтерфейсу на зміну параметрів мікроклімату.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Загальна характеристика задачі моніторингу мікроклімату та вологості ґрунту рослин

Процес моніторингу параметрів життєдіяльності рослин належить до категорії систем збору телеметричних даних у реальному часі. Базовими фізичними величинами, що безпосередньо керують інтенсивністю фотосинтезу, транспірацією та кореневим водопоглинанням, є температура повітря, атмосферний тиск, рівень освітленості та об'ємна вологість ґрунту в зоні кореневої системи. Для забезпечення оптимальних умов вирощування система повинна здійснювати безперервний збір цих показників, їх первинну алгоритмічну обробку та миттєву трансляцію на пристрій користувача.

Технічна реалізація багатоканальних систем збору телеметрії пов'язана з проблемою апаратного конфлікту ресурсів під час одночасного обслуговування різних цифрових та аналогових інтерфейсів. Зчитування слабких аналогових сигналів із зондів вологості ґрунту через аналого-цифровий перетворювач (АЦП) вимагає стабільної опорної напруги та відсутності електромагнітних завад на вимірювальній шині. Паралельне опитування цифрових кліматичних сенсорів по шині I2C вимагає забезпечення жорстких таймінгів для ліній Serial Data (SDA) та Serial Clock(SCL).

У стандартних однопотокових архітектурах мікроконтролерів (наприклад, платформ Arduino) циклічне виконання цих завдань призводить до блокування системного переривання, що відповідає за роботу мережевого стеку Wi-Fi. Це викликає переповнення буферів прийому-передачі, втрату мережевих пакетів та виникнення непередбачуваних мікрозависань радіомодуля (Watchdog Timeout).

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		7

1.2 Аналіз існуючих комерційних та відкритих систем

Ринок пристроїв для домашнього квітництва та смарт-агрономії можна розділити на три основні категорії, кожна з яких має суттєві обмеження у контексті розбудови швидкодіючих та масштабованих систем.

1.2.1 Комерційні пристрої закритого типу

Найбільш розповсюдженими представниками комерційного сегменту є інтегровані датчики типу Xiaomi Mi Flora [1]. Це компактні автономні пристрої (рисунок 1.1), які поєднують ємнісний датчик вологості, термістор та фотодіод у єдиному пластиковому корпусі. Передача телеметрії здійснюється за допомогою бездротового стеку Bluetooth Low Energy (BLE) через стандартизований профіль GATT (Generic Attribute Profile)[2].



Рисунок 1.1 Xiaomi Mi Flora

Головним недоліком такої архітектури є топологія підключення мережі типу «точка-точка» (Point-to-Point). Для інтеграції BLE-пристрою в локальну IP-мережу користувача (Wi-Fi) та забезпечення віддаленого моніторингу вимагається встановлення виділеного апаратного шлюзу ретрансляції (BLE Gateway), що підвищує загальну вартість системи. Крім того, апаратна база

пристрою обмежує вимірювання лише однією локальною зоною (один горщик). Опитування багатоканальної системи з кількох рослин вимагає придбання окремого пристрою з незалежним джерелом живлення для кожної з них, що унеможлиблює економічно виправдане масштабування.

1.2.2 Відкриті DIY-комплекси на базі класичних мікроконтролерів

Альтернативою комерційним рішенням є аматорські системи автоматизації, що здебільшого реалізуються на базі 8-бітних мікроконтролерів сімейства AVR (Arduino Uno, Nano) або одноядерних Wireless Fidelity (Wi-Fi) модулів першого покоління (ESP8266). Аналіз таких рішень виявляє критичні недоліки на рівні апаратного забезпечення.

По-перше, для зниження вартості масово використовуються резистивні датчики вологості ґрунту. Їх принцип дії базується на вимірюванні опору між двома відкритими струмопровідними електродами. Під час проходження постійного струму через вологий ґрунт виникає інтенсивний електроліз, що призводить до анодного окислення та повного руйнування металевих контактів датчика [3].

По-друге, мікроконтролер ESP8266 має лише один доступний канал АЦП (пін TOUT), здатний обробляти напругу лише до 1 В [4]. Для реалізації багатоканального моніторингу виникає потреба у застосуванні зовнішніх аналогових мультиплексорів. Впровадження мультиплексора вносить додаткові затримки на перемикання адресних пінів, збільшує паразитну ємність шини та рівень нелінійних спотворень аналогового сигналу, що знижує точність вимірювань.

1.2.3 Системи на базі зовнішніх IoT-серверів

Третій популярний підхід полягає у використанні мікроконтролерів як трансиверів, що збирають дані та відправляють їх на зовнішні IoT-сервери (Blynk, Tuuya, ThingSpeak) за протоколами прикладного рівня, такими як HTTP

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		9

або Message Queuing Telemetry Transport (MQTT). Основною проблемою цього підходу є високі накладні витрати на виконання повної процедури встановлення Transmission Control Protocol (TCP) з'єднання та передачу важких текстових заголовків для кожного сеансу зв'язку.

Сформований пакет даних проходить складний маршрут: від мікроконтролера до маршрутизатора, далі через серію провайдерських вузлів до віддаленого дата-центру хмарної платформи, і лише потім повертається на пристрій клієнта. Така маршрутизація генерує затримку обміну даними (Ping) на рівні 150–300 мс, унеможливаючи створення інтерфейсів реального часу [5].

Важливим недоліком є також критична вразливість системи: втрата з'єднання з мережею Інтернет призводить до повної зупинки процесу моніторингу, навіть за умови функціонуючої локальної Wi-Fi мережі.

1.3. Порівняльна характеристика мережевих та апаратних показників

Для обґрунтування оптимального інженерного шляху розробки було проведено порівняльний аналіз технічних показників вищепованих систем та запропонованого у дипломному проєкті комплексу. Результати зведені в таблицю 1.1.

Таблиця 1.1 Порівняння архітектур систем моніторингу мікроклімату

Технічний показник	Комерційні BLE-сенсори	DIY + Хмарні IoT-сервери	Запропонований апаратно-програмний комплекс
Мережева топологія	Точка-точка (GATT)	Через зовнішній сервер	Локальна асинхронна
Середня мережева затримка	Потребує ручної синхронізації	150-300 мс	Менше 50 мс
Багатоканальність (базова плата)	1 пристрій = 1 зона	Потребує зовнішніх мікросхем	До 6 зон вимірювання вологості ґрунту і одна для спільних показників

Технічний показник	Комерційні BLE-сенсори	DIY + Хмарні IoT-сервери	Запропонований апаратно-програмний комплекс
Тип сенсорів вологості ґрунту	Ємнісні	Резистивні	Ємнісні
Залежність від зовнішнього Інтернету	Відсутня прямий зв'язок	Повна залежність	Відсутня

1.4 Узагальнення результатів аналізу та обґрунтування теми дипломного проєкту

Аналіз існуючих рішень виявляє критичний компроміс між вартістю, масштабованістю та швидкодією. Комерційні екосистеми є економічно неефективними для багатоканального моніторингу через закриту архітектуру та топологію «один сенсор - один горщик». Водночас відкриті DIY-проєкти на базі однопоточних мікроконтролерів (ESP8266, AVR) та зовнішніх IoT-серверів мають низьку апаратну відмовостійкість і генерують високі мережеві затримки, що унеможлиблює моніторинг у реальному часі. Усунення цих недоліків вимагає розробки концептуально нового апаратно-програмного комплексу, що поєднує бюджетність із сучасними асинхронними архітектурами.

Обґрунтування апаратної бази проєкту спирається на необхідність забезпечення багатоканальності без ускладнення схемотехніки. Використання мікроконтролера з двопоточною архітектурою та кількома незалежними модулями АЦП (сімейство ESP32) дозволить одночасно опитувати до шести аналогових зондів вологості ґрунту без застосування зовнішніх мультиплексорів. Відмова від дешевих резистивних датчиків на користь ємнісних обґрунтовується їхньою фізичною будовою: мідні обкладки ізольовані від вологого середовища, що повністю виключає явище електролітичної корозії та гарантує тривалий термін експлуатації.

Комплексний аналіз умов життєдіяльності рослин вимагає одночасного використання різних типів датчиків (аналогових для ґрунту та цифрових для

клімату). Для забезпечення загальної відмовостійкості системи критично важливим є впровадження жорстких апаратних таймаутів для цифрової шини I2C. Це запобігатиме блокуванню процесора у разі виникнення електромагнітних завад під час трансляції даних Wi-Fi модулем.

Вибір мережевої архітектури диктується вимогою мінімізації затримок. Використання класичних синхронних HTTP-запитів до хмарних серверів є неефективним через високі накладні витрати на встановлення TCP-з'єднань. Перехід до локального асинхронного обміну даними за допомогою повнодуплексного протоколу WebSocket забезпечить миттєву трансляцію телеметрії безпосередньо на пристрій користувача. Такий підхід гарантує автономність комплексу та його незалежність від стабільності глобальної мережі Інтернет.

Застосування кросплатформного фреймворку (наприклад, Flutter) для розробки клієнтського застосунку обґрунтовується необхідністю правильного перерозподілу обчислювального навантаження. Замість виконання ресурсоемних циклів фільтрації на мікроконтролері, доцільно делегувати логіку динамічного маскування неактивних датчиків графічному інтерфейсу користувача. Додаткова відмовостійкість архітектури буде реалізована за рахунок паралельного виведення даних: телеметрія має одночасно відображатися на локальному Organic Light-Emitting Diode (OLED) дисплеї пристрою та транслюватися у застосунок. Це забезпечить безперервний контроль мікроклімату навіть у випадку тимчасової відсутності зв'язку з клієнтським програмним забезпеченням.

Комбінація надійної апаратної бази, швидкодіючого локального мережевого протоколу та оптимізованого кросплатформного програмного забезпечення доводить науково-практичну цінність обраної теми та є обґрунтованим шляхом для створення бюджетної системи професійного рівня.

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		12

2. ВИБІР І ОБҐРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Мікроконтролер ESP32

2.1.1 Використання ESP32 для високошвидкісних мережевих IoT-систем

Базовою вимогою до апаратного ядра системи безперервного моніторингу є здатність паралельно обслуговувати кілька незалежних інтерфейсів без блокування мережевого стеку. У системах, що використовують протоколи передачі даних у реальному часі, затримка в опитуванні периферії (наприклад, під час багаторазового аналогового зчитування для медіанної фільтрації) призводить до переповнення буферів Wi-Fi та втрати пакетів.

Використання мікроконтролера ESP32 на базі двоядерної архітектури Xtensa 32-bit LX6 вирішує цю проблему на апаратному рівні [6]. Хоча розробка мікропрограми може вестися у стандартному однопотоковому середовищі, базовий фреймворк автоматично розгортається поверх операційної системи реального часу FreeRTOS. Це дозволяє системі ізолювати процеси: стек протоколів Wi-Fi та TCP/IP обробляється на нульовому ядрі, тоді як зчитування датчиків та логіка користувача виконуються на першому ядрі. Така апаратна розпаралеленість гарантує стабільну роботу асинхронного вебсервера. Блок-схему апаратної архітектури мікроконтролера наведено на рисунку 2.1.

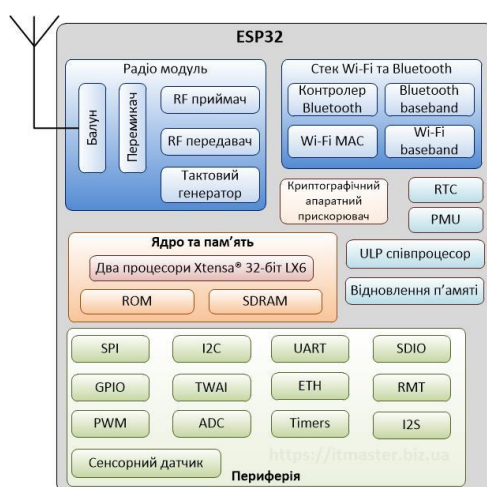


Рисунок 2.1 Блок-схема апаратної архітектури мікроконтролера ESP32

Критичним фактором для багатоканального контролю вологості ґрунту є підсистема аналого-цифрового перетворення. Мікроконтролер ESP32 має два незалежні апаратні модулі багатоканальних SAR АЦП з розрядністю 12 біт.

Важливою особливістю є те, що використання АЦП1 не конфліктує з роботою Wi-Fi модуля, на відміну від АЦП2. Використання 12-бітного підвищує точність реєстрації мінімальних змін ємності зондів порівняно зі стандартними 10-бітними архітектурами. Наявність шести вільних каналів АЦП1 робить можливим пряме підключення всіх аналогових датчиків вологості без застосування мультиплексорів. Відмова від мікросхем-комутаторів зменшує площу друкованої плати та виключає виникнення паразитних ємностей і нелінійних спотворень сигналу.

Також мікроконтролер забезпечує апаратну підтримку цифрової шини I2C з можливістю налаштування таймаутів на апаратному рівні. Це дозволяє уникати нескінченних циклів очікування відповіді у випадку збоїв на лінії живлення датчиків.

2.1.2 Порівняння з альтернативними апаратними платформами

Доцільність вибору ESP32 підтверджується шляхом контрастного аналізу технічних характеристик альтернативних рішень, що розповсюджені у сфері вбудованих систем: 8-бітних мікроконтролерів сімейства AVR, одноядерних Wi-Fi модулів ESP8266 та продуктивних платформ на базі ARM Cortex-M3. Порівняльні характеристики платформ зведено у таблицю 2.1.

Таблиця 2.1 - Порівняння технічних характеристик різних мікроконтролерів

Характеристика				
Архітектура та ядра	8-біт, 1 ядро	32-біт, 1 ядро	32-біт (ARM), 1 ядро	32-біт, 2 ядра

Тактова частота	16 МГц	80 / 160 МГц	72 МГц	160 / 240 МГц
Кількість каналів АЦП				
Розрядність АЦП	10 біт	10 біт	12 біт	12 біт
Оперативна пам'ять	2 КБ	50 КБ	20 КБ	520 КБ
Вбудований Wi-Fi	Немає	Є	Немає	Є
Необхідність мультиплексора	Ні	Так	Ні	Ні

Аналіз даних з таблиці демонструє, що платформа ATmega328P є непридатною через малий обсяг оперативної пам'яті (2 КБ), якої недостатньо для формування пакетів формату JavaScript Object Notation (JSON) та підтримки мережесесій [7]. Відсутність інтегрованого Wi-Fi вимагає підключення зовнішніх трансиверів, що створює вузьке місце у пропускій здатності.

Мікроконтролер ESP8266 володіє вбудованим Wi-Fi модулем, проте має критичні архітектурні обмеження для даної задачі. Одноядерна архітектура призводить до того, що під час тривалого опитування периферійних датчиків обробка мережевого стеку зупиняється. Крім того, ESP8266 містить лише один вивід АЦП. Реалізація опитування шести зондів вологості на цій базі вимагала б обов'язкової інтеграції аналогового мультиплексора та додаткових резистивних дільників напруги, оскільки максимальна вимірювана напруга на піні TOUT обмежена рівнем 1 В.

Лінійка мікроконтролерів STM32F103 володіє продуктивним обчислювальним ядром та швидкісними 12-бітними АЦП з часом перетворення близько 1 мкс [8]. За апаратними можливостями перетворення сигналів ця платформа перевершує аналоги, проте відсутність вбудованого радіомодуля вимагає реалізації мережевої взаємодії через окремий сопроцесор. Такий підхід

значно ускладнює схемотехніку пристрою, підвищує кінцеву вартість та створює проблеми синхронізації двох обчислювальних модулів.

Отже, ESP32 є єдиним збалансованим рішенням, яке поєднує достатню кількість аналогових каналів високої роздільної здатності для роботи без мультиплексорів, апаратну підтримку бездротових мереж та багатоядерну архітектуру, що унеможливорює виникнення мережових затримок під час збору телеметрії.

2.2 Вимірювальні сенсори та інтерфейси зв'язку

2.2.1 Датчики вологості ґрунту

Задача вимірювання об'ємної вологості ґрунту вимагає забезпечення тривалого терміну експлуатації чутливого елемента в агресивному середовищі з високим вмістом мінеральних солей та води. Традиційним підходом у бюджетних системах моніторингу є використання резистивних датчиків на кшталт FC-28 (рисунок 2.2).

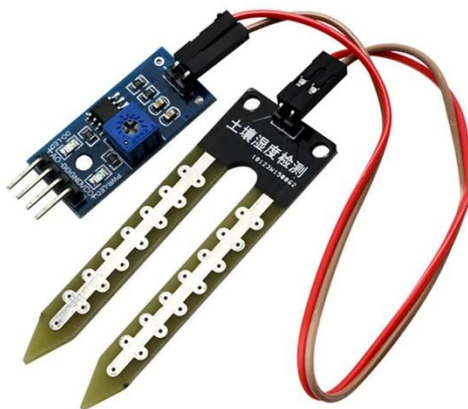


Рисунок 2.2 Резистивний датчик вологості ґрунту FC-28

Принцип їхньої дії базується на вимірюванні електричного опору між двома відкритими електродами під час пропускання постійного струму. Основним недоліком такого методу є інтенсивний електроліз металу: під дією

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		16

струму у водному розчині анод швидко окислюється і руйнується, що призводить до значної деградації показників та повної відмови датчика за 3–5 тижнів експлуатації.

Для усунення проблеми електролітичної корозії у розроблюваному комплексі застосовано аналогові ємнісні зонди вологості ґрунту – HW390 (рисунок 2.3).

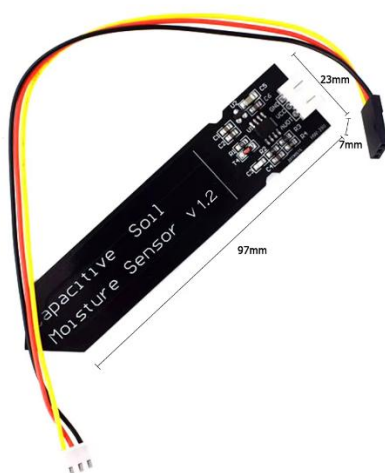


Рисунок 2.3 Ємнісний датчик вологості ґрунту HW390

Принцип їх роботи базується на вимірюванні діелектричної проникності середовища. Мідні обкладки датчика, що формують конденсатор, повністю вкриті ізолюючою паяльною маскою і не мають прямого гальванічного контакту з ґрунтом. Вбудований на платі таймер (генератор імпульсів) формує високочастотний сигнал, амплітуда якого на виході інтегруючого ланцюга змінюється залежно від реактивної ємності системи зонд-ґрунт.

Таке інженерне рішення дозволяє повністю уникнути протікання постійного струму через ґрунт. Датчик формує вихідну аналогову напругу в діапазоні від 1.2 В (сухе повітря) до 3.0 В (занурення у воду), яка безпосередньо подається на входи 12-бітного АЦП мікроконтролера ESP32. Використання 12-бітного квантування забезпечує роздільну здатність у 4096 кроків, що гарантує

високу точність реєстрації навіть незначних змін вологості. Схему принципу дії та ізоляції ємнісного зонда наведено на рисунку 2.4.

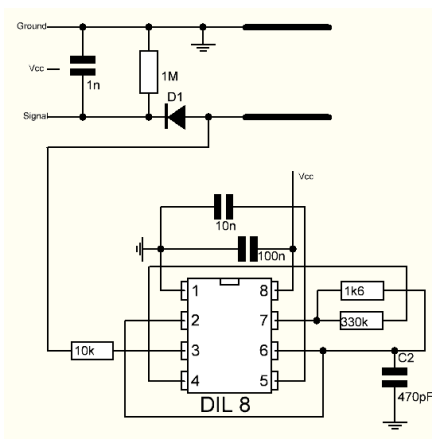


Рисунок 2.4 Принципова схема та будова ємнісного датчика вологості ґрунту

2.2.2 Цифрові сенсори мікроклімату

Контроль параметрів навколишнього середовища (температури, вологості повітря, атмосферного тиску та освітленості) вимагає збору даних від декількох різнотипних сенсорів. Використання датчиків із пропрієтарними однопровідними інтерфейсами (наприклад, популярних модулів DHT11 або DHT22 для вимірювання температури та вологості) вимагає від мікроконтролера забезпечення жорстких мікросекундних таймінгів програмним шляхом. В умовах роботи операційної системи FreeRTOS та активної трансляції пакетів радіомодулем Wi-Fi програмна реалізація протоколу 1-Wire часто переривається системними перериваннями з вищим пріоритетом. Це призводить до спотворення бітової послідовності під час зчитування та втрати фреймів даних.

Для гарантування надійності апаратної частини усі кліматичні сенсори комплексу об'єднано на єдиній цифровій шині I2C. Використання апаратного контролера I2C всередині ESP32 дозволяє здійснювати опитування датчиків на тактовій частоті до 400 кГц (режим Fast Mode) паралельно з виконанням

основного коду, не залучаючи ресурси центрального процесора на побітове формування сигналів.

Вимірювання температури та відносної вологості повітря реалізовано на базі інтегрованого цифрового датчика АНТ20. На відміну від серії DHT, сенсор АНТ20 містить вбудований калібрований АЦП і підтримує стандартний протокол I2C, гарантуючи похибку вимірювання температури не більше $\pm 0.3\text{ }^{\circ}\text{C}$ та вологості $\pm 2\%$ [9].

Атмосферний тиск вимірюється цифровим барометром BMP280 [10]. Даний модуль забезпечує точність до $\pm 1\text{ гПа}$ та містить апаратні фільтри низьких частот. Їх наявність критично важлива для усунення короточасних флуктуацій показників тиску, спричинених рухом повітря (протягами) у приміщенні. Мікросхема, що об'єднує АНТ20 та BMP280 зображена на рисунку 2.5.



Рисунок 2.5 Датчик температури, вологості повітря та тиску

Для вимірювання рівня освітленості обрано спеціалізований сенсор BH1750 (рисунок 2.6).

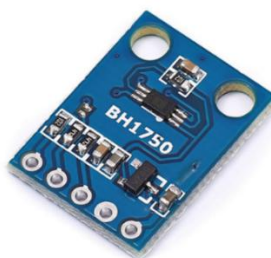


Рисунок 2.6 Датчик освітленості BH1750

На відміну від дешевих аналогових фоторезисторів, які мають нелінійну характеристику і вимагають складного програмного перерахунку спаду напруги в люкси, модуль ВН1750 оснащений власним 16-бітним АЦП. Мікросхема містить апаратний фільтр придушення оптичних пульсацій від штучних джерел освітлення (50/60 Гц) і напряму видає у шину I2C готове цілочисленне значення освітленості в діапазоні від 1 до 65535 лк [11]. Використання повністю цифрових датчиків усуває необхідність додаткової фільтрації аналогових шумів на мікроконтролері.

2.3 Засоби локальної візуалізації даних

Підвищення загальної відмовостійкості апаратно-програмного комплексу вимагає забезпечення безперервного доступу до телеметричних показників. Відмова локального маршрутизатора або тимчасова недоступність клієнтського застосунку не повинні призводити до втрати контролю над мікрокліматом. Для вирішення цієї задачі на апаратному рівні передбачено інтеграцію модуля безпосередньої візуалізації результатів.

Традиційним рішенням для бюджетних вбудованих систем є використання символьних рідкокристалічних індикаторів стандарту LCD 1602. Їх застосування має низку інженерних недоліків. РК-матриця вимагає постійного живлення модуля фоновому підсвічування, що збільшує струм споживання до 20–40 мА. Для інтеграції такого екрана по шині I2C необхідне використання мікросхеми-розширювача портів (PCF8574), яка часто функціонує на логічному рівні 5 В. Це створює потребу в додаткових каскадах узгодження логічних рівнів із 3.3-вольтовою шиною ESP32. Застосування альтернативних кольорових тонкоплівкових транзисторних матриць вимагає передачі даних по високошвидкісній шині Serial Peripheral Interface (SPI) із залученням мінімум 4-5 цифрових виводів мікроконтролера. Такий підхід є неможливим в умовах дефіциту вільних General-Purpose Input/Output (GPIO) після розведення шести аналогових зондів вологості.

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		20

На основі контрастного аналізу обрано монохромний графічний OLED-дисплей (рисунок 2.7) діагоналю 0.91 дюйма з роздільною здатністю 128x32 пікселі, побудований на контролері SSD1306 [12].

Технологія органічних світлодіодів не потребує зовнішнього підсвічування, оскільки кожен активний піксель є самостійним джерелом світла. Це знижує струм споживання до 10-15 мА під час відтворення тексту та забезпечує максимальну контрастність.

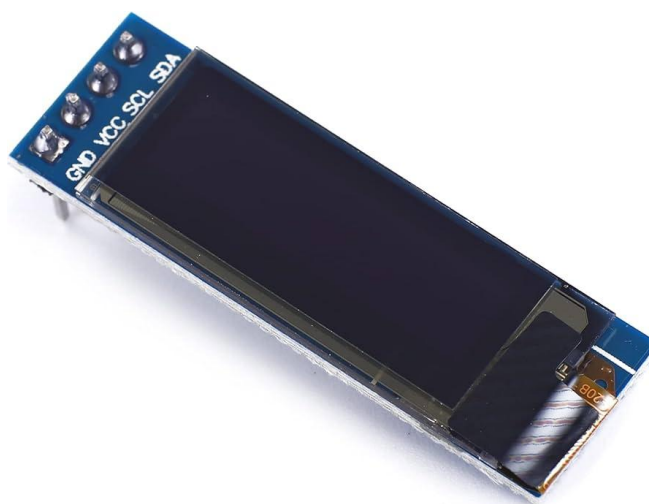


Рисунок 2.7 OLED-дисплей SSD1306

Модуль SSD1306 оснащений нативним апаратним інтерфейсом I2C і працює з логічними рівнями 3.3 В. Це дозволило підключити дисплей до існуючої цифрової магістралі паралельно з кліматичними сенсорами АНТ20, BMP280 та ВН1750, не використовуючи жодного додаткового GPIO-виводу мікроконтролера.

Впровадження графічного OLED-дисплея з керуванням через спільну шину I2C дозволяє реалізувати повноцінний інтерфейс локального керування апаратною частиною. Це гарантує повну автономність комплексу, дозволяючи отримувати актуальну телеметрію та проводити апаратне калібрування зондів без обов'язкового підключення до програмного застосунку на персональному комп'ютері.

2.4 Джерело живлення та підсистеми фільтрації завад

2.4.1 Вибір джерела живлення

Процес безперервної трансляції даних через протокол WebSocket із частотою оновлення 500 мілісекунд унеможливорює використання режимів глибокого енергозбереження (Deep Sleep) мікроконтролера. У такому режимі роботи радіомодуль підтримує постійне з'єднання з точкою доступу, що зумовлює базове споживання струму на рівні 80–120 мА. Під час активної передачі пакетів даних (TX) мікроконтролер генерує короточасні стрибки споживання амплітудою до 400–500 мА [13]. Використання автономних джерел живлення, таких як літій-іонні акумулятори формату 18650, є економічно недоцільним через необхідність їх частого заряджання.

Оптимальним рішенням для стаціонарної системи моніторингу обрано живлення від зовнішньої мережі через вбудований у відлагоджувальну плату ESP32-WROOM-32D порт USB (5 В). Такий підхід усуває необхідність розробки окремого зовнішнього блоку живлення. Для живлення ядра мікроконтролера та зовнішньої периферії, що працює на логічному рівні 3.3 В, використовується вже інтегрований на платі розробника лінійний стабілізатор напруги з низьким падінням. На відміну від зовнішніх імпульсних DC-DC перетворювачів, які генерують високочастотні комутаційні завади, вбудований лінійний стабілізатор забезпечує чисту вихідну напругу. Це є критично важливою умовою для отримання еталонної опорної напруги та коректної роботи 12-бітного аналого-цифрового перетворювача під час опитування аналогових зондів ґрунту.

2.4.2 Підсистема апаратної фільтрації завад та захисту шин даних

Функціонування високочутливого багатоканального АЦП та цифрових шин у безпосередній близькості до активного радіотрансивера на частоті 2.4 ГГц створює проблему електромагнітної сумісності. Стрибки струму під час передачі даних через Wi-Fi викликають короточасні просадки напруги на загальній шині 3.3 В, а радіочастотне випромінювання наводить паразитні високочастотні

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		22

завади на довгих вимірювальних лініях аналогових зондів. Це призводить до флуктуації (шуму) показників вологості ґрунту та мікрозависань I2C-периферії.

Для стабілізації живлення та нівелювання просадок напруги розроблено дворівневу підсистему фільтрації магистралі 3.3 В. Перший електролітичний конденсатор ємністю 100 мкФ встановлено безпосередньо біля виводів живлення мікроконтролера ESP32 для компенсації пікових струмів радіомодуля. Другий конденсатор аналогічної ємності (100 мкФ) розміщено в зоні підключення цифрової I2C-периферії. Такий розподіл локальних буферів енергії гарантує стабільність опорної напруги як для обчислювального ядра, так і для віддалених кліматичних сенсорів та дисплея.

Особливу увагу приділено захисту аналогових ліній зв'язку. Оскільки вимірювальні зонди ґрунту підключаються до каналів АЦП1 (які здатні працювати паралельно з активним Wi-Fi модулем), на кожному з шести аналогових входів мікроконтролера реалізовано апаратний фільтр. Він складається з послідовно включеного резистора номіналом 100 Ом та керамічного шунтуючого конденсатора ємністю 0.1 мкФ, підключеного до землі. Дане схемотехнічне рішення ефективно зрізає наведені високочастотні радіозавади та згладжує аналоговий сигнал до його потрапляння на вхід АЦП, що значно підвищує точність 12-бітного квантування.

2.5 Технології мережевого обміну

Забезпечення миттєвої реакції графічного інтерфейсу на зміни параметрів мікроклімату вимагає оптимізації транспортного рівня мережі. У системах багатоканального моніторингу в реальному часі критичним показником є не стільки загальна пропускну здатність каналу, скільки мінімізація накладних витрат на ініціалізацію сеансів зв'язку та утримання мережевої затримки на мінімальному рівні.

Традиційним підходом для побудови IoT-систем є використання архітектури REST API поверх протоколу HTTP. Аналіз цього рішення виявляє

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		23

його неефективність для процесів безперервної трансляції даних. Реалізація класичних HTTP-запитів вимагає виконання процедури потрібного рукописання для кожної транзакції. Формування об'ємних текстових HTTP-заголовків мікроконтролером споживає значний обсяг динамічної пам'яті та призводить до її швидкої фрагментації. Крім того, синхронна обробка запитів клієнта блокує процесорний час, що унеможлиблює паралельне зчитування показників з АЦП без внесення часових похибок.

Альтернативним варіантом є використання протоколу MQTT, спеціально розробленого для телеметрії. Проте його архітектура вимагає обов'язкової наявності проміжного серверного вузла - MQTT-брокера. Залучення віддалених хмарних брокерів створює критичну залежність системи від стабільності підключення до глобальної мережі та збільшує час доставки пакетів до 150–300 мс. Розгортання ж локального брокера вимагає введення в систему додаткового мікрокомп'ютера, що невиправдано ускладнює апаратну базу та підвищує загальну вартість комплексу.

Для усунення зазначених недоліків технологію мережевого обміну обрано на базі повнодуплексного протоколу WebSocket. Ключовою інженерною перевагою такого вибору є зміна транзакційної моделі на єдине постійне з'єднання. Це усуває накладні часові та ресурсні витрати на багаторазове ініціювання сеансів зв'язку, що є головним фактором затримок у класичному HTTP. Крім того, кардинальне зменшення обсягу службової інформації у пакетах даних розвантажує оперативну пам'ять мікроконтролера та запобігає її швидкій фрагментації під час тривалого безперервного моніторингу.

Використання асинхронної природи WebSocket є критично важливим для збереження точності апаратних вимірювань. Трансляція об'ємних масивів телеметрії з високою частотою (кожні 500 мілісекунд) не викликає зупинки основного циклу опитування периферії. Це гарантує відсутність пропусків під час зчитування слабких аналогових сигналів із зондів ґрунту.

Доцільність такого підходу підтверджується інструментальними

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		24

метриками: перехід на повнодуплексний обмін повністю усуває явище блокування цифрової шини I2C під час передачі даних, знижує завантаження процесорного ядра обробки мережі до 5–8%, а затримка доставки пакетів у локальній Wi-Fi мережі стабілізується на мінімальному рівні 5–12 мс.

2.6 Середовища розробки

Проектування апаратно-програмного комплексу, що інтегрує низькорівневу мікроконтролерну логіку та високорівневу клієнтську частину, вимагає застосування уніфікованого інструментарію. Для мінімізації когнітивного навантаження розробника та забезпечення швидкості ітерацій, як основне середовище розробки для обох частин проєкту обрано Visual Studio Code. Це середовище забезпечує безшовний перехід між контекстами розробки та надає потужний функціонал для інтеграції різних екосистем в одному робочому просторі.

2.6.1 Середовище розробки для мікроконтролера ESP32

Для програмування мікроконтролера ESP32 у середовищі VS Code використовується розширення PlatformIO (рисунок 2.8).

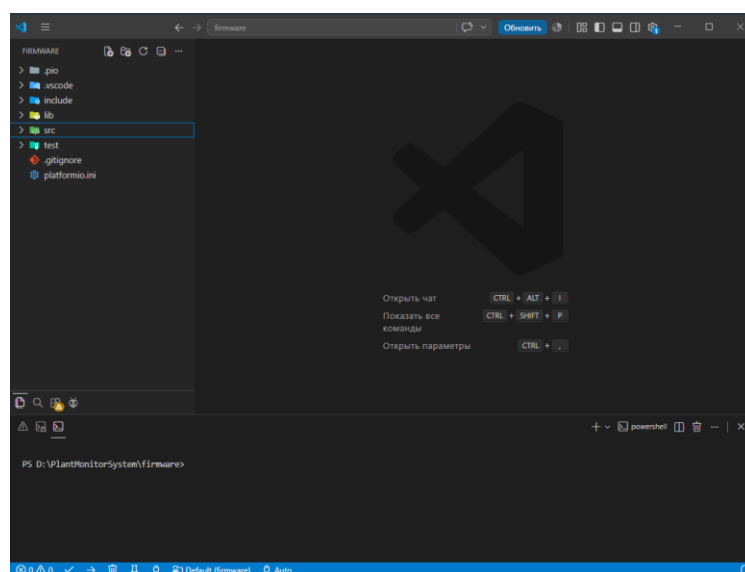


Рисунок 2.8 Інтерфейс середовища PlatformIO у Visual Studio Code

Вибір цього інструменту замість класичного Arduino Integrated Development Environment (IDE) зумовлений його професійною орієнтацією на вбудовані системи та забезпеченням відтворюваності результатів компіляції. Замість глобальних налаштувань IDE, проєкт використовує файл конфігурації platformio.ini. Це забезпечує сувору фіксацію версій усіх бібліотек, що гарантує ідентичність компіляції прошивки на будь-якому робочому місці та унеможливорює виникнення конфліктів при оновленні сторонніх залежностей.

Платформа надає повноцінний інтелектуальний аналіз коду IntelliSense для C/C++, що значно пришвидшує навігацію структурою бібліотек та дозволяє виявляти помилки ще на етапі написання коду. Окрім цього, інтегровані інструменти налагодження дозволяють легко підключити монітор послідовного порту для відстеження логів роботи ESP32 в режимі реального часу, що є критично важливим для налагодження асинхронного мережевого стеку та виявлення прихованих затримок у виконанні коду.

2.6.2 Середовище розробки кросплатформного клієнтського застосунку

Для розробки кросплатформного клієнтського застосунку також обрано середовище Visual Studio Code, що забезпечує повну уніфікацію інструментарію розробки в межах одного проєкту. Оскільки в якості основного інструменту для програмування мікроконтролера вже використовується Visual Studio Code, застосування цього ж середовища для клієнтської частини дозволяє уникнути контекстного перемикання між різними інтерфейсами, що суттєво зменшує витрати часу та знижує імовірність виникнення помилок при інтеграції програмних модулів.

Використання єдиного середовища для розробки як низькорівневої прошивки, так і високорівневого графічного інтерфейсу, дозволяє ефективно керувати всім репозиторієм проєкту з одного робочого вікна. Такий підхід забезпечує цілісність конфігурації робочого простору, де системні змінні, шляхи

до бібліотек та налаштування налагодження можуть бути централізовано налаштовані під потреби конкретного етапу розробки. Розробник має змогу одночасно редагувати код логіки обробки даних на C++ та код клієнтської частини, що забезпечує швидкий доступ до спільних структур даних та узгодженість протоколів передачі інформації.

Середовище Visual Studio Code надає широкі можливості для налагодження та профілювання, що є критично важливими при роботі з потоковими даними. Завдяки підтримці інтегрованих інструментів профілювання, розробник може в реальному часі відстежувати споживання оперативної пам'яті, завантаження центрального процесора та коректність роботи мережевих сокетів безпосередньо в інтерфейсі редактора. Можливість перемикання між цільовими платформами збірки (Windows, Android) в одному середовищі дозволяє оперативно проводити тестування інтерфейсу на різних операційних системах, що значно пришвидшує ітераційний цикл розробки та підвищує стабільність кінцевого програмного продукту.

2.7 Бібліотеки та фреймворки

Успішна реалізація багатоканальної системи збору телеметрії та кросплатформного клієнтського застосунку значною мірою залежить від коректного вибору програмних інструментів. При відборі бібліотек пріоритет надавався рішенням із відкритим кодом, які мають високу стабільність роботи в обмежених ресурсах мікроконтролера та забезпечують низькі накладні витрати на виконання коду, гарантуючи при цьому високу швидкість обробки даних у реальному часі.

2.7.1 Бібліотеки для мікроконтролера ESP32

Програмна частина мікроконтролера базується на використанні спеціалізованих драйверів, які забезпечують ефективну взаємодію з периферією через шини I2C та GPIO. Однією з найважливіших бібліотек для реалізації мережевого стеку є ESPAsyncWebServer. Вибір цієї бібліотеки обґрунтований її

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		27

асинхронною архітектурою, яка є принципово відмінною від класичних синхронних рішень. Вона не блокує виконання основного циклу мікроконтролера під час очікування запиту від клієнта. Це дозволяє ESP32 продовжувати опитування аналогових зондів вологості навіть під час активного обміну даними через WebSocket, що є критично важливим для уникнення часових затримок, які в однопотокових системах призводять до спотворення вимірювань.

Для взаємодії з цифровими сенсорами клімату обрано набір бібліотек Adafruit_AHTX0, Adafruit_BMP280 та BH1750. Дані інструменти обрано через їхню високу оптимізацію під архітектуру ESP32 та мінімальні вимоги до оперативної пам'яті. Вони надають прямий та низькорівневий доступ до внутрішніх регістрів цифрових сенсорів, що дозволяє мінімізувати кількість ітерацій передачі даних по шині I2C. Це дозволяє отримувати вже калібровані значення фізичних величин без необхідності додаткової математичної обробки сирих даних безпосередньо на боці мікроконтролера, що значно розвантажує центральний процесор.

Для серіалізації даних перед відправкою через WebSocket обрано бібліотеку ArduinoJson. Її перевага полягає у здатності до динамічного розрахунку розміру пам'яті, необхідної для формування пакету даних перед його відправкою. Це дозволяє уникнути переповнення стеку під час роботи з великими структурами даних, що містять показники всіх шести каналів вологості одночасно. Використання цієї бібліотеки гарантує стабільність формату передачі, що дозволяє клієнтському застосунку безпомилково парсити телеметрію.

2.7.2 Інструментарій клієнтського застосунку та фреймворк Flutter

При розробці клієнтської частини пріоритетним вибором є фреймворк Flutter, який завдяки своїй архітектурній гнучкості забезпечує високу продуктивність візуалізації телеметричних даних. Головною перевагою даного фреймворку є використання власного графічного рушія Impeller, який взаємодіє

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		28

безпосередньо з графічним Application Programming Interface (API) операційної системи [14]. Це дозволяє уникнути проміжних шарів обробки, що є критично важливим для забезпечення плавності інтерфейсу під час відображення динамічних графіків та миттєвого оновлення стану системи.

Архітектура Flutter базується на використанні мови Dart, яка забезпечує потужні механізми асинхронного програмування за допомогою потоків (Streams) та Future. У контексті моніторингу реального часу це дає змогу організувати неблокуючу обробку вхідних даних від мікроконтролера через WebSocket. Використання асинхронних потоків у поєднанні з віджетами StreamBuilder забезпечує реактивну модель оновлення інтерфейсу: візуальні елементи перемальовуються лише в момент отримання нових даних. Такий підхід мінімізує навантаження на процесор та запобігає появі мікрозатримок, що часто виникають при використанні класичних циклів опитування інтерфейсу.

Важливою особливістю фреймворку є компонентна модель побудови інтерфейсу на основі ієрархії віджетів. Кожен логічний елемент дашборду - від цифрових показників температури до діаграм вологості - реалізується як окремий віджет, що відповідає за власну частину візуалізації. Декларативна парадигма програмування дозволяє автоматично підтримувати відповідність між поточним станом даних та візуальним представленням, виключаючи необхідність ручного керування кожним елементом при оновленні параметрів. Це суттєво підвищує надійність інтерфейсу та спрощує його масштабування при додаванні нових сенсорів.

Для забезпечення стабільності мережевої взаємодії застосовуються спеціалізовані бібліотеки, такі як web_socket_channel. Використання подібних абстракцій дозволяє надійно підтримувати постійне TCP-з'єднання з мікроконтролером ESP32, забезпечуючи автоматичне відновлення зв'язку у разі виникнення мережевих перешкод. Це гарантує безперервність відображення телеметрії, що є ключовою вимогою для систем безперервного контролю мікроклімату. Додатково, використання засобів локального зберігання

налаштувань, як-от Shared_preferences, дає змогу забезпечити миттєве відновлення користувацької конфігурації одразу після запуску застосунку, усуваючи потребу в повторному ручному налаштуванні системи.

Застосування такого інструментарію дозволяє побудувати надійний клієнтський застосунок, здатний до стабільної роботи в режимі моніторингу 24/7 без деградації продуктивності, забезпечуючи користувачеві актуальну інформацію з мінімальними затримками доставки даних.

3. ПРОЄКТУВАННЯ ПРИСТРОЮ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Апаратна реалізація та підключення компонентів

На етапі апаратної реалізації розробленої системи здійснюється перехід від концептуального вибору компонентів до їх фізичного об'єднання в єдиний комплекс. Головним завданням цього етапу є забезпечення коректної маршрутизації сигнальних ліній та створення надійної інфраструктури живлення. Макетування системи виконується на безпайковій макетній платі. Фізичне розташування елементів на макетній платі вимагає дотримання правил просторового розділення високочастотних цифрових шин та чутливих аналогових ліній для мінімізації перехресних завад.

3.1.1 Схема підключення мікроконтролера ESP32, цифрової шини I2C та аналогових зондів

Основою апаратної збірки є відлагоджувальна плата ESP32-WROOM-32D, яка розміщується в центральній частині макетної плати для забезпечення оптимальної довжини з'єднувальних провідників. Цифрова магістраль I2C формується шляхом паралельного підключення всіх сумісних периферійних модулів до двох виділених портів загального призначення. Датчики температури та вологості АНТ20, атмосферного тиску BMP280, освітленості BH1750, а також графічний OLED-дисплей SSD1306 фізично об'єднуються на єдиній шині даних SDA та шині тактування SCL.

Підключення шести ємнісних зондів вологості ґрунту вимагає особливої уваги до вибору портів мікроконтролера. Аналогові виходи сенсорів підключаються виключно до виводів, які належать до першого аналого-цифрового перетворювача (ADC1). Використання каналів другого перетворювача (ADC2) є апаратно неможливим, оскільки його ресурси жорстко резервуються внутрішнім драйвером Wi-Fi під час будь-якої мережевої активності. Сигнальні дроти від зондів ґрунту прокладаються таким чином, щоб

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		31

вони не перетиналися з лініями I2C та антеною радіомодуля. Загальний вигляд фізичного з'єднання всіх компонентів, розподіл сигнальних ліній та розташування периферії на макетній платі наведено на рисунку 3.1.

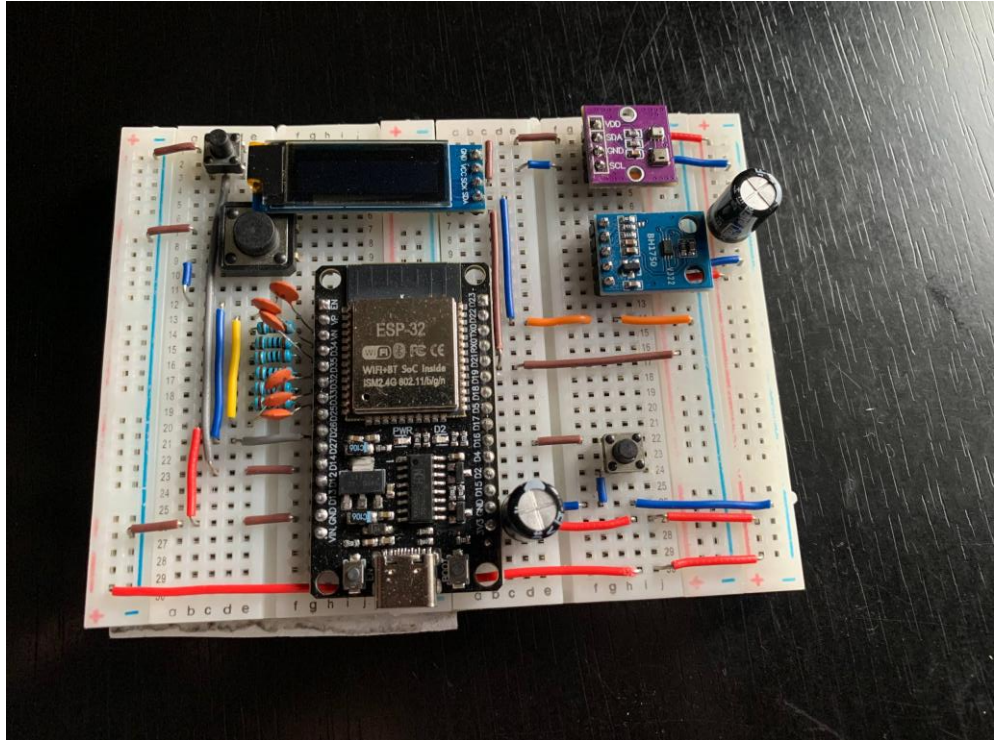


Рисунок 3.1 Підключення компонентів апаратно-програмного комплексу на макетній платі

3.1.2 Організація живлення та апаратний захист від імпульсних завад радіомодуля

Розподіл живлення на макетній платі організовано через дві поздовжні шини, на які подається напруга 3.3 В від вбудованого стабілізатора плати ESP32. Під час активної роботи Wi-Fi трансивера виникають короточасні просадки напруги, які поширюються по всій довжині макетної плати. Для фізичного придушення цих просадок на шині живлення встановлюються два електролітичні конденсатори ємністю 100 мкФ. Перший конденсатор монтується максимально близько до виводу 3.3 В мікроконтролера для миттєвої компенсації стрибків струму радіомодуля. Другий конденсатор розміщується на

протилежному кінці макетної плати, безпосередньо в зоні підключення кластера цифрових I2C-сенсорів та OLED-дисплея, створюючи локальний буфер енергії для периферії.

Апаратний захист аналогових ліній реалізується шляхом монтажу фільтрів низьких частот (RC-фільтрів) безпосередньо перед аналоговими входами мікроконтролера. У розрив кожного з шести сигнальних провідників, що йдуть від зондів вологості, встановлюється резистор номіналом 100 Ом. Одразу після резистора, між аналоговим входом ESP32 та шиною заземлення (GND), підключається керамічний конденсатор ємністю 0.1 мкФ. Фізичне розміщення цих компонентів впритул до мікроконтролера є критично важливим, оскільки довгі дроти від датчиків ґрунту працюють як антени, вловлюючи електромагнітне випромінювання від Wi-Fi антени. Зібрана на макетній платі конфігурація фільтрів ефективно замикає наведені високочастотні завади на землю, гарантуючи надходження чистого аналогового сигналу на канали ADC1 для подальшого точного оцифрування.

3.2 Розробка мікропрограмного забезпечення ESP32

Процес розробки програмного забезпечення для мікроконтролера ESP32 побудований навколо концепції неблокуючої багатозадачності та забезпечення повної автономності вимірювального пристрою. Оскільки апаратно-програмний комплекс повинен одночасно опитувати ряд сенсорів, оновлювати локальний графічний дисплей, обробляти апаратні переривання від фізичних кнопок та підтримувати постійне мережеве з'єднання через протокол WebSocket, використання стандартних функцій затримки потоку виконання є повністю неприпустимим. Замість цього архітектура мікропрограми базується на використанні системного таймера, що дозволяє кожному логічному модулю виконуватися у власні кванти часу без зупинки основного циклу процесора. Головний цикл програми постійно перевіряє різницю між поточним часом та мітками останнього виконання завдань, ініціюючи зчитування даних лише тоді,

коли проходить заданий інтервал. Це гарантує високу реактивність системи, відсутність пропусків під час зчитування критичних телеметричних даних та запобігає розривам мережевого з'єднання через тайм-аути. Важливою вимогою до системи є можливість її повноцінного використання, моніторингу та апаратного налаштування без обов'язкового підключення до персонального комп'ютера чи смартфона, що повністю реалізується через розгалужену систему меню на вбудованому дисплеї.

3.2.1 Організація багатозадачності та скінченний автомат локального інтерфейсу

Для керування локальним відображенням інформації та взаємодією з користувачем реалізовано структурний патерн проектування скінченного автомата. Вся логіка дисплея та навігації поділена на дев'ять чітко визначених станів, що описуються відповідним переліком.

```
enum DisplayState
{
    STATE_SLEEP,
    STATE_CLIMATE,
    STATE_PLANT_1,
    STATE_PLANT_2,
    STATE_PLANT_3,
    STATE_NETWORK_INFO,
    STATE_CALIB_MENU,
    STATE_CALIB_DRY,
    STATE_CALIB_WET
};
```

До цих станів належать режим енергозбереження екрана, базове відображення мікроклімату, попарне виведення даних із шести зондів ґрунту, інформація про локальну мережу та низка станів інтерактивного меню калібрування. Такий підхід дозволяє суворо розмежувати програмну логіку: обробник апаратних кнопок виконує виключно зміну поточного стану автомата та встановлює прапорець необхідності перемальовування, а підсистема

рендерингу відмальовує кадр відповідно до активного стану. Це суттєво спрощує масштабування інтерфейсу користувача, роблячи додавання нових інформаційних екранів безпечним для загальної стабільності мікропрограми.

Перехід між інформаційними станами здійснюється за допомогою системи обробки натискань фізичних кнопок, для яких реалізовано програмний алгоритм придушення брязкання контактів із затримкою у 50 мілісекунд.

Після успішної ініціалізації всіх шин даних та підключення до мережі, система переходить у базовий стан відображення мікроклімату. На цьому екрані в режимі реального часу виводяться усереднені дані з цифрових датчиків: температура, відносна вологість повітря та рівень освітленості.

Оскільки обмежена площа матриці дисплея не дозволяє читабельно розмістити показники вологості ґрунту з усіх шести каналів одночасно, реалізовано три окремі стани екрана для попарного виведення відсоткових значень зондів.

Додатково у скінченному автоматі дисплея передбачено стан інформаційного екрана мережі, на який виводиться поточний статус підключення до маршрутизатора та видана IP-адреса. Це повністю позбавляє користувача необхідності застосовувати стороннє програмне забезпечення для пошуку адреси пристрою в локальній мережі. Програмна реалізація демонстрації показників клімату наведено нижче.

```
switch (currentState){
    case STATE_CLIMATE:
        display.setCursor(0, 0);
        display.print("Temp");
        display.setCursor(0, 16);
        display.print(currentTemp, 1);
        display.print("C");
        display.setCursor(46, 0);
        display.print("Hum");
        display.setCursor(46, 16);
        display.print(currentHum, 1);
        display.print("%");
        display.setCursor(85, 0);
        display.print("Light");
```

```

display.setCursor(85, 16);
display.print(currentLight, 0);
display.print("Lx");
break;
...}

```

Граф переходів скінченного автомата інтерфейсу зображено на рисунку 3.2:

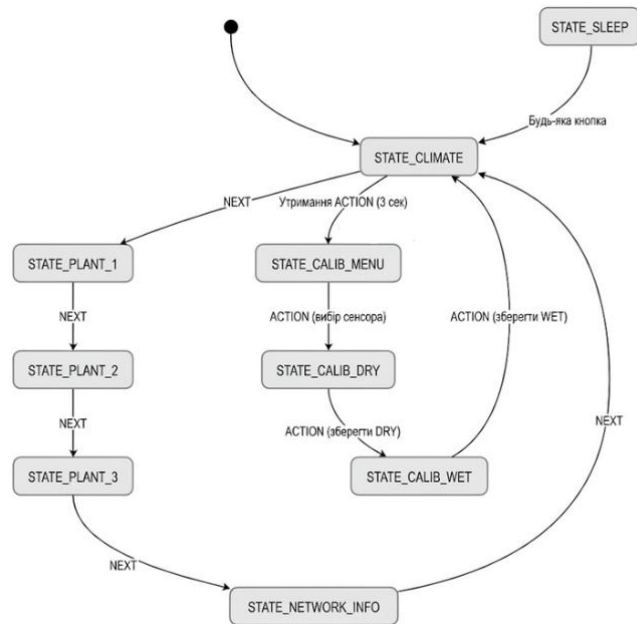


Рисунок 3.2 Граф переходів скінченного автомата локального інтерфейсу

3.2.2 Алгоритми цифрової фільтрації та стабілізації аналогових показань

Зчитування показників вологості ґрунту за допомогою вбудованого аналого-цифрового перетворювача значно ускладнюється наявністю високочастотних завад від роботи активного модуля Wi-Fi та електромагнітних наведень на довгих сигнальних лініях аналогових зондів. Для забезпечення еталонної точності вимірювань розроблено програмний механізм багаторівневої фільтрації. Перший рівень реалізовано шляхом взяття серії з п'ятнадцяти послідовних вимірювань для кожного з шести каналів із мікросекундними затримками між циклами перетворення. Другий рівень полягає у застосуванні

алгоритму медіанної фільтрації до зібраного пулу даних. Отриманий масив сирих значень сортується алгоритмом, після чого для подальшої математичної обробки обирається виключно центральне значення масиву. На відміну від звичайного середнього арифметичного, медіанний фільтр повністю ігнорує екстремальні одиничні сплески напруги та аномалії, що гарантує абсолютну стабільність показань вологості навіть за умов сильних радіоперешкод.

Для згладжування повільних флуктуацій показників від цифрових сенсорів мікроклімату застосовано метод ковзного середнього з кільцевим буфером. Це дозволяє ефективно нівелювати вплив локальних протягів або тимчасових змін інтенсивності освітленості на загальні графіки, роблячи візуалізацію даних у кросплатформному клієнтському застосунку більш плавною, передбачуваною та зручною для подальшого аналітичного опрацювання. Процес стабілізації сирих значень можна наочно відстежувати під час налагодження апаратної частини через монітор послідовного порту середовища розробки. Логіку фільтрації у вивигляді коду наведено нижче.

```
void sortArray(uint32_t a[], int size)
{
    for (int i = 0; i < size - 1; i++)
    {
        for (int j = i + 1; j < size; j++)
        {
            if (a[i] > a[j])
            {
                uint32_t temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
}

uint32_t getMedianAnalogRead(int pin)
{
    uint32_t samples[SAMPLE_COUNT];
    for (int i = 0; i < SAMPLE_COUNT; i++)
    {
        int rawValue = analogRead(pin);
        if (rawValue < 0 || rawValue > 4095)
        {
```

```

        rawValue = 4095;
    }
    samples[i] = rawValue;
    delayMicroseconds(500);
}
sortArray(samples, SAMPLE_COUNT);
return samples[SAMPLE_COUNT / 2];
}

```

3.2.3 Асинхронний мережевий стек та динамічна серіалізація телеметрії

Транспортний рівень мережевої взаємодії комплексу повністю базується на використанні бібліотеки асинхронного веб-сервера, яка обробляє відкриті TCP-з'єднання на рівні системних переривань. Головною архітектурною перевагою цього підходу є відсутність необхідності зупиняти основний цикл для очікування підтвердження отримання мережевого пакету від клієнта. Програмний алгоритм кожні 500 мілісекунд ініціює широкомовну розсилку поточних телеметричних показників усім активним клієнтам, підключеним через веб-сокети. Завдяки цій технології досягається максимальний ефект роботи в реальному часі в клієнтському застосунку без надмірного перевантаження процесора мікроконтролера циклічними запитами на оновлення стану.

Форматування масивів телеметричних даних перед відправкою здійснюється шляхом їх серіалізації у формат JSON за допомогою спеціалізованої бібліотеки. Це забезпечує кросплатформеність та універсальність протоколу обміну: клієнтський застосунок отримує чітко структурований текстовий документ, який швидко піддається синтаксичному розбору без ризику пошкодження пам'яті. Механізм серіалізації динамічно виділяє обсяг пам'яті для формування пакету, що містить поточні значення кліматичних сенсорів та згрупований масив відсоткових показників шести зондів вологості. Пакування даних ґрунту в єдиний внутрішній масив JSON-об'єкта оптимізує загальний обсяг вихідного мережевого трафіку, виключаючи необхідність передавати надлишкові текстові ключі для кожного окремого

фізичного каналу вимірювання.

```
void handleNetwork(){
    ws.cleanupClients();
    unsigned long currentMillis = millis();
    if (ws.count() > 0 && (currentMillis - lastWsBroadcastTime
    >= WS_BROADCAST_INTERVAL_MS)){
        lastWsBroadcastTime = currentMillis;
        JsonDocument doc;
        doc["temperature"] = currentTemp;
        doc["humidity"] = currentHum;
        doc["pressure"] = currentPress;
        doc["light"] = currentLight;
        JsonArray soilArray = doc["soil"].to<JsonArray>();
        for (int i = 0; i < TOTAL_SOIL_CHANNELS; i++){
            soilArray.add(soilPercent[i]);
        }
        String response;
        serializeJson(doc, response);
        ws.textAll(response);
    }
```

3.2.4 Інтерактивна система локального калібрування та енергонезалежна пам'ять

Специфічною фізичною особливістю ємнісних датчиків вологості є поступова деградація їх еталонних показників протягом тривалого часу експлуатації через старіння ізоляційного матеріалу текстоліту та зміну щільності навколишнього ґрунту. Для вирішення цієї експлуатаційної проблеми без необхідності підключення пристрою до комп'ютера та його перепрошивки, розроблено інтерактивну систему ручного апаратного калібрування.

Вхід до цієї системи надійно захищено програмним таймером: перехід у режим налаштувань відбувається виключно після безперервного утримання кнопки дії протягом трьох секунд, що унеможливорює випадкове порушення калібрування під час звичайної експлуатації. У цьому режимі користувач має змогу послідовно обрати необхідний сенсор та зафіксувати граничні значення мікроконтролера спочатку для сухого стану на відкритому повітрі, а потім для максимально вологого стану при повному зануренні у воду.

Збереження отриманих калібрувальних коефіцієнтів реалізовано з використанням механізму запису даних у вбудовану енергонезалежну флеш-пам'ять мікроконтролера.

```
void saveCalibration(int sensorIndex)
{
    preferences.begin("calibration", false);
    char dryKey[10];
    char wetKey[10];
    sprintf(dryKey, "dry%d", sensorIndex);
    sprintf(wetKey, "wet%d", sensorIndex);

    preferences.putUInt(dryKey, calDry[sensorIndex]);
    preferences.putUInt(wetKey, calWet[sensorIndex]);
    preferences.end();
}
```

Цей підхід гарантує абсолютне збереження життєво важливих налаштувань кожного сенсора навіть після тривалого повного знеструмлення пристрою. Алгоритм автоматично зчитує ці парні значення під час первинного завантаження операційної системи мікроконтролера і постійно використовує їх для математичного масштабування сирих значень АЦП у лінійні відсотки вологості від нуля до ста. Якщо калібрувальні значення для певного каналу збігаються або відсутні, мікропрограма розпізнає такий датчик як неналаштований і безпечно ігнорує його показання, передаючи нульове значення, що запобігає виникненню критичних помилок під час виконання програми.

3.3 Розробка клієнтського десктопного застосунку

3.3.1 Архітектурна структура застосунку на базі фреймворку *Flutter*

Основою для створення клієнтської частини апаратно-програмного комплексу було обрано кросплатформний фреймворк Flutter від компанії Google, що використовує об'єктно-орієнтовану мову програмування Dart. Вибір цієї технології зумовлений її здатністю компілювати єдину кодову базу в нативні

машинні інструкції для операційної системи Windows, забезпечуючи високу продуктивність десктопного застосунку, порівнянну з програмами, написаними мовами C++ або C#. Крім того, архітектура Flutter базується на власному високопродуктивному русії рендерингу (Skia/Impeller), що дозволяє повністю контролювати процес відмальовування кожного пікселя на екрані незалежно від системних бібліотек інтерфейсу користувача.

Програмна архітектура розробленого десктопного застосунку побудована за принципом чіткого розподілу відповідальності. Весь вихідний код розділено на три логічні рівні: рівень представлення (UI), рівень бізнес-логіки та роботи з мережею та рівень моделювання даних. Рівень представлення відповідає виключно за візуалізацію даних та обробку дій користувача, делегуючи всі обчислення рівню сервісів. Такий підхід суттєво полегшує подальше масштабування застосунку, оскільки зміна структури мережових пакетів не вимагає переписувати графічний інтерфейс.

Центральним елементом архітектури є клас ApiService, який виступає єдиною точкою входу (Single Source of Truth) для всього застосунку. Цей сервіс інкапсулює в собі логіку встановлення WebSocket-з'єднання з мікроконтролером, управління таймерами відмовостійкості, парсинг вхідних даних та ведення каскадних буферів історії. Ініціалізація ApiService відбувається асинхронно на найвищому рівні ієрархії під час запуску програми в методі main(), після чого екземпляр сервісу передається дочірнім віджетам через механізм впровадження залежностей. Це гарантує, що застосунок має лише один активний мережовий сокет, унеможливлюючи конфлікти доступу до ресурсів та витoki оперативної пам'яті.

Оскільки Flutter використовує реактивну парадигму програмування, графічний інтерфейс реагує на зміни даних автоматично. Замість традиційного (імперативного) підходу, де розробник повинен явно викликати функції оновлення екрана, у ApiService створено ряд широкомовних потоків, на які підписуються віджети. Коли мікроконтролер надсилає новий пакет телеметрії,

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		41

сервіс додає його у відповідний потік, і фреймворк самостійно ініціює перемальовування лише тих частин екрана, де дані фактично змінилися.

Для представлення телеметричних даних у застосунку створено спеціалізовану модель `SensorData`. Вона містить визначення всіх необхідних полів для збереження показників клімату (температура, вологість, тиск, освітленість) та масиву значень шести ємнісних зондів вологості ґрунту. Особливістю цього класу є наявність фабричного конструктора `fromJson`, який виконує безпечну десеріалізацію вхідного рядка формату JSON, перетворюючи його на строго типізований об'єкт мови Dart. Конструктор містить вбудовані механізми перевірки наявності ключів (`json['key'] ?? 0.0`), що захищає програму від критичних збоїв у разі отримання пошкодженого пакету від мікроконтролера.

```
class SensorData {
  final double temperature;
  final double humidity;
  final double light;
  final List<int> soil;

  SensorData({
    required this.temperature,
    required this.humidity,
    required this.light,
    required this.soil,
  });

  factory SensorData.fromJson(Map<String, dynamic> json) {
    return SensorData(
      temperature: (json['temperature'] ?? 0.0).toDouble(),
      humidity: (json['humidity'] ?? 0.0).toDouble(),
      light: (json['light'] ?? 0.0).toDouble(),
      soil: json['soil'] != null
        ? List<int>.from(json['soil'])
        : List.filled(6, 0),);
  }

  factory SensorData.empty() {
    return SensorData(
      temperature: 0.0,
      humidity: 0.0,
      light: 0.0,
      soil: List.filled(6, 0),);
  }
}
```

					ІАЛЦ.467200.004 ПЗ	Арк.
						42
Змін.	Арк.	№ докум.	Підпис	Дата		

3.3.2 Реалізація служби безперервного бездротового зв'язку

Для забезпечення двостороннього обміну телеметричними даними з мінімальною мережевою затримкою між десктопним клієнтським застосунком та мікроконтролером ESP32 реалізовано з'єднання за протоколом WebSocket. Використання цього протоколу дозволяє уникнути накладних витрат на встановлення нових HTTP-з'єднань при високій частоті оновлення даних. Реалізація клієнтської частини сокета виконана за допомогою пакета `web_socket_channel`.

Ініціалізація з'єднання відбувається у методі `connectWebSocket`. Цей функціональний блок створює канал зв'язку за збереженою локальною IP-адресою пристрою та підписується на потік вхідних повідомлень. При отриманні кожного нового текстового пакета відбувається зміна статусу підключення на активний, скидання сторожового таймера та декодування даних з формату JSON. У разі виникнення поодинокі помилки під час передачі чи парсингу пакета (наприклад, через втрату частини фрейму в бездротовій мережі), застосунок перехоплює виняток за допомогою блоку `try-catch`, ігнорує пошкоджений кадр та продовжує прослуховування каналу без розриву з'єднання.

Специфікою бездротових IoT-систем є ймовірність раптової втрати зв'язку через фізичне знеструмлення апаратної частини або проблеми з маршрутизатором. Для апаратного контролю життєздатності з'єднання інтегровано механізм сторожового таймера. Метод `_resetWatchdog` перезапускає 4-секундний таймер при кожному отриманні валідного пакета. Оскільки мікроконтролер надсилає дані з більшою частотою, таймер за звичайних умов не досягає ліміту. Якщо передача даних припиняється більше ніж на 4 секунди, лічильник пропущених з'єднань збільшується, якщо лічильник дорівнює 4, тоді генерується подія розриву з'єднання.

```
void _startWatchdog() {  
    _watchdogTimer?.cancel();  
    _missedHeartbeats = 0;  
}
```

```

        _watchdogTimer = Timer.periodic(const Duration(seconds:
4), (timer) {
            _missedHeartbeats++;
            if (_missedHeartbeats >= 4) {
                _handleDisconnect();
            }
        });
}

```

Для забезпечення автономного функціонування застосунку без необхідності ручного втручання користувача розроблено алгоритм автоматичного відновлення зв'язку. При спрацюванні сторожового таймера, виникненні мережевої помилки або примусовому закритті сокета мікроконтролером викликається метод `_handleDisconnect`. Його призначення полягає в коректному звільненні зайнятих портів, оновленні реактивного стану інтерфейсу для інформування користувача про втрату зв'язку та запуску циклічного таймера. Цей таймер ініціює спробу встановлення нового з'єднання кожні 3 секунди. Після успішного відновлення каналу зв'язку цикл перепідключення переривається автоматично.

3.3.3 Алгоритми обробки телеметрії та багаторівнева система буферизації даних

З метою збереження накопичених історичних даних між сеансами роботи програми розроблено механізм запису буферів у постійну пам'ять за допомогою пакету `SharedPreferences`. Дані серіалізуються у формат JSON перед записом. Для уникнення блокування головного потоку інтерфейсу та зменшення зносу накопичувача застосовано техніку тротлінгу за допомогою таймера `_saveTimer`, який обмежує частоту фонових збережень до одного разу на 15 секунд. Додатково, клас сервісу підключено до системного спостерігача `WidgetsBindingObserver`, що дозволяє перехоплювати зміни життєвого циклу програми. Метод `didChangeAppLifecycleState` гарантує миттєвий дамп усіх масивів у сховище в момент, коли вікно застосунку згортається, призупиняється або закривається користувачем.

```

Future<void> _persistHistoryToStorage() async {
    final prefs = await SharedPreferences.getInstance();
    prefs.setString('h_1m', jsonEncode(buf1m.map((e) =>
e.toJson()).toList()));
    prefs.setString('h_5m', jsonEncode(buf5m.map((e) =>
e.toJson()).toList()));
    prefs.setString('h_30m', jsonEncode(buf30m.map((e) =>
e.toJson()).toList()),);
    prefs.setString('h_1h', jsonEncode(buf1h.map((e) =>
e.toJson()).toList()));
    prefs.setString('h_6h', jsonEncode(buf6h.map((e) =>
e.toJson()).toList()));
    prefs.setString('h_12h', jsonEncode(buf12h.map((e) =>
e.toJson()).toList()),);
    prefs.setString('h_24h', jsonEncode(buf24h.map((e) =>
e.toJson()).toList()),);}

```

Емпіричні дані з датчиків фізичних величин схильні до впливу апаратного шуму, електромагнітних наведень та короткочасних флуктуацій. Для стабілізації графіків у десктопному застосунку та усунення хибних спрацьовувань системи попереджень реалізовано програмний фільтр низьких частот на базі алгоритму ковзного середнього (Rolling Average). У тілі обробника WebSocket-повідомлень створено кільцевий буфер `_rollingBuffer` місткістю 10 елементів, що відповідає приблизно 5 секундам реального часу при частоті відправки пакетів 2 Гц. Кожне нове значення додається до буфера, витісняючи найстаріше. На основі цих даних розраховується середнє арифметичне значення для показників навколишнього середовища та всіх шести зондів вологості ґрунту. Згладжені дані інкапсулюються у новий екземпляр `SensorData` і передаються до графічного інтерфейсу.

Для оптимізації використання оперативної пам'яті клієнтського пристрою та запобігання переповненню графіків розроблено каскадну систему збереження історії. Пряме збереження всіх вхідних пакетів є неефективним, оскільки за добу генерується понад 170 тисяч точок даних. Замість цього система розподіляє телеметрію на сім незалежних масивів `HistoryPoint`. Кожен буфер відповідає за свій часовий масштаб відображення (від 1 хвилини до 24 годин) і має

індивідуальну частоту фіксації точок (від 1 секунди до 20 хвилин). Метод `_updateTieredHistory` здійснює перевірку часових міток і додає нові координати лише тоді, коли пройдено відповідний інтервал для конкретного рівня каскаду.

```
void _updateTieredHistory(SensorData data) {
    final now = DateTime.now();
    final hp = HistoryPoint(now, data);

    if (_last1m == null ||
        now.difference(_last1m!).inSeconds
            >= 1) {
        buf1m.add(hp);
        if (buf1m.length > 60) buf1m.removeAt(0);
        _last1m = now;
    }
    if (_last5m == null || now.difference(_last5m!).inSeconds
        >= 10) {
        buf5m.add(hp);
        if (buf5m.length > 30) buf5m.removeAt(0);
        _last5m = now;
    }
    if (_last30m == null ||
        now.difference(_last30m!).inMinutes >= 1) {
        buf30m.add(hp);
        if (buf30m.length > 30) buf30m.removeAt(0);
        _last30m = now;
    }
    if (_last1h == null || now.difference(_last1h!).inMinutes
        >= 2) {
        buf1h.add(hp);
        if (buf1h.length > 30) buf1h.removeAt(0);
        _last1h = now;
    }
    ...
    _throttledSave();
}
```

3.3.4 Розробка адаптивного графічного інтерфейсу користувача

Для забезпечення коректного відображення інформації на екранах різного розміру графічний інтерфейс побудовано за принципом адаптивної сітки. Головний екран моніторингу логічно розділено на два незалежні інформаційні блоки: «Навколишнє середовище» та «Вологість ґрунту». Для оптимізації рендерингу та забезпечення плавного прокручування використано віджет

CustomScrollView у поєднанні з елементами архітектури Sliver. Розташування карток датчиків реалізовано через SliverGrid із делегатом SliverGridDelegateWithMaxCrossAxisExtent. Цей метод дозволяє встановити максимальну ширину однієї картки (450 логічних пікселів), після чого рушій фреймворку самостійно обчислює оптимальну кількість стовпців. На широких дисплеях елементи розміщуються у чотири колонки, а при звуженні вікна динамічно перебудовуються, запобігаючи візуальному перекриттю. Висота карток (mainAxisExtent) змінюється адаптивно залежно від стану системного перемикача відображення аналітичних графіків.

Інтерфейс застосунку включає реактивну панель попереджень, яка інформує про критичні відхилення показників від заданої норми. Візуальний компонент побудовано на базі StreamBuilder, що асинхронно прослуховує широкомовний потік triggerStream з класу ApiService. За відсутності активних сповіщень фреймворк відмальовує порожній об'єкт SizedBox.shrink(), який не споживає апаратних ресурсів. При надходженні попередження дерево віджетів миттєво перемальовується, відображаючи контрастний контейнер із текстовим описом критичної події.

Управління параметрами системи винесено в окрему вкладку налаштувань. Графічний інтерфейс цієї вкладки дозволяє користувачеві змінювати IP-адресу мікроконтролера, керувати відображенням графіків, очищати локальну базу даних телеметрії та індивідуально конфігурувати кожен із шести апаратних портів датчиків ґрунту. Для запобігання збоям у роботі систем доступності операційної системи Windows (AXTree), складні інтерактивні елементи, такі як списки чекбоксів та перемикачі, ізольовано за допомогою системного віджета ExcludeSemantics. Конфігурація активних слотів формується динамічно циклом List.generate. Кожен елемент списку містить чекбокс для увімкнення/вимкнення датчика та кнопку для виклику діалогового вікна перейменування робочої зони. Зміни зберігаються асинхронно і миттєво застосовуються на головному екрані моніторингу.

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		47

3.3.5 Аналітична візуалізація даних та оптимізація рендерингу

Для побудови інформативних графіків зміни показників у часі було інтегровано бібліотеку `fl_chart`. Цей пакет дозволяє створювати високопродуктивні векторні графіки, які малюються безпосередньо за допомогою рушія Flutter, що забезпечує плавність навіть при відображенні великої кількості точок з каскадних буферів історії.

Однією з ключових вимог до аналітичного блоку була реалізація фіксованих, але адаптивних шкал вимірювання. Для більшості датчиків (температура, вологість, тиск) межі осі ординат (Y) є сталими та визначеними заздалегідь. Однак для датчика освітленості застосовано алгоритм динамічного масштабування. Оскільки рівень освітленості може коливатися від 0 (повна темрява) до 65000 люкс (пряме сонячне світло), фіксована шкала зробила б неможливим аналіз незначних коливань у темному приміщенні. Алгоритм попередньо сканує всі точки у вибраному часовому вікні, знаходить максимальне значення та динамічно підлаштовує верхню межу і крок сітки під один із заданих порогів: 100, 1000, 10000, 30000 або 65000.

```
double chartMaxY = maxY;

double chartInterval = yInterval;

if (isLight && spots.isNotEmpty) {
  double currentMax = 0;
  for (var spot in spots) {
    if (spot.y > currentMax) currentMax = spot.y;
  }
  if (currentMax <= 100) {
    chartMaxY = 100;
    chartInterval = 20;
  }
  else if (currentMax <= 500) {
    chartMaxY = 1000;
    chartInterval = 200;
  }
  else {
    chartMaxY = 65000;
    chartInterval = 10000;
  }
}
```

3.4 Тестування та результати роботи програмно-апаратного комплексу

3.4.1 Робота апаратної частини та інтеграція датчиків

Етап практичного тестування розпочався з перевірки функціонування апаратного вузла моніторингу, що включає мікроконтролер ESP32-WROOM-32D та вбудований OLED-дисплей для локальної візуалізації даних. Проведені дослідження підтвердили високу точність зчитування показників мікроклімату: датчики температури, вологості повітря та атмосферного тиску продемонстрували відповідність до контрольних приладів. На рис. 3.3 продемонстровано роботу апаратної частини з активованим OLED-дисплеєм, на якому відображаються актуальні параметри навколишнього середовища.

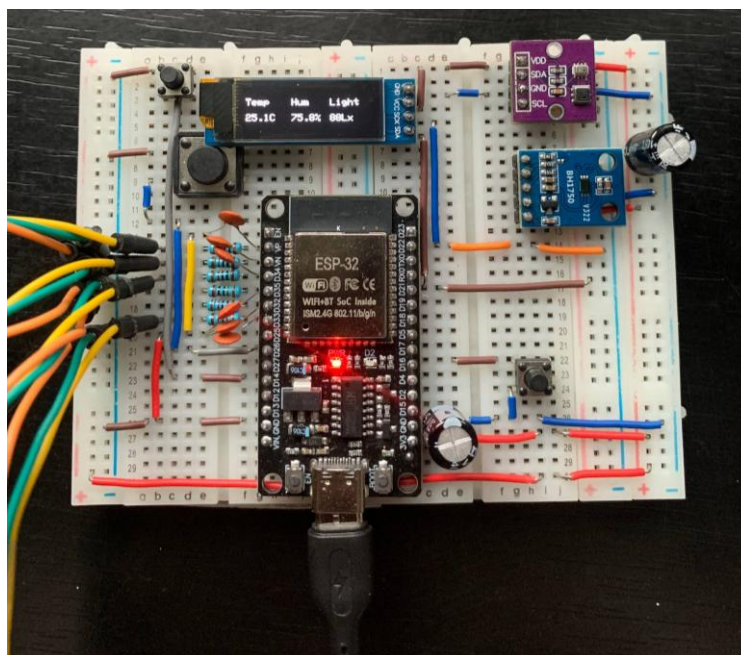


Рисунок 3.3 Моніторинг з демонстрацією кліматичних показників на OLED-дисплеї

Наступним кроком була перевірка функціональності ємнісних датчиків вологості ґрунту. Процес тестування включав калібрування кожного зонда у двох крайніх точках: у повітрі (ініціалізація 0% вологості) та у слянці з водою (ініціалізація 100% вологості). Це дозволило встановити лінійну залежність між

рівнем сигналу та фактичним вмістом вологи. Коректність вимірювань була підтверджена багаторазовими контрольними замірами в умовах експлуатації. На рис. 3.4 зафіксовано вихідні дані з OLED-дисплея під час моніторингу стану ґрунту у двох робочих зонах, що підтверджує стабільність збору даних та їх коректну обробку мікроконтролером.

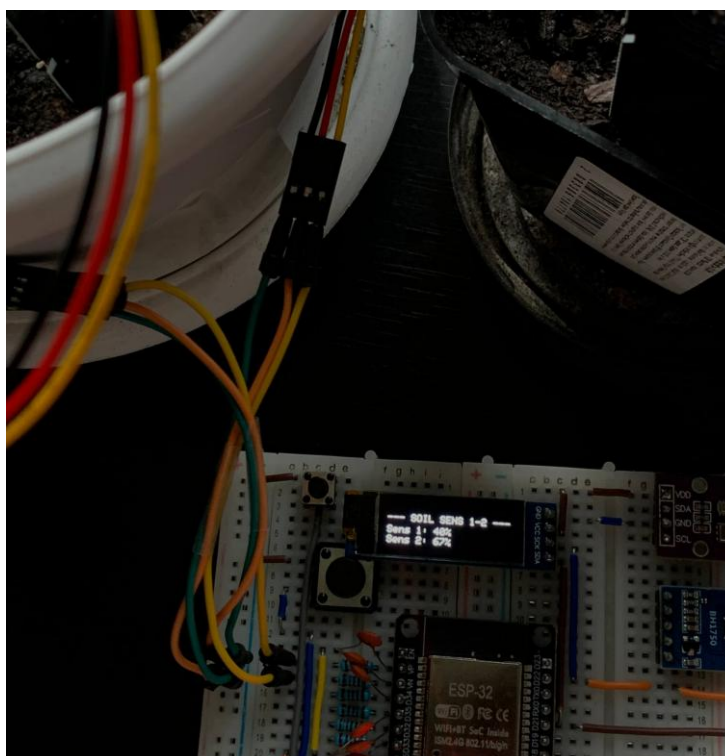


Рисунок 3.4 Результати вимірювань вологості ґрунту для двох активних зон моніторингу на OLED-дисплеї

3.4.2 Результати роботи десктопного застосунку та візуалізація

Після успішної перевірки апаратного рівня було проведено фінальне тестування програмного комплексу, що включало встановлення WebSocket-з'єднання з контролером ESP32 та накопичення історичних даних. Головний екран програми (рис. 3.5) продемонстрував коректну роботу інтерфейсу в реальному часі, де відображаються актуальні показники клімату та стан вологості ґрунту для кожної активної зони. Система автоматично опрацювала дані, отримані з шести датчиків, забезпечуючи інтуїтивне відображення

критичних параметрів.

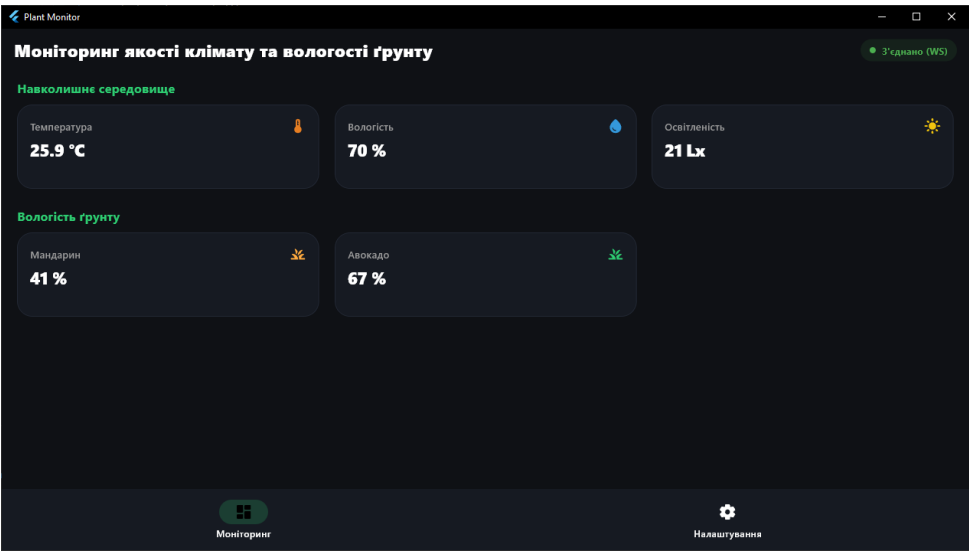


Рисунок 3.5 Головний екран десктопного застосунку з відображенням телеметрії в реальному часі

Для верифікації аналітичного модуля було здійснено перевірку каскадної системи буферизації. На рис. 3.6 представлено графіки динаміки зміни температури та вологості повітря за вибраний часовий інтервал. Впроваджений алгоритм ковзного середнього (rolling average) дозволив отримати плавні криві, які ефективно візуалізують тренди, мінімізуючи вплив випадкових похибок вимірювань. Графічна оболонка бібліотеки `fl_chart` забезпечила чітку інтерактивність: при наведенні курсора на будь-яку точку графіка користувач отримує доступ до точного значення показника у відповідний момент часу.

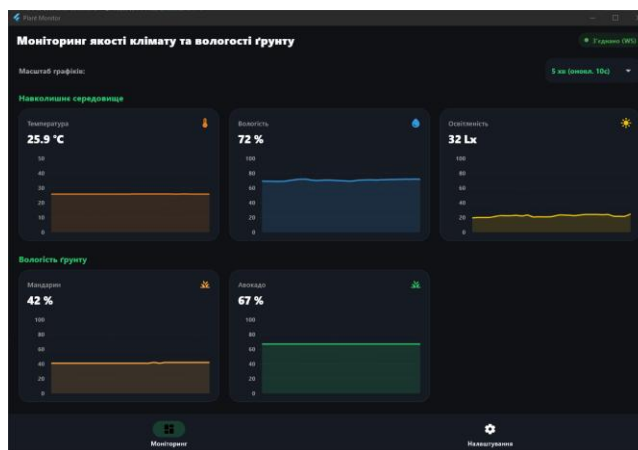


Рисунок 3.6 Аналітичні графіки динаміки зміни кліматичних показників

Важливим результатом тестування стала перевірка адаптивності системи до зміни інтенсивності освітлення. При переході з умов слабкого освітлення до яскравого світла алгоритм динамічного масштабування осі ординат автоматично перебудував межі графіка (рис. 3.7), що дозволило зберегти високу інформативність візуалізації незалежно від зовнішніх умов. Тести підтвердили, що програмна обробка даних та рендеринг графіків виконуються з мінімальною затримкою, що відповідає вимогам до швидкодії десктопних систем моніторингу.

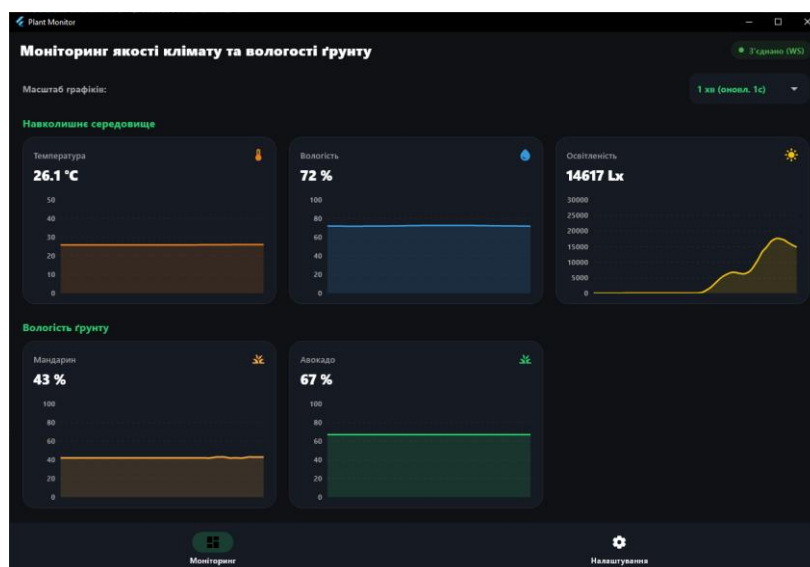


Рисунок 3.7 Адаптація графічної шкали освітленості при зміні вхідних даних

ВИСНОВКИ

У результаті виконання роботи розроблено та впроваджено апаратно-програмний комплекс для безперервного моніторингу параметрів мікроклімату та вологості ґрунту. Робота охоплює повний цикл проектування: від апаратної інтеграції до створення клієнтського десктопного застосунку.

На апаратному рівні реалізовано збір телеметрії на базі мікроконтролера ESP32-WROOM-32D. Інтегровано датчики температури, вологості повітря, атмосферного тиску, освітленості та шість ємнісних зондів вологості ґрунту. Забезпечено локальну візуалізацію даних через OLED-дисплей та передачу масивів інформації по бездротовій мережі Wi-Fi. Калібрування ємнісних датчиків підтвердило лінійну залежність показників і високу точність вимірювань у різних станах субстрату.

Програмну частину комплексу реалізовано у вигляді клієнтського десктопного застосунку з використанням кросплатформного фреймворку Flutter. Встановлено двосторонній асинхронний зв'язок за протоколом WebSocket. Розроблено алгоритм відмовостійкості на базі сторожового таймера, який ініціює автоматичне відновлення з'єднання у разі розриву мережі або знеструмлення апаратної частини.

Створено багаторівневу систему обробки телеметрії. Застосовано алгоритм ковзного середнього з кільцевим буфером на 10 ітерацій для апаратного згладжування та фільтрації шуму. Реалізовано каскадну систему збереження історії, яка розподіляє дані на сім незалежних часових масивів (від 1 хвилини до 24 годин). Налаштовано алгоритм фонового збереження накопичених даних у енергонезалежну пам'ять пристрою із прив'язкою до системних подій життєвого циклу програми.

Розроблено адаптивний графічний інтерфейс користувача з використанням архітектури Sliver. Інтерфейс виконує автоматичний перерахунок та масштабування елементів залежно від роздільної здатності

екрана. Впроваджено аналітичний модуль на базі векторних графіків з логікою динамічного масштабування осей для датчика освітленості. Налаштовано реактивну систему тригерів для миттєвого візуального сповіщення про критичні відхилення показників ґрунту.

Виконано глибоку інтеграцію застосунку з операційною системою комп'ютера. Програма підтримує фонову роботу з перехопленням подій віконного менеджера та взаємодію через контекстне меню системного трею, що оновлюється актуальною телеметрією в режимі реального часу. Ізоляція семантичного дерева під час рендерингу графіків дозволила повністю усунути ризики перевантаження систем доступності ОС.

Комплексне тестування підтвердило стабільність роботи системи під навантаженням, точність обробки телеметричних даних та надійність механізмів відновлення зв'язку. Розроблений комплекс вирішує поставлені завдання та повністю готовий до практичної експлуатації.

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		54

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Mi Flora Monitor. Xiaomi Global : URL: https://grokipedia.com/page/Xiaomi_Mi_Flora (дата звернення: 25.05.2026).
2. Bluetooth Core Specification Version 5.3. Bluetooth SIG : URL: <https://www.bluetooth.com/specifications/specs/> (дата звернення: 25.05.2026).
3. Meadow.Foundation.Sensors.Moisture.Fc28 : URL: <https://developer.wildernesslabs.co/docs/api/Meadow.Foundation/Meadow.Foundation.Sensors.Moisture.Fc28/> (дата звернення: 25.05.2026).
4. ESP8266EX Datasheet. Espressif Systems : URL: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf (дата звернення: 25.05.2026).
5. How Server Location Affects Latency. FDC Servers : URL: <https://fdcservers.net/uk/blog/how-server-location-affects-latency> (дата звернення: 25.05.2026).
6. ESP32 Series Datasheet. Espressif Systems : URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (дата звернення: 25.05.2026).
7. ATmega328P Datasheet. Microchip Technology : URL: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (дата звернення: 25.05.2026).
8. STM32F103 Datasheet. STMicroelectronics : URL: <https://www.st.com/resource/en/datasheet/stm32f103c8.pdf> (дата звернення: 25.05.2026).
9. AHT20 Product Datasheet. Aosong Electronics : URL: <https://www.aosong.com/userfiles/files/media/Data%20Sheet%20AHT20.pdf> (дата звернення: 25.05.2026).
10. BMP280 Digital Pressure Sensor Datasheet. Bosch Sensortec : URL:

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		55

<https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp280-ds001.pdf> (дата звернення: 25.05.2026).

11.BH1750FVI Datasheet. ROHM Semiconductor : URL: <https://www.mouser.com/datasheet/2/348/bh1750fvi-e-186247.pdf> (дата звернення: 25.05.2026).

12.SSD1306 Datasheet. Solomon Systech : URL: <https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf> (дата звернення: 25.05.2026).

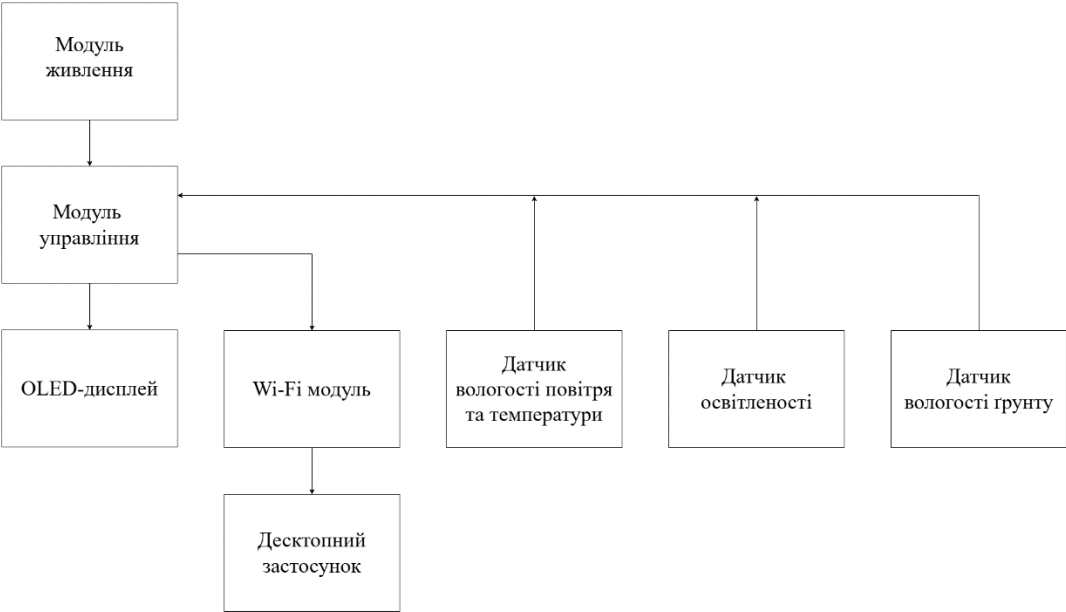
13.ESP32 Hardware Design Guidelines. Espressif Systems : URL: <https://docs.espressif.com/projects/esp-hardware-design-guidelines/en/latest/esp32/> (дата звернення: 25.05.2026).

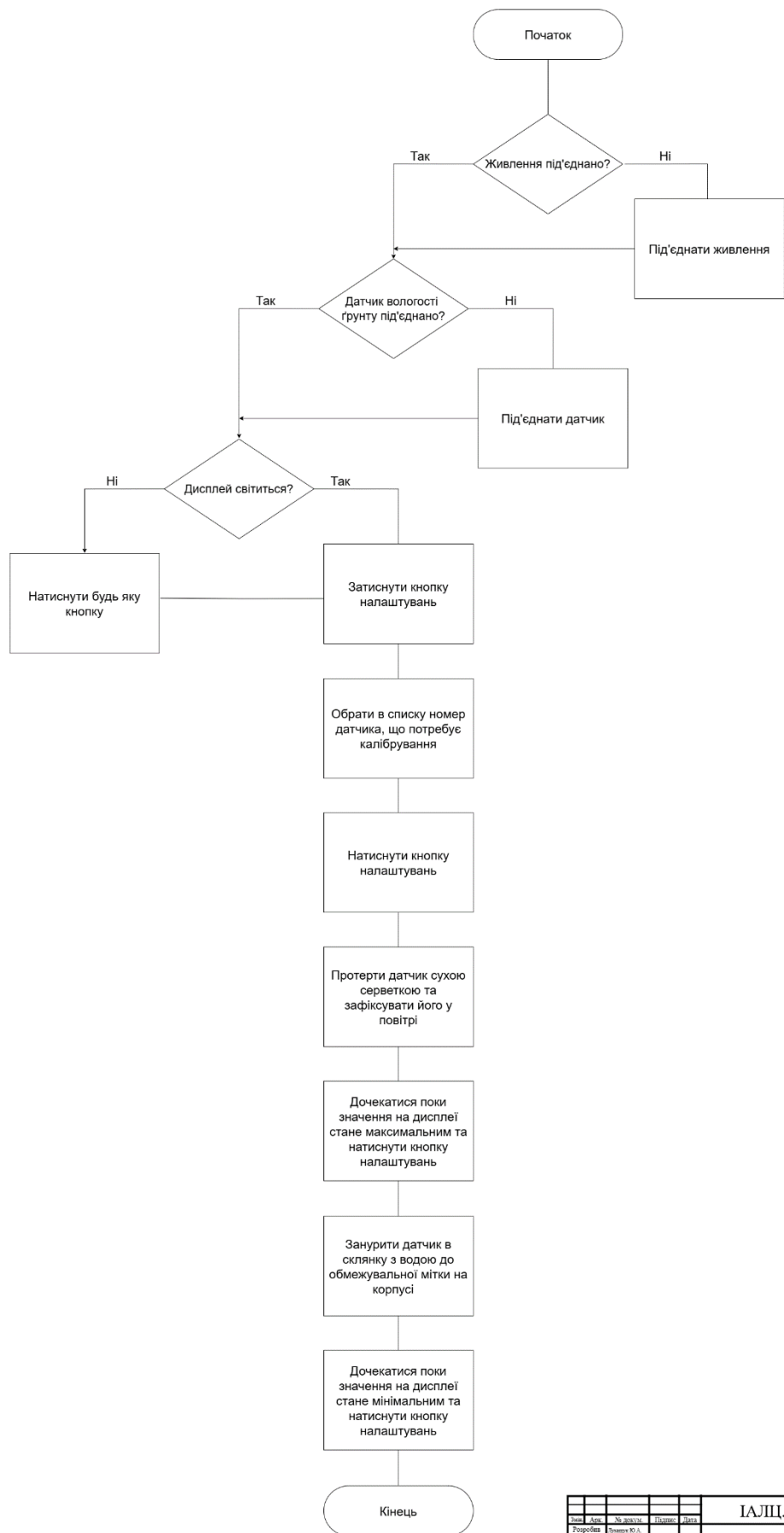
14.Flutter Documentation : URL: <https://docs.flutter.dev/> (дата звернення: 25.05.2026)

					ІАЛЦ.467200.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		56

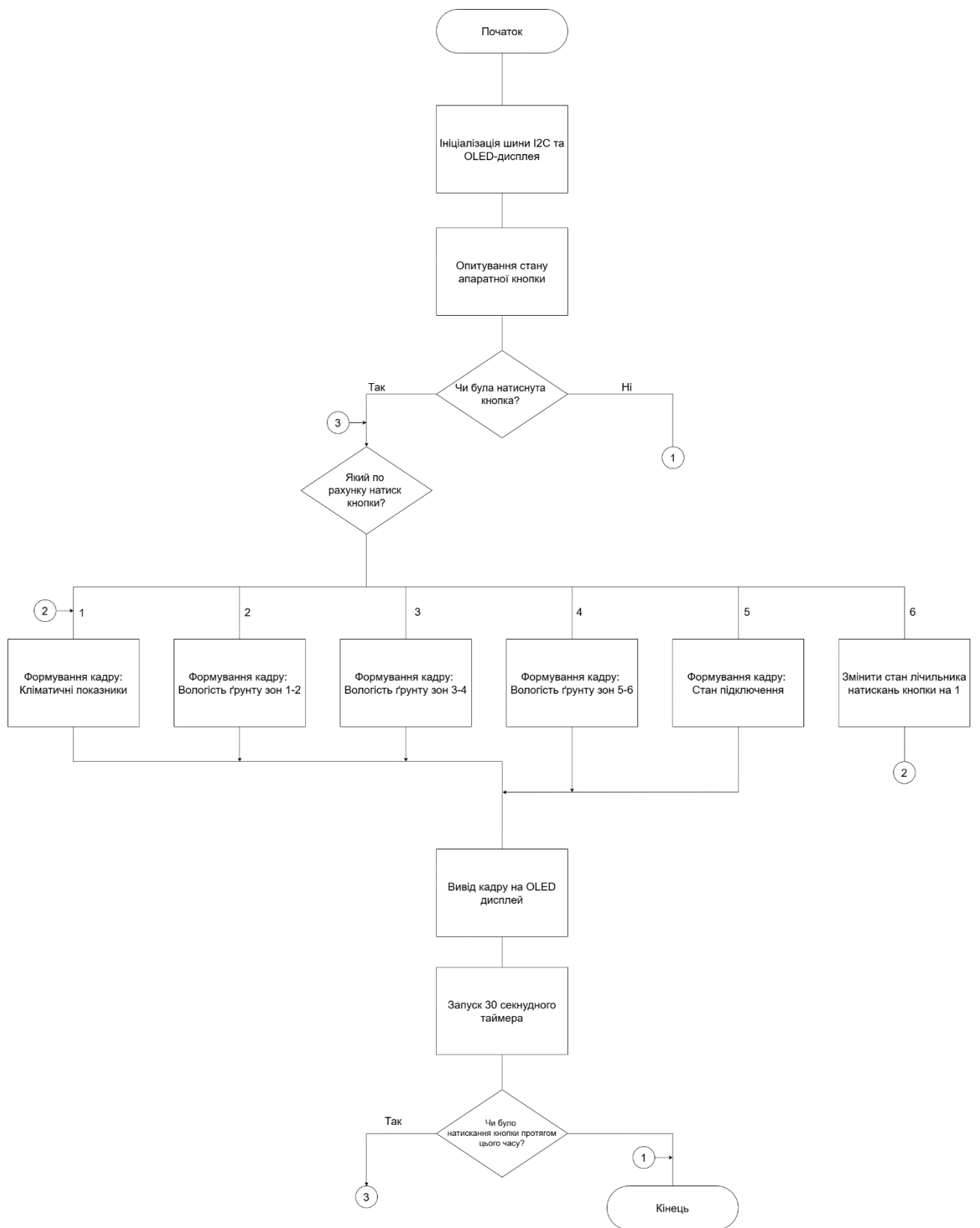
ДОДАТКИ

ДОДАТОК А
КОПІЇ ГРАФІЧНИХ МАТЕРІАЛІВ









ДОДАТОК Б
ФРАГМЕНТИ КОДУ

```

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

    handleNetwork();

    if (currentMillis - lastSensorReadTime >= SENSOR_READ_INTERVAL_MS)
    {
        lastSensorReadTime = currentMillis;
        readSensors();
        if (currentState != STATE_SLEEP)
        {
            needsRedraw = true;
        }
    }

    if (currentState != STATE_SLEEP && currentState < STATE_CALIB_MENU &&
        (currentMillis - lastActivityTime >= SLEEP_TIMEOUT_MS))
    {
        currentState = STATE_SLEEP;
        needsRedraw = true;
    }

    for (int i = 0; i < 3; i++)
    {
        bool reading = digitalRead(buttons[i].pin);

        if (reading != buttons[i].lastReading)
        {
            buttons[i].lastDebounceTime = currentMillis;
        }

        if ((currentMillis - buttons[i].lastDebounceTime) > DEBOUNCE_DELAY_MS)
        {
            if (reading != buttons[i].currentState)
            {
                buttons[i].currentState = reading;

                if (buttons[i].currentState == LOW)
                {
                    buttons[i].pressStartTime = currentMillis;
                    buttons[i].isLongPressHandled = false;
                }
            }
        }
    }
}

```

```

bool justWokeUp = resetActivityTimer();

if (!justWokeUp)
{
    if (currentState == STATE_CALIB_MENU)
    {
        if (buttons[i].pin == PIN_BTN_NEXT)
        {
            calibMenuIndex++;
            if (calibMenuIndex > TOTAL_SOIL_CHANNELS + 1)
                calibMenuIndex = 0;
            needsRedraw = true;
        }
        else if (buttons[i].pin == PIN_BTN_PREV)
        {
            calibMenuIndex--;
            if (calibMenuIndex < 0)
                calibMenuIndex = TOTAL_SOIL_CHANNELS + 1;
            needsRedraw = true;
        }
        else if (buttons[i].pin == PIN_BTN_ACTION)
        {
            if (calibMenuIndex < TOTAL_SOIL_CHANNELS)
            {
                selectedSensorForCalib = calibMenuIndex;
                currentState = STATE_CALIB_DRY;
            }
            else if (calibMenuIndex == TOTAL_SOIL_CHANNELS)
            {
                showWipingWifiMessage();
                resetWifiSettings();
                delay(1000);
                ESP.restart();
            }
            else
            {
                currentState = STATE_CLIMATE;
            }
            needsRedraw = true;
        }
    }
}
else if (currentState == STATE_CALIB_DRY)
{

```

```

    if (buttons[i].pin == PIN_BTN_ACTION)
    {
        calDry[selectedSensorForCalib] = soilRaw[selectedSensorForCalib];
        currentState = STATE_CALIB_WET;
        needsRedraw = true;
    }
}
else if (currentState == STATE_CALIB_WET)
{
    if (buttons[i].pin == PIN_BTN_ACTION)
    {
        calWet[selectedSensorForCalib] = soilRaw[selectedSensorForCalib];
        saveCalibration(selectedSensorForCalib);
        currentState = STATE_CLIMATE;
        needsRedraw = true;
    }
}
else if (currentState != STATE_SLEEP)
{
    if (buttons[i].pin == PIN_BTN_NEXT)
    {
        if (currentState == STATE_NETWORK_INFO)
            currentState = STATE_CLIMATE;
        else
            currentState = static_cast<DisplayState>(currentState + 1);
        needsRedraw = true;
    }
    else if (buttons[i].pin == PIN_BTN_PREV)
    {
        if (currentState == STATE_CLIMATE)
            currentState = STATE_NETWORK_INFO;
        else
            currentState = static_cast<DisplayState>(currentState - 1);
        needsRedraw = true;
    }
}
}
else
{
    buttons[i].isLongPressHandled = true;
}
}
else if (buttons[i].currentState == HIGH)

```

```

    {
        if (buttons[i].pin == PIN_BTN_ACTION && !buttons[i].isLongPressHandled
&& currentState < STATE_CALIB_MENU)
        {
            bool justWokeUp = resetActivityTimer();
            if (!justWokeUp)
            {
                currentState = STATE_NETWORK_INFO;
                needsRedraw = true;
            }
        }
    }
}

if (buttons[i].pin == PIN_BTN_ACTION && buttons[i].currentState == LOW &&
!buttons[i].isLongPressHandled)
{
    if (currentMillis - buttons[i].pressStartTime >= LONG_PRESS_MS)
    {
        buttons[i].isLongPressHandled = true;
        if (currentState < STATE_CALIB_MENU)
        {
            currentState = STATE_CALIB_MENU;
            calibMenuIndex = 0;
            needsRedraw = true;
        }
    }
}

buttons[i].lastReading = reading;
}

if (needsRedraw)
{
    drawScreen();
    needsRedraw = false;
}
}

```

ДОДАТОК В

ПРЕЗЕНТАЦІЯ

Апаратно-програмний комплекс для моніторингу мікроклімату рослин на базі мікроконтролера

Виконала: студентка групи КВ-22, Лукашук Юлія Андріївна
Керівник: доцент кафедри СПІСКС, д.т.н. Романкевич Олексій Михайлович



АКТУАЛЬНІСТЬ ТА МЕТА

Мета

Створення точної, автономної та високошвидкісної системи для багатоканального контролю умов життєдіяльності рослин

Актуальність

Необхідність точного контролю параметрів середовища для професійного та домашнього рослинництва. Проблема корозії дешевих сенсорів та затримок у хмарних IoT-рішеннях.



ВИБІР АПАРАТНИХ ЗАСОБІВ



Мікроконтролер ESP32

Двоядерна архітектура
Вбудований модуль Wi-Fi
Наявність 12-бітних АЦП та I2C



Ємнісні сенсори HW-390

Відсутність електрохімічної корозії
Висока точність
Легкість налаштування



Датчик АНТ20

Вимірювання температури, вологості повітря
Підтрима I2C протоколу



Датчик BH1750

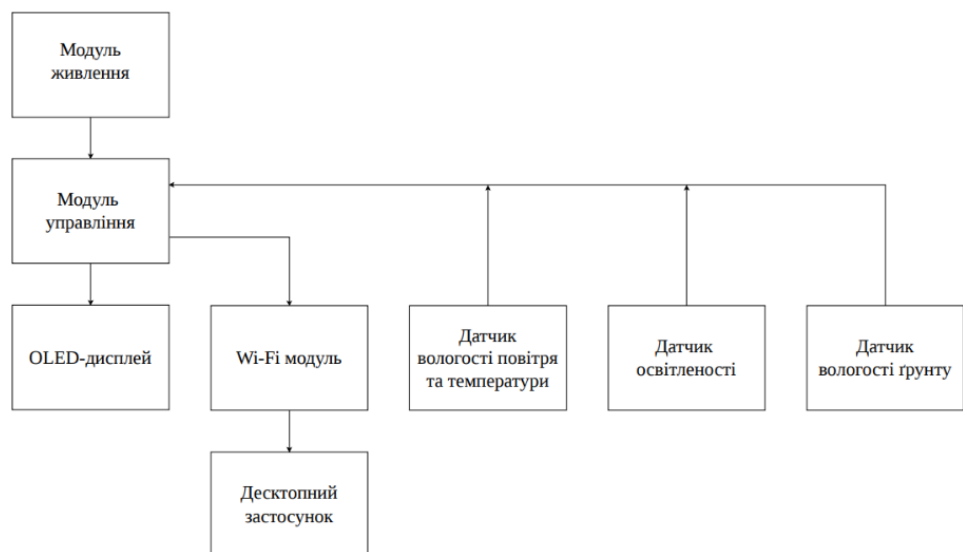
Вимірювання освітленості
Підтрима I2C протоколу

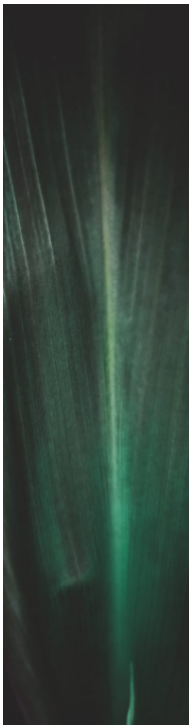


OLED дисплей SSD1306

Низьке енергоспоживання.
Висока контрастність та кути огляду.
Підтрима I2C протоколу

СТРУКТУРА СИСТЕМИ





АРХІТЕКТУРА ДЕСКТОПНОГО ЗАСТОСУНКУ

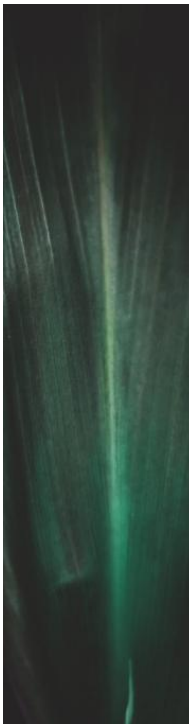
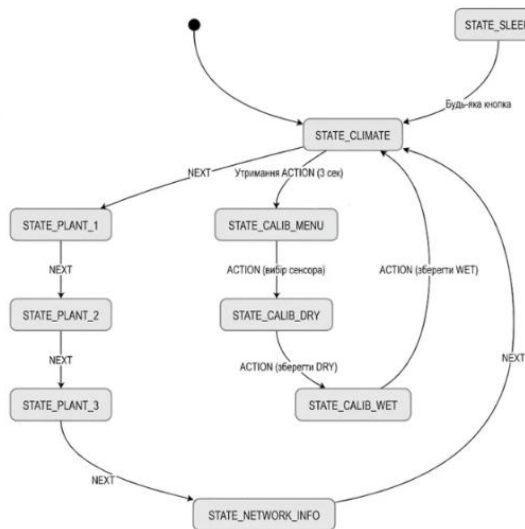
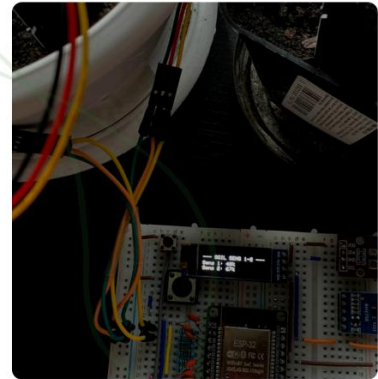
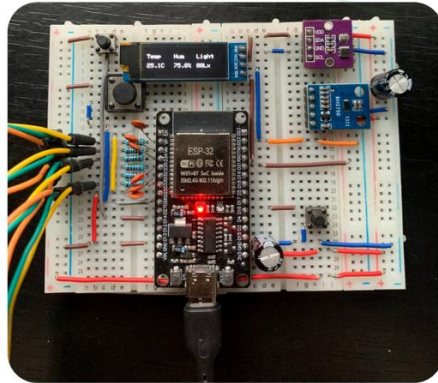


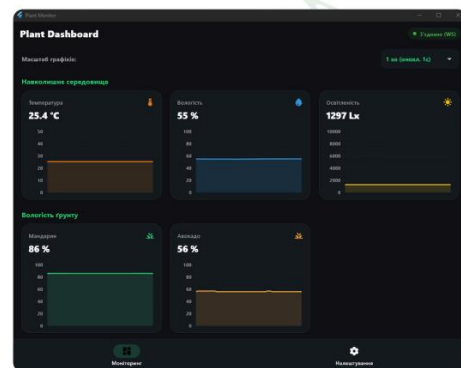
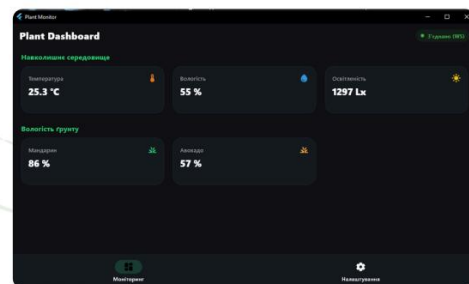
СХЕМА НАВІГАЦІЇ МЕНЮ ЛОКАЛЬНОГО ІНТЕРФЕЙСУ



ФІЗИЧНА РЕАЛІЗАЦІЯ ПРИСТРОЮ



КЛІЄНСЬКИЙ ЗАСТОСУНОК





НАСТУПНІ СТАДІЇ РОЗВИТКУ КОМПЛЕКСУ

Інтеграція механізмів для автоматичного поливу та керування фітолампами

Додавання ШІ-агенту для рекомендацій та порад щодо догляду рослин на основі даних

Кросплатформове портування клієнтського застосунку

Забезпечення доступу через Інтернет

Впровадження технології ESP-MESH

Проектування підсистеми автономного живлення



ВИСНОВКИ

Створено надійний АПК з собівартістю значно нижчою за аналоги

Створено десктопний клієнтський застосунок на базі фреймворку Flutter

Реалізовано локальну мережеву взаємодію

Забезпечено високий рівень відмовостійкості комплексу

Впроваджено алгоритми математичної обробки телеметрії

Підтверджено економічну доцільність та практичну цінність розробки

Відео презентація роботи комплексу

ДЯКУЮ ЗА УВАГУ!