МАГІСТЕРСЬКА РОБОТА

МР. КІм - 240426

Група КІм-24-2

Дидик Роман Юрійович

2025

Міністерство освіти і науки України
Івано-Франківський Національний Технічний Університет Нафти і Газу
Інститут Інформаційних Технологій
Кафедра комп'ютерних систем і мереж

Дидик Роман Юрійович

УДК 004.02

МАГІСТЕРСЬКА РОБОТА

Вирішення задачі автоматизованого керування параметрами середовища мікроферми

Комп'ютерна інженерія
(назва освітньої програми)

123 – Комп'ютерна інженерія
(шифр і назва спеціальності)

Дидик Р.Ю.
(підпис, ініціали та прізвище здобувача)

Науковий керівник *к.т.н., доцент, Слабінога М.О.*
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри *С.І. Мельничук*
(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент *Я.І. Заячук*
(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2025 рік

Івано-Франківський національний технічний університет нафти і газу

(повне найменування вищого навчального закладу)

Інститут *інформаційних технологій*

Кафедра *комп'ютерних систем і мереж*

Освітній ступінь *магіст*

Спеціальність *123 – Комп'ютерна інженерія*

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри _____

(С.І. Мельничук)

«_____» _____ 2025 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТОВІ

Дидик Роман Юрійович

(прізвище, ім'я, по батькові)

1. **Тема роботи** Вирішення задачі автоматизованого керування параметрами середовища мікроферми

керівник роботи *к.т.н., доцент, Слабінога М.О.*

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

затверджені наказом закладу вищої освіти від *" "2025 №*

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи *Матеріали і результати отримані під час проходження переддипломної практики, методичні вказівки, технічна література.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1 Аналітично-дослідницька частина

Розділ 2 Алгоритми і моделі рішень даного класу

Розділ 3 Побудова власного рішення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи

Розділ	**Прізвище, ініціали та посада консультанта**	Підпис, дата	
		завдання видав	завдання прийняв

7. **Дата видачі завдання**_____.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	**Назва етапів дипломного проекту (роботи)**	**Строк виконання етапів проекту (роботи)**	**Примітка**
1	*Дослідження аналогів та постановка задачі*		
2	*Опис архітектури системи та аналіз необхідних даних для автоматичного функціонування*		
3	*Побудова власного рішення*		
4	*Оформлення пояснювальної записки*		

Студент _____ Дидик Р.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Слабінога М.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить 87 сторінок, 36 рисунків, список використаних джерел із 30 найменуванями.

Метою дипломної роботи є розробка системи автоматизованого керування параметрами середовища мікроферми, яка забезпечує зменшення участі людини у процесі догляду за фермою, підтримку стабільних умов вирощування та можливість дистанційного моніторингу і аналізу даних.

Об'єкт дослідження: процес автоматизованого керування параметрами середовища мікроферми.

У першому розділі магістерської роботи проаналізовано актуальність задачі автоматизованого керування параметрами середовища мікроферми, виконано огляд існуючих аналогів у сфері вирощування зеленої кормової маси та мікрозелені, а також сформульовано вимоги до системи і визначено мету та основні задачі дослідження.

У другому розділі розглянуто алгоритми та моделі рішень даного класу, описано архітектуру системи керування на основі мікроконтролера ESP32, серверної частини, бази даних і веб-інтерфейсу. Проаналізовано необхідні дані для автоматичного функціонування системи та виконано вибір апаратних і програмних технологій.

У третьому розділі реалізовано апаратно-програмне рішення системи керування параметрами мікроферми. Розроблено серверну і клієнтську частини, реалізовано взаємодію з мікроконтролером через REST API, забезпечено збереження, аналіз і візуалізацію даних, а також проведено інтеграцію компонентів і тестування роботи системи в штатних та нештатних режимах.

Ключові слова: автоматизоване керування, мікроферма, IoT, ESP32, моніторинг середовища, веб-інтерфейс, REST API, аналіз даних.

ABSTRACT

The master's thesis consists of 87 pages, 36 figures, and a list of references containing 30 sources.

The aim of the master's thesis is to develop an automated control system for managing the environmental parameters of a micro-farm, which reduces human involvement in farm maintenance, ensures stable growing conditions, and enables remote monitoring and data analysis.

Object of the study: the process of automated control of environmental parameters of a micro-farm.

In the first chapter of the master's thesis, the relevance of the problem of automated control of micro-farm environmental parameters is analyzed. Existing solutions in the field of green fodder mass and microgreen cultivation are reviewed, and the system requirements, research objectives, and main tasks are defined.

The second chapter examines algorithms and models of solutions for this class of systems and describes the control system architecture based on the ESP32 microcontroller, server-side components, a database, and a web interface. The data required for the system's automatic operation are analyzed, and appropriate hardware and software technologies are selected.

In the third chapter, a hardware-software solution for the micro-farm environmental control system is implemented. Server-side and client-side components are developed, interaction with the microcontroller via a REST API is implemented, data storage, analysis, and visualization are ensured, and system integration and testing under normal and abnormal operating conditions are performed.

Keywords: automated control, micro-farm, IoT, ESP32, environmental monitoring, web interface, REST API, data analysis.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CEA – Controlled Environment Agriculture

HTTP – HyperText Transfer Protocol

JSON – JavaScript Object Notation

Wi-Fi – Wireless Fidelity

БД – База даних

GPIO – General Purpose Input/Output

SSR/SSG – Server-Side Rendering / Static Site Generation

API – Application Programming Interface

ЗМІСТ

ВСТУП

Сучасний розвиток сільського господарства дедалі більше орієнтується на впровадження автоматизованих та інформаційних технологій, що дозволяють підвищити ефективність виробництва, зменшити витрати ресурсів та мінімізувати залежність від людського фактора. Особливо актуальним це є для систем вирощування соковитої зеленої кормової маси та мікрозелені методом пророщування насіння у контрольованому середовищі без ґрунту, де стабільність параметрів середовища безпосередньо впливає на якість та обсяг врожаю.

Малі та середні фермерські господарства, а також локальні виробничі комплекси, дедалі частіше потребують компактних, енергоефективних та гнучких систем керування, здатних забезпечити автоматичний контроль поливу, освітлення, вентиляції та мікроклімату. Водночас значна частина існуючих рішень на ринку орієнтована на великі промислові ферми або має обмежений функціонал і не підтримує повноцінний дистанційний моніторинг та аналіз даних.

Особливої актуальності набувають системи, які не лише автоматизують технологічні процеси, а й забезпечують збереження та аналіз даних протягом усього циклу вирощування. Наявність історії вимірювань, журналу подій та результатів циклів дозволяє оптимізувати режими роботи, підвищувати врожайність та зменшувати експлуатаційні витрати, що є критично важливим для невеликих господарств з обмеженими ресурсами.

Об’єктом дослідження є процес автоматизованого керування параметрами середовища мікроферми.

Предметом дослідження є апаратно-програмні засоби та алгоритми керування параметрами середовища у системах вирощування зеленої кормової маси та мікрозелені.

Метою магістерської роботи є розробка системи автоматизованого керування параметрами середовища мікроферми, яка забезпечує зменшення

участі людини у процесі догляду за фермою, підтримку стабільних умов вирощування та можливість дистанційного моніторингу і аналізу даних.

Для досягнення поставленої мети в роботі необхідно вирішити такі основні задачі:

- проаналізувати існуючі рішення та підходи у сфері автоматизованих систем вирощування;
- сформувати вимоги до системи керування з урахуванням потреб малих і середніх фермерських господарств;
- спроєктувати архітектуру апаратно-програмного комплексу;
- реалізувати серверну частину системи з підтримкою зберігання та обробки даних;
- розробити адаптивний веб-інтерфейс для керування фермою та аналізу результатів;
- реалізувати апаратну частину на базі мікроконтролера з підтримкою сенсорів;
- провести інтеграцію та тестування системи, включно з перевіркою роботи у нештатних ситуаціях.

Практична цінність роботи полягає у можливості використання розробленої системи як універсальної платформи керування параметрами середовища, яка може бути інтегрована з реальними виконавчими компонентами різної потужності та адаптована для використання у різних аграрних сценаріях, зокрема на малих і середніх фермерських господарствах.

Наукова новизна роботи полягає в подальшому розвитку методів проєктування апаратно-програмних засобів для вирішення задачі автоматизованого керування параметрами мікроклімату мікроферми на основі централізованої логіки прийняття рішень, використання сучасних IoT-платформ та веб-орієнтованих інтерфейсів дистанційного керування і аналізу даних.

РОЗДІЛ 1 АНАЛІТИЧНО-ДОСЛІДНИЦЬКА ЧАСТИНА

Актуальність теми магістерської роботи зумовлена стрімким розвитком технологій контрольованого середовища вирощування рослин (Controlled Environment Agriculture, CEA), які набувають все ширшого застосування у різних галузях, від промислового тепличного виробництва до компактних систем для малих ферм та мікроферм. Одним із напрямів, що активно розвивається, є технології вирощування соковитої зеленої кормової маси для худоби та птиці шляхом пророщування насіння в умовах, повністю контрольованих за температурою, вологістю, освітленням та режимами зрошення. Такий метод не потребує ґрунту, спирається на використання води як основного джерела живлення та дозволяє отримувати стабільний, екологічно чистий та поживний корм протягом усього року.

На відміну від великих фермерських комплексів, які можуть дозволити собі дорогі сучасні автоматизовані системи, малі господарства та сімейні міні-ферми мають значно обмежені ресурси - як фінансові, так і трудові. Саме в цьому сегменті існує найбільший попит на доступні, компактні та легко обслуговувані системи автоматизації вирощування зеленої маси. Ручний догляд за фермою, що передбачає регулярний контроль параметрів середовища, поливу, освітлення та вентиляції, потребує значних часових витрат та постійної присутності оператора. Будь-які відхилення можуть призводити до погіршення якості корму або зниження продуктивності.

Світові тенденції переходу до точного землеробства, вертикальних ферм, гідропонічних систем та систем автоматизованого моніторингу створюють підґрунтя для впровадження компактних CEA-рішень на рівні малого та середнього агровиробництва. Зокрема, технології пророщування зерна для отримання зеленої кормової маси демонструють високу ефективність щодо продуктивності, стабільності врожаю та економного використання води. Проте їх широке застосування стримує складність ручного догляду та висока залежність від людського фактора.

Автоматизація таких процесів дозволяє істотно зменшити участь людини, мінімізувати часові витрати, усунути ризики помилок та забезпечити стабільне підтримання оптимальних параметрів середовища. Створення інтегрованої апаратно-програмної системи, здатної контролювати і керувати всіма ключовими аспектами мікроклімату: температурою, вологістю, поливом та освітленням, є актуальним як з практичного, так і з науково-технічного погляду.

Крім того, розвиток міського фермерства та популяризація малих форм агровиробництва формують потребу у доступних технологічних рішеннях, які можуть працювати на базі недорогих апаратних платформ і бути адаптованими під різні масштаби виробництва. Комбінація компактності, енергоефективності та можливості дистанційного керування робить такі системи перспективними для широкого кола користувачів.

У підсумку, розробка системи автоматизованого керування параметрами середовища мікроферми для вирощування соковитої зеленої кормової маси є надзвичайно актуальним завданням, яке сприяє впровадженню сучасних цифрових технологій у сферу малих агровиробництв, підвищує ефективність роботи фермерів та створює нові можливості для оптимізації трудових витрат.

1.1 Дослідження аналогів

У рамках даної роботи аналізуються не стільки повні ферми чи виробничі лінії, скільки підходи до автоматизованого керування параметрами середовища в існуючих рішеннях для вирощування зеленої кормової маси та мікрозелені.

Основна увага приділена:

- способам автоматизації (полив, вентиляція, освітлення, клімат);
- наявності або відсутності систем моніторингу;
- використанню веб-інтерфейсів та віддаленого керування;
- реалізації методів ebb-and-flow (flood & drain);

- рівню залучення оператора.

Це дозволяє сформувати вимоги до системи керування, яку розробляється в межах магістерського дослідження.

1.1.1 Огляд основних прикладів на ринку

Великі комерційні системи (контейнери, автоматизовані рішення)

У комерційних контейнерних фермах та вертикальних установках основний акцент робиться на автоматизоване керування параметрами середовища: температурою, вологістю, поливом, режимом освітлення. Такі системи включають комплексний набір обладнання, але саме автоматизація є ключовим елементом, що забезпечує стабільний цикл вирощування з мінімальним втручанням людини. Приклади: HydroGreen / CubicFarms та ряд постачальників контейнерних рішень, які позиціонують свої продукти як «fully automated» системи для щоденного виробництва корму. Ці системи показують, що на комерційному рівні існує попит на повністю автономні рішення з малою участю оператора.

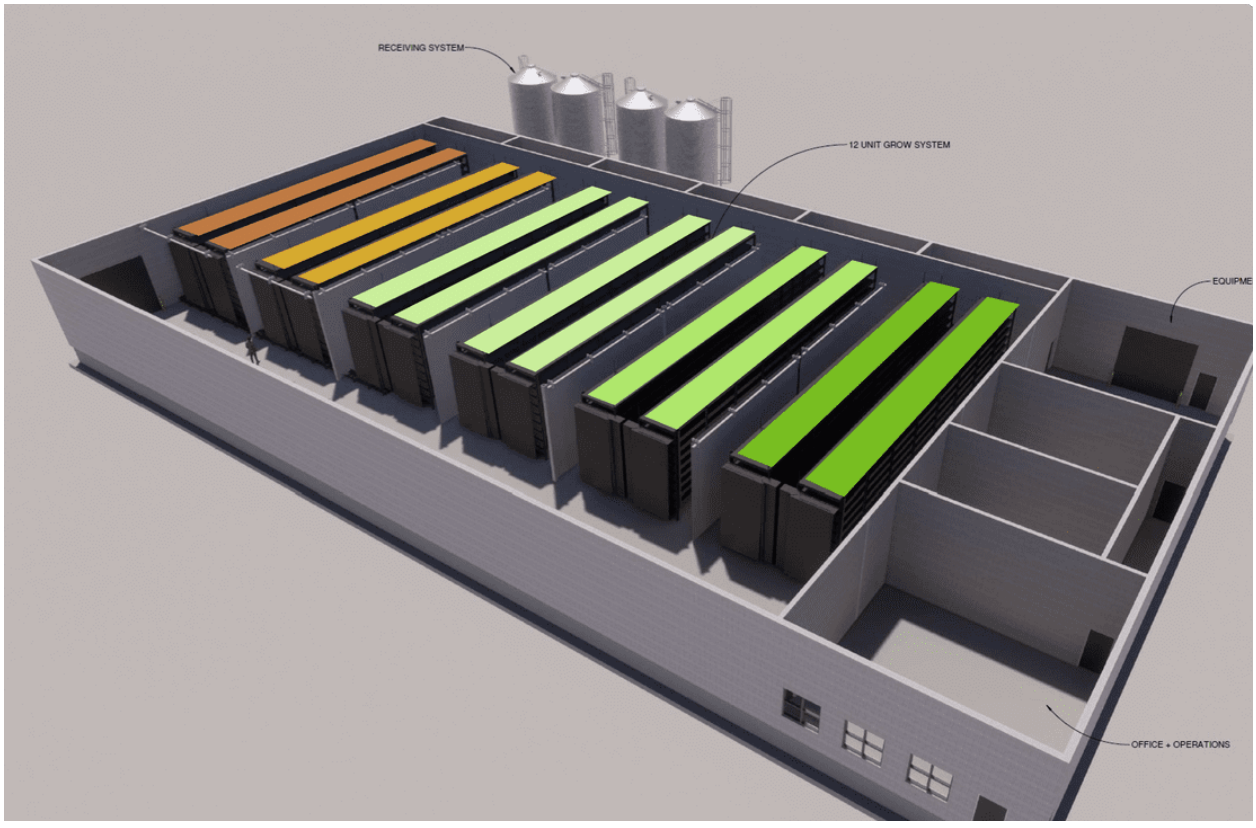

Рисунок 1.1 Комплексне рішення Automated Vertical Pastures від HydroGreen

Середні/малі комерційні установки (кімнати, полиці)

Henan Miracle пропонує обладнання для гідропонічного пророщування корму, у якому застосовуються:

- таймерні системи поливу,
- базовий клімат-контроль,
- системи освітлення та вентилювання,
- автоматизація циклів живлення лотків.

Попри те, що компанія пропонує повністю готові ферми, у контексті цієї роботи важливо інше: їхні продукти демонструють типові підходи до автоматизації, які можуть бути реалізовані й у менших установках. Саме технологічні рішення з керування параметрами середовища становлять дослідницький інтерес.

Рисунок 1.2 Сільськогосподарська гідропонічна кормовиробна система від Henan Miracle Industry

Малі ферми і системи для мікрозелені (tray-системи, локальні вертикальні стелажі)

У сегменті мікрозелені переважають невеликі системи:

- багаторівневі стелажі з LED-освітленням,
- прості помпи з таймерами поливу,
- мінімальні або відсутні системи моніторингу,
- майже повна відсутність веб-інтерфейсів.

Багато з них застосовують метод ebb-and-flow, але керування здійснюється переважно вручну або через примітивні контролери.

Це підкреслює брак недорогих рішень, які б дозволяли:

- віддалено керувати параметрами середовища,
- збирати та аналізувати дані,
- отримувати сповіщення про відхилення умов.

Рисунок 1.3 Компактне рішення від MicroGreens World

1.1.2 Принципи зрошення: чи застосовують метод затоплення (flood & drain)?

Метод затоплення і зливу (ebb-and-flow / flood & drain) широко застосовується в системах пророщування корму: лотки періодично заповнюються живильним розчином або просто водою, потім зливаються назад у бак, що дозволяє підтримувати оптимальну вологість насіння/ростків без постійного перетримування у воді. У багатьох системах, від великих контейнерних рішень до малих установок для мікрозелені, використовується періодичне затоплення лотків із подальшим зливом води.

Але рівень автоматизації цього процесу суттєво відрізняється:

- у великих фермах ним керує контролер середовища,
- у системах середнього сегмента — таймерні помпи,
- у малих системах — ручне керування або окремі реле.

Цей огляд показує, що реалізація інтелектуального керування flood & drain у компактній формі залишається актуальним завданням.

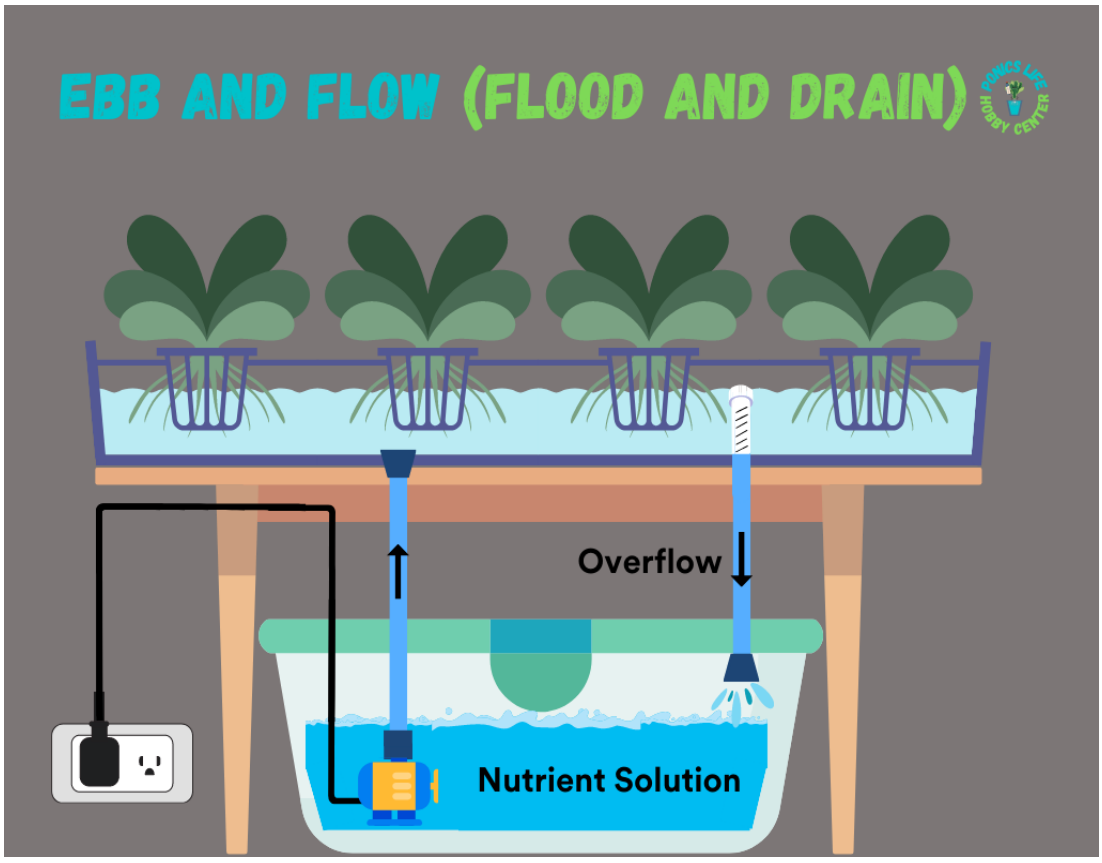

Рисунок 1.4 Принцип роботи flood & drain гідропоніки

1.1.3 Порівняння за ключовими ознаками

Критерій	Великі системи	Системи середнього сегмента (Miracle)	Малі установки / мікрозелень
Автоматизація	Висока, багатомодульна	Середня, локальна	Мінімальна
Полив	Автоматичний, керований контролером	Таймерні цикли	Прості таймери або вручну
Flood & drain	Так	Частково	Так, але без розвиненого контролю
Вентиляція	Автоматична	Базова	Рідко автоматизована
Збір даних	Є, включно з аналітикою	Обмежений	Майже відсутній
Веб-інтерфейс	Є в дорогих моделях	Зазвичай відсутній	Відсутній
Роль оператора	Мінімальна	Середня	Висока

1.1.4 Виявлення прогалин та можливостей розробки

Аналіз аналогів демонструє:

1. Відсутність компактних систем керування, які б могли інтегруватися у невеликі ферми та мікроферми.
2. Нестача рішень із веб-керуванням, адаптивними інтерфейсами та хмарним моніторингом для недорогих установок.
3. Низький рівень автоматизації у сегменті мікрозелені, де навіть ключові процеси (полив, вентиляція, освітлення) часто керуються вручну.
4. Відсутність систем, що інтегруються з планувальниками (наприклад, Google Calendar) для нагадувань і ведення журналів вирощування.
5. Неавтоматизоване flood & drain у більшості малих систем, що створює простір для інтелектуальної керуючої платформи.

1.1.5 Висновки розділу

Існуючі рішення переважно пропонують повні ферми або обладнання, тоді як тема машістерської роботи захоплює іншу нішу - створення універсальної системи автоматизованого керування параметрами середовища, яку можна:

- інтегрувати у різні типи мікроферм,
- адаптувати під різні методи вирощування,
- масштабувати під потреби малого та середнього господарства.

Таким чином, розробка саме системи керування є актуальною, новою та необхідною для сегмента малих агровиробників і фермерських господарств.

1.2 Постановка задачі

Системи вирощування зеленої кормової маси та мікрозелені в контрольованому середовищі використовуються як у промислових аграрних підприємствах, так і в невеликих фермерських чи приватних господарствах, що потребують стабільного, відтворюваного та передбачуваного процесу пророщування насіння. Попри активний розвиток гідропонних технологій, значна частина рішень на ринку зосереджена на апаратних платформах, тоді як гнучкі, доступні та масштабовані системи керування параметрами середовища залишаються обмеженими.

Наявні автоматизовані ферми часто мають або закриті пропрієтарні системи керування, або спрощені контролери без інтеграції зі смартфоном, адаптивного веб-інтерфейсу, зручної аналітики чи віддаленого моніторингу. Для малих господарств це створює бар’єр у впровадженні технологій через високу вартість комплексних рішень та недостатню гнучкість готових контролерів.

Малі ферми, що займаються вирощуванням зеленої кормової маси або мікрозелені, мають обмежені трудові ресурси. Тому постає потреба в розробці компактної, енергоефективної та недорогої системи керування, яка

зможе автоматично підтримувати параметри середовища - зокрема полив та освітлення - незалежно від конструктивних особливостей самої ферми.

Роль оператора має бути зведена до мінімуму завдяки автоматизації рутинних процесів, а також можливості контролю через веб-інтерфейс та отримання системних сповіщень.

Отже виникає потреба у створенні апаратно-програмної системи керування мікрофермою, яка забезпечуватиме моніторинг параметрів середовища в реальному часі, автоматизацію ключових технологічних процесів та інтуїтивно зрозумілий інтерфейс для віддаленої роботи.

Мета роботи

Розробити систему автоматизованого керування параметрами середовища мікроферми, що дозволить зменшити трудовитрати оператора шляхом автоматизації процесів поливу, освітлення та моніторингу, а також забезпечити можливість дистанційного керування через адаптивний веб-інтерфейс та інтегровані календарні сервіси.

Основні задачі

Для досягнення поставленої мети передбачається виконання таких задач:

- Проаналізувати існуючі рішення у сфері автоматизованих гідропонних систем та систем для вирощування зеленої кормової маси.
- Сформувати вимоги до архітектури системи керування з урахуванням потреб невеликих фермерських господарств.
- Розробити апаратний модуль керування, який включає: керування поливом методом затоплення (ebb-and-flow) та керування освітленням з можливістю налаштування режимів,
- Вимірювання температури та інших параметрів середовища.
- Розробити серверну частину для обробки даних, зберігання історії та взаємодії з веб-інтерфейсом.

- Створити адаптивний веб-інтерфейс для керування системою, перегляду статистики та формування подій.
- Реалізувати інтеграцію з календарними сервісами для сповіщень що до технічного обслуговування станції.
- Забезпечити роботу демонстраційного макета системи на низьковольтному живленні (5В).
- Провести тестування системи в рамках лабораторного макета.

Вимоги до системи

Функціональні вимоги

1. Система має забезпечувати автоматичний або ручний запуск циклів поливу методом затоплення з подальшим відведенням води.
2. Повинна бути можливість налаштування розкладу поливу, освітлення та вентиляції.
3. Мікроконтролер повинен вимірювати актуальні параметри середовища та передавати їх на сервер.
4. Система повинна забезпечувати зберігання історичних даних у базі.
5. Веб-інтерфейс має бути адаптивним, відображати параметри в реальному часі та підтримувати роботу зі смартфона.
6. Оператор повинен мати можливість дистанційного керування.
7. Сповіщення про події (наприклад відхилення температури чи збої в поливі) мають передаватися оператору.

Нефункціональні вимоги

1. Система повинна бути енергоефективною та працювати від низьковольтних джерел живлення (5В у макеті).
2. Архітектура має бути модульною та масштабованою.
3. Інтерфейс повинен бути інтуїтивно зрозумілим без спеціальної технічної підготовки.
4. База даних повинна забезпечувати цілісність історичних даних.

5. Система має залишатися контрольованою при тимчасовій втраті інтернет-з’єднання.

6. Програмне забезпечення повинно підтримувати подальше розширення функціональності.

РОЗДІЛ 2 АЛГОРИТМИ І МОДЕЛІ РІШЕНЬ ДАНОГО КЛАСУ

Проєктування систем автоматизованого керування параметрами середовища для мікроферм вимагає узгодженого поєднання апаратних компонентів, програмної логіки та мережевих механізмів обміну даними. На відміну від комплексних гідропонних установок, орієнтованих на промислове використання, розроблювана система фокусується на створенні універсального програмного рішення, здатного працювати із широким спектром апаратних конфігурацій ферми.

Основна логіка автоматизації, аналізу даних та прийняття рішень реалізована на серверній частині, тоді як роль мікроконтролера зводиться до виконання команд, збору показників датчиків та передачі їх на сервер у реальному часі. Такий підхід забезпечує масштабованість, гнучкість під час зміни апаратної частини та можливість розширення системи без зміни базових алгоритмів.

У цьому розділі розглядається архітектура системи, структура потоків даних між компонентами, логіка функціонування режимів керування, вибір технологій для апаратної та програмної частини, а також моделі, які використовуються для автоматичного контролю параметрів середовища. Додатком до опису виступають діаграми, що відображають взаємодію компонентів та алгоритмічні процеси виконання команд.

2.1 Опис архітектури системи

Розроблена система складається з п’яти основних компонентів(рис 2.1):

1. Мікроконтролер (ESP32)

ESP32 виконує роль апаратного контролера, який:

- зчитує показники з датчика температури й вологості (DHT11);
- вимірює рівень води в ємності за допомогою ультразвукового датчика HC-SR04P;

- передає зібрані дані на сервер у форматі JSON через HTTP-запити;
- отримує від сервера команди керування роботою модулів ферми.

У демонстраційному макеті замість насосів, клапанів, вентиляторів та освітлення використовуються світлодіоди, які індикують включення відповідних модулів. Це дозволяє зосередитися на логіці керування, а не на силовій частині ферми.

2. Мережевий рівень

ESP32 має вбудований Wi-Fi модуль, що дозволяє йому:

- виконувати HTTP-запити до сервера кожні 5 секунд (інтервал змінний),
- отримувати JSON-відповідь із серверними командами,
- функціонувати у локальній мережі або через точку доступу.

Такий принцип дає можливість використовувати систему як у стаціонарних, так і в портативних умовах.

3. Серверна частина (Node.js + Express)

Бекенд є центральним елементом системи керування та виконує такі функції:

- обробка даних від мікроконтролера та фронтенду;
- прийняття рішень відповідно до робочого режиму ферми (manual або scheduled);
- генерування команд для мікроконтролера (включення/вимкнення модулів);
- контроль циклів вирощування згідно з планом;
- відправлення сповіщень про помилки чи попередження на електронну пошту;
- збереження історичних даних у MongoDB.

У режимі scheduled сервер визначає, коли вмикати полив, коли проводити злив води, який рівень наповнення має бути досягнуто, та коли вмикати освітлення.

У режимі manual оператор керує параметрами безпосередньо через веб-інтерфейс.

4. База даних (MongoDB + Mongoose)

У MongoDB зберігаються такі колекції:

- плани вирощування — шаблони циклів з розкладами;
- цикли ферми — реальні вирощувальні цикли, створені на основі планів;
- історичні показники — температура, вологість, рівень води;
- налаштування ферми — режими роботи, параметри тривалості та умов;
- логи системи — помилки, попередження, службова інформація.

Нереляційна структура дозволяє гнучко додавати нові параметри середовища без зміни схеми БД.

5. Фронтенд (Next.js)

Веб-інтерфейс забезпечує:

- дистанційний контроль параметрів ферми;
- перегляд стану модулів та сенсорних показників;
- керування режимами (manual / scheduled);
- створення нових розкладів і планів;
- моніторинг циклів вирощування;
- перегляд логів та звітів;

Фронтенд та бекенд функціонують як окремі сервіси, обмінюючись даними через HTTP API.

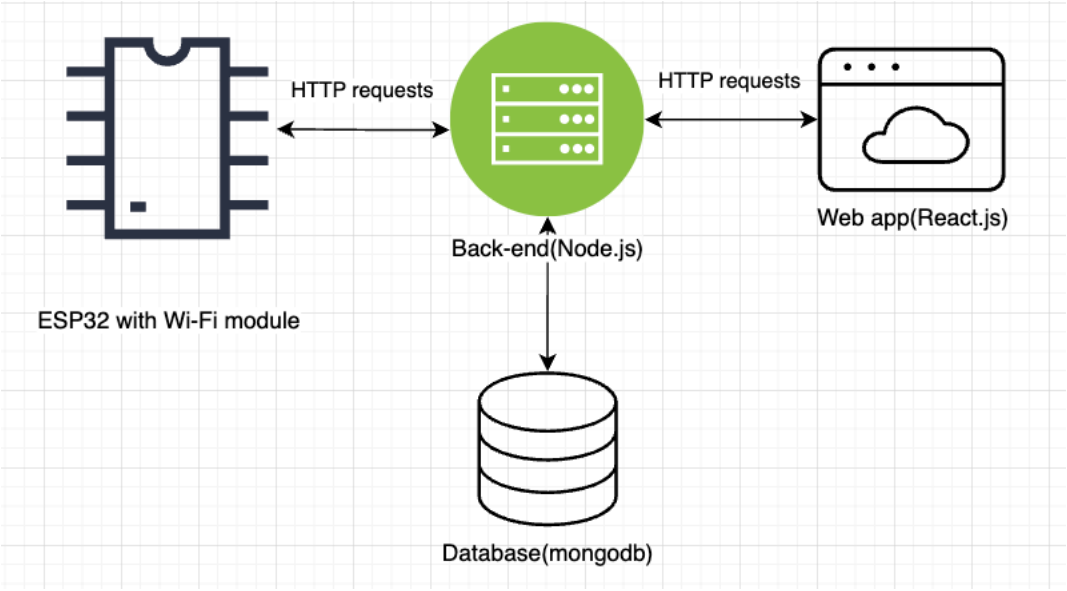

Рисунок 2.1 Взаємодія між компонентами системи

2.2 Аналіз необхідних даних для автоматичного функціонування

Для забезпечення повноцінної та стабільної роботи системи автоматизованого керування параметрами мікроферми необхідно визначити структуру даних, що використовуються на різних етапах функціонування: під час моніторингу, планування, формування рішень та виконання команд.

У розробленій системі всі дані поділено на три ключові групи, відповідно до їхнього призначення та ролі у процесі автоматизації:

1. Дані, що збираються з сенсорів та зберігаються у базі як історичні записи.
2. Дані, що генеруються системою відповідно до режимів роботи та алгоритмів керування.
3. Дані, які вводить користувач і які визначають логіку вирощувальних циклів.

Такий поділ дозволяє чітко структурувати інформаційні потоки та забезпечує прозорість алгоритмів, що реалізуються на сервері.

Дані, що збираються автоматично та зберігаються у базі

Ця група охоплює всі параметри, що надходять від сенсорів ESP32 та записуються в базу даних для аналітики, контролю працездатності та формування звітів.

Температура повітря (°C)

Отримується з датчика DHT11. Використовується для:

- оцінки умов у фермі;
- генерації попереджень (warning) у разі небажаних значень;
- діагностики помилкових показів датчика (error);
- формування історичних записів.

Вологість повітря (%)

Також зчитується DHT11. Зберігається для аналізу стану середовища та створення статистики.

Рівень води

Вимірюється ультразвуковим датчиком HC-SR04P. Значення зчинуються для:

- контролю роботи алгоритмів поливу та зливу,
- фіксації можливих збоїв,
- побудови історії циклів поливу.

Логи, попередження і помилки

Вони формуються в момент некоректних значень датчика або невідповідності фактичної поведінки очікуваній.

Система зберігає:

- системні помилки (error),
- попередження (warning),
- службові записи.

Ці дані є важливими для аналізу роботи ферми оператором.

Дані, що генеруються системою під час роботи (керуючі дані)

Це ключові параметри, які визначають, які дії має виконувати мікроконтролер.

Вони формуються виключно сервером відповідно до:

- обраного режиму роботи (manual / scheduled),
- циклу вирощування (у режимі scheduled),
- актуальних показників датчиків,
- логіки алгоритмів (наприклад, досягнуто рівень води → вимкнути насос).

До цих даних належать:

Булеві команди для модулів ферми

- light – включення/вимкнення освітлення, за потреби з параметром яскравості;
- pump – дозвіл на наповнення ємності;
- drain – дозвіл на злив.

Ці команди є центральними для роботи як фізичної ферми, так і демонстраційного макета (де це індикується світлодіодами).

Поводження даних у різних режимах:

Manual mode

У цьому режимі користувач керує модулями безпосередньо.

Система зберігає:

- поточний стан ферми у колекції Farm (light, pump, drain),
- значення вмикаються та вимикаються за командою оператора.

Цей режим не використовує планів і циклів.

Scheduled mode

У цьому режимі всі команди генеруються автоматично на основі циклу, створеного з плану.

У базі зберігаються:

- lightSchedule — графік роботи освітлення
- pumpSchedule — масив операцій поливу
- drainSchedule — графік зливу

У кожному з них містяться:

- час початку,
- тривалість,
- бажане значення (on/off або інтенсивність).

Сервер аналізує ці масиви та вирішує:

- чи настав час виконувати дію,
- чи модуль має бути вимкненим,
- чи очікується інше підрядне завдання.

Такі дані формують основу автоматичного керування.

Дані, які додає користувач (планування вирощування)

Користувач створює плани (Plan) — універсальні шаблони вирощування, на основі яких формуються цикли.

План може містити:

- тривалість циклу;
- тривалість та періодичність поливу;
- параметри освітлення;
- цільові рівні води;
- назву рослини;
- кількість посівного матеріалу.

Після запуску ферми план перетворюється на цикл (Cycle) із конкретними розкладами, а пізніше історією виконаних дій.

Плани є користувацькою «надбудовою», яка дозволяє фермі працювати автономно і повторювано.

2.3 Вибір технологій

Проєктування системи автоматизованого керування параметрами середовища мікроферми вимагає ретельного підбору апаратних і програмних засобів, здатних забезпечити стабільну роботу, інтегрованість та можливість подальшого масштабування. У цьому підрозділі наведено обґрунтований вибір компонентів, що формують апаратну й програмну основу системи.

2.3.1 Вибір апаратної частини

Мікроконтролер: ESP32

Для реалізації системи був обраний мікроконтролер ESP32, який найкраще відповідає вимогам проєкту завдяки таким характеристикам:

- Вбудований Wi-Fi-модуль, що усуває потребу в додаткових комунікаційних платах.
- Значний обсяг оперативної пам’яті та Flash, що дозволяє обробляти складніші алгоритми та JSON-структури.
- Велика кількість GPIO-виводів, що забезпечує інтеграцію з кількома датчиками та виконавчими модулями.
- Низьке енергоспоживання, що важливо для демонстраційного макета на живленні 5 В.
- Підтримка двох ядер і багатозадачності, що підвищує надійність роботи при одночасному виконанні мережевих операцій та зчитуванні датчиків.
- Висока популярність і підтримка спільноти, що полегшує розробку, налагодження та розширення функцій.

Таким чином, ESP32 забезпечує оптимальний баланс між функціональністю, вартістю та можливістю масштабування.

Датчик температури та вологості: DHT11

У проєкті використано DHT11. Хоча цей датчик має нижчу точність, ніж, наприклад, DHT22, його можливостей достатньо для задачі контролю параметрів мікроферми, де невеликі відхилення не впливають критично на процес вирощування зеленої кормової маси або мікрозелені. Основні аргументи вибору:

- достатня точність для умов ферми;
- широке застосування та простота підключення;
- низьке енергоспоживання;
- доступність і стабільність роботи.

Датчик рівня води: ультразвуковий HC-SR04P

Вибір HC-SR04P обумовлений такими перевагами:

- працює від 3.3 В, що сумісно з ESP32;
- забезпечує достатню точність для вимірювання рівня до поверхні води;
- більш довговічний так як не контактує з водою;
- простий у використанні та надійний у довготривалій роботі;
- дає змогу визначати рівень води для алгоритмів затоплення та зливу.

Контактні датчики, на відміну від ультразвукових, мають низьку точність і можуть давати хибні спрацьовування, тому HC-SR04P є оптимальним рішенням.

2.3.2 Вибір способу комунікації між мікроконтролером і сервером

У проєкті використано HTTP-запити кожні 5 секунд. Причини такого вибору:

- Простота реалізації на ESP32 та Node.js.
- Висока надійність у нестабільних мережевих умовах: у разі переривання зв’язку ферма безпечно чекає наступного запиту.
- JSON-формат відповіді легко обробляється на мікроконтролері.
- Менша складність у порівнянні з MQTT чи WebSocket, що важливо для демонстраційного макета.

Така схема забезпечує стабільну роботу та передбачуваність алгоритмів.

2.3.3 Вибір технологій серверної частини

Сервер: Node.js + Express

Node.js було обрано з таких причин:

- Одна мова програмування (JavaScript) на всіх рівнях системи: сервер, фронтенд, комунікація. Це спрощує розробку й підтримку.
- Асинхронна модель чудово підходить для обробки великої кількості коротких запитів від мікроконтролера.
- Велика екосистема пакетів, яка прискорює розробку.
- Легка інтеграція з базами даних, JSON-форматом і REST-архітектурою.

Express.js — мінімалістичний і гнучкий фреймворк, що дозволяє формувати API з чіткою структурою.

База даних: MongoDB + Mongoose

Вибір MongoDB пояснюється:

- Документно-орієнтованою структурою, яка природно підходить для зберігання:
  - історії вимірювань,
  - шаблонів планів,
  - структур циклів,
  - параметрів ферми.
- Гнучкою схемою, що дозволяє легко змінювати структуру даних.
- Хорошою масштабованістю, що важливо при переході від макета до реальної ферми.
- Ефективністю при записі великих обсягів телеметрії.

Mongoose дає зручні моделі та валідатори, що спрощує роботу з даними.

Коротке порівняння альтернатив

- PostgreSQL — потужна, але потребує суворих схем; складніше зберігати структуровані, але змінні JSON-структури.
- SQLite — не підходить для системи, яка постійно пише історію вимірювань.

- InfluxDB — добре для телеметрії, але гірше підходить для планів, циклів і користувацьких структур.

MongoDB виявилася найуніверсальнішим варіантом.

2.3.4 Вибір технологій фронтенду

Next.js

Цей фреймворк обрано через:

- сучасний підхід до розробки,
- високу продуктивність,
- вбудований рендеринг (SSR/SSG),
- зручність структуризації коду,
- потужні можливості для створення веб-панелей з частими оновленнями даних.

Він забезпечує швидку й плавну роботу адаптивного інтерфейсу.

Axios

- зручний інструмент для HTTP-запитів;
- підтримує перехоплення помилок та централізовану конфігурацію;
- добре інтегрується в Next.js та клієнтську частину.

UI-бібліотека: shadcn/ui

Було обрано shadcn, оскільки вона:

- має велику кількість готових компонентів;
- добре інтегрується з Tailwind CSS;
- дозволяє швидко збирати інтуїтивно зрозумілий інтерфейс;
- є сучасним, активним і зручним рішенням для побудови панелей керування.
-

2.3.5 Масштабованість і перспективи розвитку

Обрана архітектура дозволяє:

- легко додавати нові датчики;
- інтегруватися з іншими фермерськими системами;
- розширити логіку автоматизації (наприклад, PID-регулятори);
- перейти від одного макета до мережі мікроферм;
- зберігати значні обсяги історичних даних;
- замінити HTTP-polling на MQTT чи WebSocket без зміни загальної логіки.

Таким чином, система здатна еволюціонувати разом із потребами фермерського господарства.

2.4 Проєктування системи

Проєктування системи автоматизованого керування параметрами середовища мікроферми є ключовим етапом розробки, оскільки саме на цьому рівні формалізується логіка взаємодії між користувачем, серверною частиною та апаратним контролером. Для складних багатокомпонентних систем важливо не лише реалізувати функціональність, але й забезпечити її зрозумілість, передбачуваність та безпечну поведінку у різних режимах роботи.

Використання діаграм дозволяє наочно відобразити:

- послідовність виконання алгоритмів;
- умови переходу між станами;
- взаємодію програмних і апаратних компонентів;
- логіку прийняття рішень системою в автоматичному та ручному режимах.

З точки зору користувача, продумана логіка керування забезпечує прозору роботу ферми, мінімізує кількість ручних дій та знижує ризик помилок. З точки зору системи, чітко визначені алгоритми та стани дозволяють реалізувати стабільну, масштабовану та відмовостійку архітектуру. Саме тому в даному розділі основна увага приділяється

проєктуванню алгоритмів керування, представлених у вигляді логічних діаграм.

2.4.1 Логіка поливу методом затоплення та зливу

Одним із ключових технологічних процесів мікроферми є полив методом періодичного затоплення з подальшим зливом води (ebb-and-flow). Запропонована логіка поливу побудована таким чином, щоб забезпечити достатнє зволоження піддону, уникнути переливу та мінімізувати участь оператора. Діаграма логіки полику зображена на рисунку 2.2.

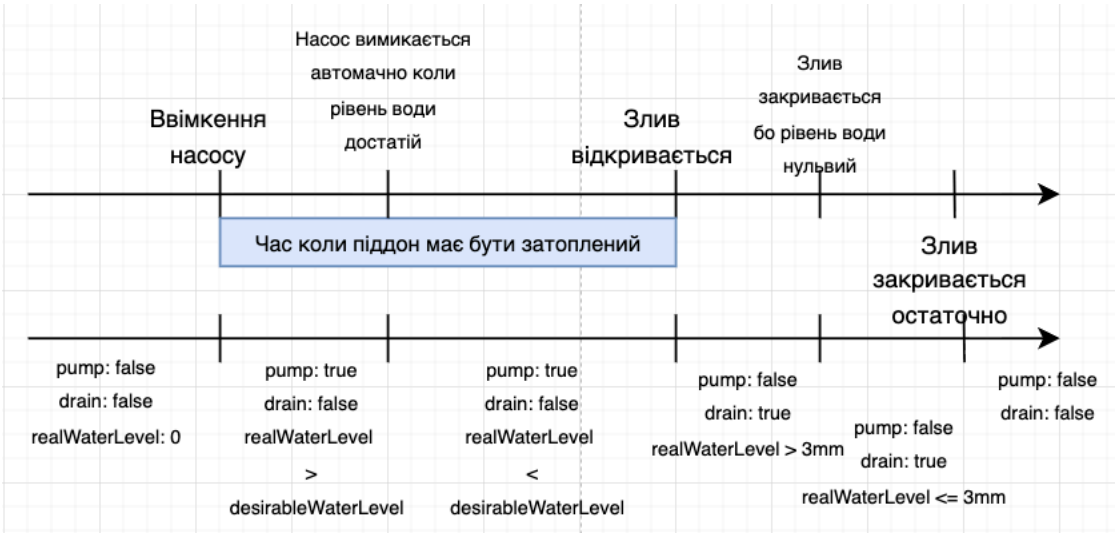

Рисунок 2.2 Логіка поливу

Процес поливу складається з кількох послідовних фаз:

1. Початковий стан.

У системі насос і дренаж вимкнені, рівень води дорівнює нулю або мінімальному значенню. Система очікує команди на початок поливу від серверної частини.

2. Фаза наповнення (затоплення)

Після отримання дозволу на полив (pump = true, drain = false) вмикається насос. Подача води триває до моменту, поки фактичний рівень води не досягне або не перевищить заданий цільовий рівень

(desirableWaterLevel). Досягнення потрібного рівня води є умовою автоматичного вимкнення насоса, незалежно від тривалості фази затоплення.

3. Фаза утримання води

Протягом заданого часу піддон залишається затопленим. У цей період насос вимкнений, дренаж закритий, а система лише контролює рівень води та очікує команди переходу до фази зливу.

4. Фаза зливу

Після завершення періоду затоплення сервер надсилає команду на злив (pump = false, drain = true). Дренаж відкривається, і вода поступово відводиться з піддону.

5. Завершення циклу

Коли рівень води зменшується до мінімального значення (близького до нуля або заданого порогу, наприклад ≤ 3 мм), дренаж автоматично вимикається. Система повертається у початковий стан та готова до наступного циклу поливу.

Запропонована логіка забезпечує:

- автоматичне відключення насоса при досягненні потрібного рівня води;
- захист від переливу;
- чітке розділення фаз затоплення, утримання та зливу;
- можливість гнучкого налаштування тривалості кожної фази на серверній частині.

Така модель є універсальною та може бути застосована як у демонстраційному макеті, так і в реальних мікрофермах з різними об’ємами резервуарів і способами розміщення датчиків.

2.4.2 Алгоритм роботи апаратної частини системи

Апаратна частина системи автоматизованого керування параметрами середовища реалізована на базі мікроконтролера ESP32 та функціонує у тісній взаємодії з серверною частиною. Основне призначення контролера

полягає у зборі сенсорних даних, періодичному обміні інформацією з сервером та виконанні керуючих команд відповідно до поточного режиму роботи ферми. Алгоритм роботи апаратної частини зображено на рисунку 2.3.

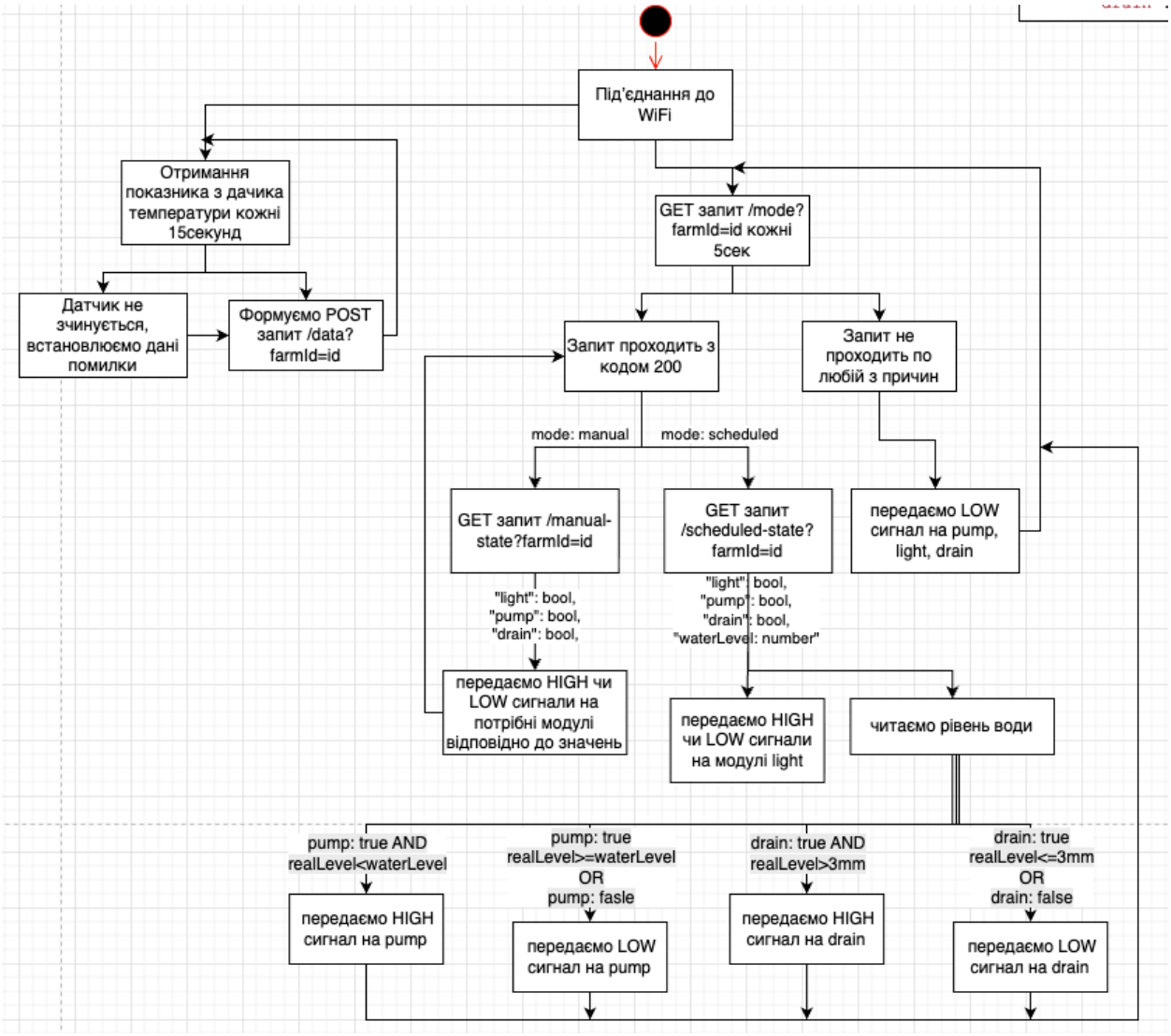

Рисунок 2.3 Алгоритм роботи апаратної частини

Алгоритм роботи ESP32 побудований за принципом циклічного виконання операцій із чітко визначеними часовими інтервалами, що забезпечує стабільність системи та її передбачувану поведінку у випадку збоїв зв'язку або некоректних даних.

Початковий етап роботи

Після запуску мікроконтролер виконує ініціалізацію апаратних та програмних компонентів, зокрема встановлює з’єднання з бездротовою мережею Wi-Fi. Успішне підключення до мережі є необхідною умовою для подальшої взаємодії із серверною частиною системи.

Після встановлення з’єднання всі виконавчі модулі (насос, дренаж, освітлення) переводяться у безпечний початковий стан з вимкненими вихідними сигналами.

Збір та передача сенсорних даних

З фіксованим інтервалом часу (кожні 15 секунд) ESP32 зчитує показники температури та вологості з відповідного датчика. Отримані значення формуються у структуру даних та передаються на сервер у вигляді POST-запиту.

У разі, якщо датчик не відповідає або повертає некоректні значення, система фіксує помилку та надсилає на сервер резервні значення, що дозволяє зберегти цілісність часових рядів у базі даних і забезпечити безперервність збору інформації.

Отримання режиму роботи ферми

Паралельно зі збором сенсорних даних, з меншим інтервалом (кожні 5 секунд), ESP32 надсилає GET-запит на сервер для визначення поточного режиму роботи ферми. Сервер повертає один із можливих режимів:

- manual — ручний режим керування;
- scheduled — автоматичний режим відповідно до циклу;
- або порожню/некоректну відповідь у разі збою.

Якщо відповідь сервера не отримана або є некоректною, контролер переходить у безпечний стан, вимикаючи всі виконавчі модулі.

Логіка роботи в ручному режимі (manual)

У разі, якщо сервер повідомляє про активний ручний режим, ESP32 виконує окремий запит для отримання поточних команд керування. У відповіді містяться булеві значення для кожного з модулів: освітлення, насоса та дренажу.

Отримані значення безпосередньо передаються на відповідні вихідні піни контролера у вигляді логічних сигналів HIGH або LOW. У цьому режимі ESP32 не виконує жодної додаткової перевірки рівня води чи інших параметрів, оскільки повний контроль за станом системи здійснює користувач через веб-інтерфейс.

Логіка роботи в автоматичному режимі (scheduled)

У автоматичному режимі ESP32 отримує від сервера розширений набір параметрів, що включає:

- стан освітлення;
- дозвіл на роботу насоса;
- дозвіл на злив води;
- цільовий рівень води.

На основі отриманих даних контролер виконує послідовність дій, що забезпечують автоматичний полив методом затоплення та зливу.

Освітлення вмикається або вимикається безпосередньо згідно з командою сервера. Для керування насосом і дренажем ESP32 додатково виконує вимірювання поточного рівня води за допомогою ультразвукового датчика.

Обробка рівня води та керування насосом

Після отримання даних від датчика рівня води контролер обчислює фактичний рівень заповнення резервуара. Якщо покази датчика виходять за допустимі межі або відсутні, система фіксує помилку, негайно вимикає насос і надсилає повідомлення про несправність на сервер.

За умови коректної роботи датчика насос вмикається лише у випадку, якщо сервер дозволив подачу води та фактичний рівень води є нижчим за задане цільове значення. Після досягнення необхідного рівня насос автоматично вимикається.

Логіка зливу води

Злив води активується лише за умови, що сервер дозволив виконання цієї операції та рівень води перевищує мінімальний поріг. Коли рівень води

зменшується до мінімального значення, дренаж вимикається автоматично, що запобігає роботі системи «всуху».

Поведінка системи у разі збоїв

У разі втрати зв’язку з сервером, отримання некоректних відповідей або помилок сенсорів ESP32 переходить у захищений режим, вимикаючи всі виконавчі механізми. Такий підхід дозволяє уникнути аварійних ситуацій, зокрема переливу води або неконтрольованої роботи обладнання.

2.4.3 Проєктування веб-інтерфейсу системи керування мікрофермою

Веб-інтерфейс системи автоматизованого керування параметрами середовища мікроферми є основним інструментом взаємодії користувача з системою. Його структура та функціональність спроєктовані таким чином, щоб забезпечити зручне керування процесом вирощування, доступ до аналітичних даних, а також контроль стану системи в різних режимах роботи.

Інтерфейс поділено на кілька логічних сторінок, кожна з яких відповідає за окремий аспект роботи системи: планування, керування, аналітику та діагностику. Структура сторінки Plans зображена на рисунку 2.4.

Рисунок 2.4 Структура сторінки Plans

Сторінка Plans

Сторінка Plans призначена для управління шаблонами вирощування, які визначають автоматичну поведінку ферми у режимі scheduled. План є універсальним описом циклу вирощування та може використовуватись повторно для запуску нових циклів.

На цій сторінці користувач має можливість:

- створювати нові плани вирощування з визначеними параметрами поливу, освітлення та тривалості циклу;
- переглядати перелік наявних планів;
- редагувати параметри існуючих планів;
- видаляти плани, які більше не використовуються.

Таким чином, сторінка Plans реалізує рівень планування, відокремлений від безпосереднього керування фермою.

Структура сторінки Control зображена на рисунку 2.5.

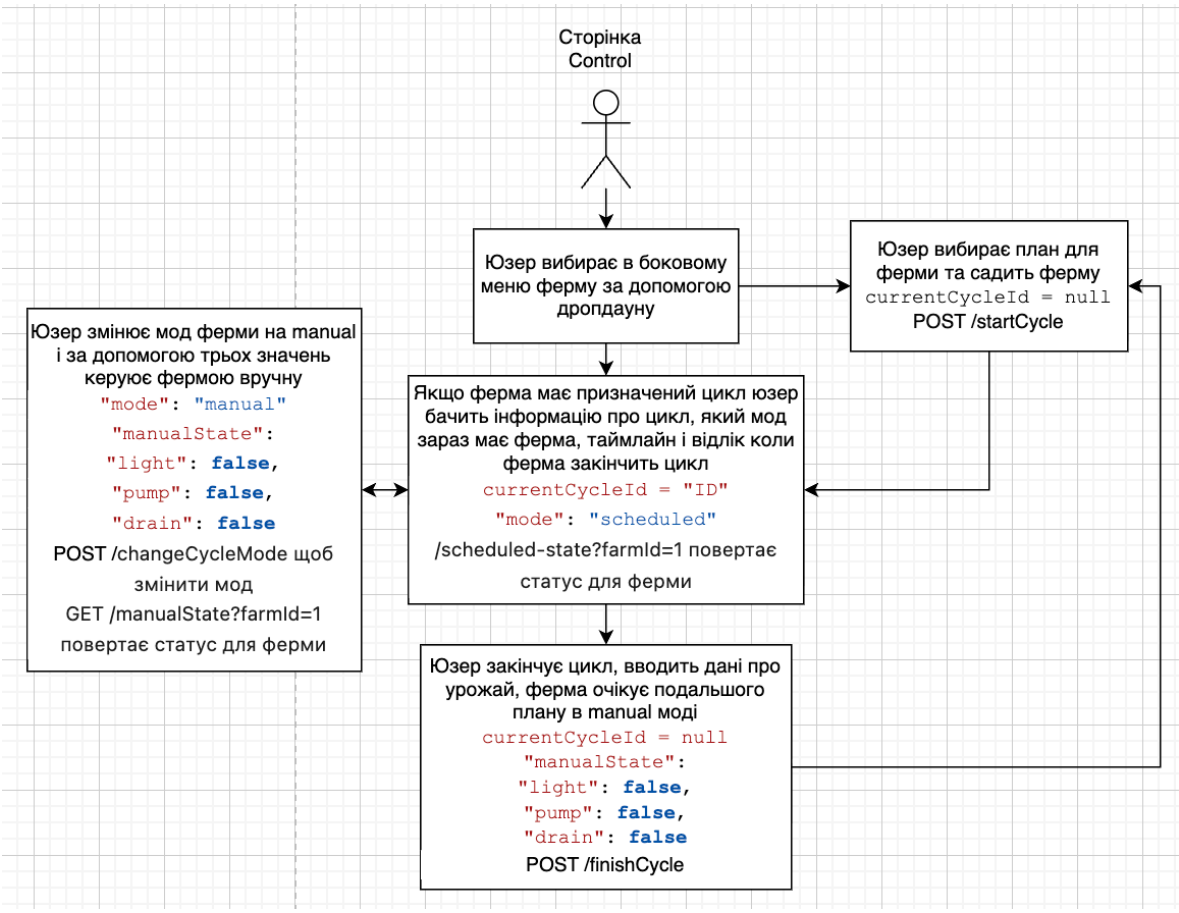

Рисунок 2.5 Структура сторінки Control

Сторінка Control

Сторінка Control є центральною сторінкою оперативного керування фермою. Вона дозволяє користувачеві взаємодіяти з конкретною фермою та контролювати її поточний стан.

Після вибору ферми з бокового меню користувач бачить актуальну інформацію про режим роботи ферми та, залежно від наявності активного циклу, отримує доступ до різних сценаріїв керування.

Ручний режим керування

Якщо ферма не має активного циклу вирощування (currentCycleId = null), вона перебуває у ручному режимі. У цьому стані користувач може:

- перемикати режим роботи ферми у manual;
- вручну керувати модулями освітлення, поливу та зливу;
- підготувати ферму до запуску автоматичного циклу.

Поточний стан модулів зберігається на сервері та використовується для формування команд керування.

Запуск і виконання циклу

Користувач може обрати один із раніше створених планів та ініціювати запуск циклу вирощування. Після запуску:

- фермі присвоюється ідентифікатор активного циклу;
- режим роботи автоматично перемикається у scheduled;
- система починає виконувати дії згідно зі згенерованим розкладом.

У цьому режимі користувач бачить:

- інформацію про активний цикл;
- поточний режим роботи;
- таймлайн виконання циклу;
- прогнозований час завершення.

Завершення циклу

Після завершення циклу користувач може:

- зафіксувати інформацію про зібраний урожай;
- завершити цикл вирощування;
- перевести ферму назад у ручний режим.

Після цього активний цикл скидається, і ферма переходить у стан очікування нового плану.

Структура сторінок Reports, Cycles та Logs зображена на рисунку 2.6.

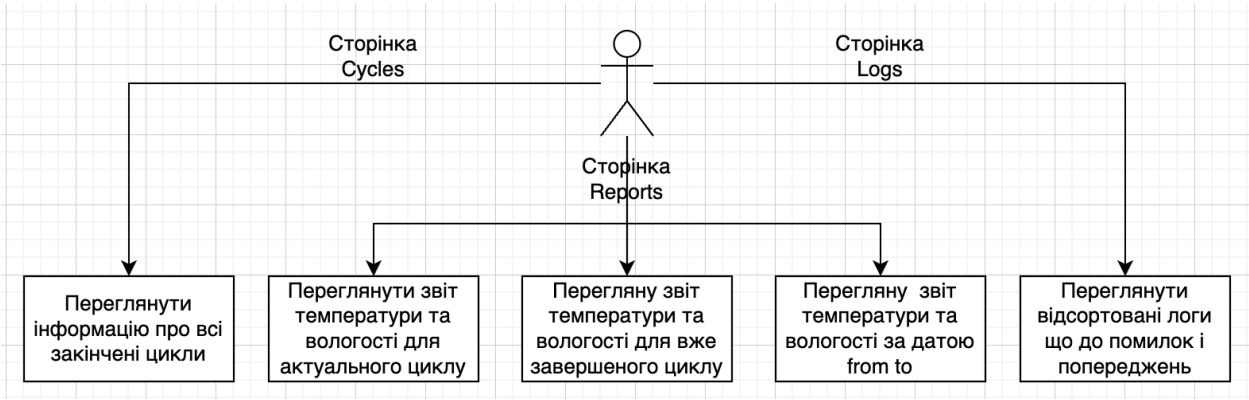

Рисунок 2.6 Структура сторінок Reports, Cycles та Logs

Сторінки Reports, Cycles та Logs

Аналітична та діагностична частини веб-інтерфейсу реалізовані за допомогою сторінок Reports, Cycles та Logs, які забезпечують доступ до історичних даних і журналів роботи системи.

Сторінка Cycles

Сторінка Cycles дозволяє переглядати перелік завершених циклів вирощування та слугує точкою входу для аналізу результатів конкретних періодів роботи ферми.

Сторінка Reports

Сторінка Reports надає користувачеві можливість:

- переглядати звіти температури та вологості для поточного циклу;
- аналізувати дані завершених циклів;
- формувати звіти за довільний проміжок часу (from–to).

Це дозволяє оцінювати стабільність параметрів середовища та ефективність різних планів вирощування.

Сторінка Logs

Сторінка Logs призначена для перегляду системних журналів, які містять повідомлення про помилки та попередження. Вона забезпечує контроль коректності роботи системи та спрощує виявлення й аналіз несправностей.

2.5 Проміжні висновки

У другому розділі магістерської роботи було розглянуто алгоритми та моделі рішень, що лежать в основі системи автоматизованого керування параметрами середовища мікроферми.

У підрозділі, присвяченому архітектурі системи, було обґрунтовано доцільність використання розподіленої апаратно-програмної моделі, у якій серверна частина відповідає за прийняття рішень і планування, а мікроконтролер ESP32 - за виконання команд та збір сенсорних даних. Такий підхід забезпечує гнучкість, масштабованість і можливість подальшого розширення системи без істотних змін апаратної частини.

Аналіз необхідних даних дозволив структурувати інформаційні потоки системи та розділити їх на три основні групи: сенсорні дані, керуючі параметри, що генеруються системою, та дані планування, які задаються користувачем. Це створило основу для реалізації автоматизованої логіки керування та забезпечило чітке розмежування відповідальності між компонентами системи.

У підрозділі вибору технологій було обґрунтовано використання сучасного стеку апаратних і програмних засобів, який відповідає вимогам енергоефективності, простоти інтеграції та підтримки. Застосування ESP32, Node.js, MongoDB та сучасного веб-фреймворку дозволяє створити систему, орієнтовану на малі фермерські господарства, із можливістю подальшого масштабування.

Проєктування системи у вигляді алгоритмів та діаграм дало змогу формалізувати поведінку апаратної та програмної частин у різних режимах

роботи. Особливу увагу було приділено логіці поливу методом затоплення та зливу, яка забезпечує автоматичне досягнення заданого рівня води та захист від аварійних ситуацій. Окремо було розглянуто структуру веб-інтерфейсу, що дозволяє користувачеві планувати цикли, керувати фермою, аналізувати звіти та контролювати стан системи.

Отримані результати підтверджують, що запропонована архітектура та алгоритмічні рішення є придатними для реалізації системи автоматизованого керування параметрами середовища мікроферми. Сформовані в цьому розділі моделі та підходи створюють основу для практичної реалізації системи, її тестування та подальшої оцінки ефективності, які будуть розглянуті в наступних розділах роботи.

РОЗДІЛ 3 ПОБУДОВА ВЛАСНОГО РІШЕННЯ

3.1 Розробка програмної частини

Програмна частина системи автоматизованого керування параметрами середовища мікроферми реалізує основну логіку роботи системи та забезпечує взаємодію між апаратним контролером і користувачем. Вона включає серверну та клієнтську складові, які відповідають за обробку даних, формування керуючих команд, збереження інформації та відображення стану системи через веб-інтерфейс.

У межах даного підрозділу розглядається процес розробки серверної частини системи, що реалізує програмні інтерфейси для обміну даними з мікроконтролером, а також клієнтської частини, яка забезпечує зручну взаємодію користувача з системою. Особливу увагу приділено структурі програмних компонентів та принципам організації програмного коду.

3.1.1 Розробка серверної (back-end) частини. Створеннння та тестування кінцевих точок (Postman)

Серверна частина системи автоматизованого керування параметрами середовища мікроферми реалізована у вигляді модульного застосунку на платформі Node.js з використанням фреймворка Express, який забезпечує створення REST-орієнтованого програмного інтерфейсу. Для роботи з базою даних MongoDB використовується бібліотека Mongoose, що дозволяє описувати структуру даних у вигляді схем і спрощує виконання операцій з колекціями.

Back-end частина виконує роль центрального компонента системи, який:

- приймає телеметричні дані від мікроконтролера ESP32;
- формує керуючі команди залежно від режиму роботи ферми;
- зберігає історичні дані вимірювань та подій;
- надає API для веб-інтерфейсу користувача;

- забезпечує тестування системи через інструмент Postman.

Структура серверного проєкту

Серверний застосунок має наступну логічну структуру:

```
/app
 ├─ index.mjs              – ініціалізація сервера та реєстрація ендпоінтів
 ├─ /database
 │   └─ Mongo.database.js  – підключення до MongoDB через Mongoose
 ├─ /mailer
 │   └─ mailer.js          – надсилання сповіщень на пошту
 ├─ /models
 │   └─ models.js          – опис схем даних MongoDB
 ├─ /routes
 │   └─ routes.js          – реалізація REST API
 └─ .env                   – конфігураційний файл (PORT, CONNECTION_STRING)
```

Файл index.mjs відповідає за запуск Express-сервера, підключення до бази даних та реєстрацію всіх маршрутів. У ньому реалізується базова конфігурація серверного застосунку та ініціалізація основних сервісів.

Підключення до бази даних винесене в окремий модуль, що підвищує модульність проєкту та спрощує його подальше розширення або зміну джерела зберігання даних.

Опис моделей даних серверної частини

Для зберігання та обробки інформації в серверній частині системи використовується нереляційна база даних MongoDB. Структура даних описана за допомогою схем Mongoose, що дозволяє формалізувати модель предметної області, забезпечити цілісність даних та спростити взаємодію між серверною логікою і базою даних.

У системі реалізовано п’ять основних схем, кожна з яких відповідає за окремий аспект функціонування мікроферми.

1. Схема плану вирощування (PlanSchema)

Схема PlanSchema призначена для зберігання шаблонів планів вирощування, які використовуються для формування автоматичних циклів роботи ферми. План описує узагальнену логіку вирощування без прив’язки до конкретних календарних дат.

Основні поля схеми:

- name — назва плану вирощування;
- description — текстовий опис плану;
- hoursNumber — загальна тривалість циклу пророщування у годинах;
- plant — тип рослини або зерна;
- plantWeight — маса посадкового матеріалу.

Для керування середовищем у межах плану використовуються вкладені структури:

- lightPlan — масив об’єктів, що описує розклад освітлення. Кожен елемент містить:
  - startTime — час початку освітлення відносно старту циклу;
  - duration — тривалість освітлення;
  - intensity — рівень яскравості освітлення.
- floadingPlan — масив об’єктів, що визначає графік поливу методом затоплення. Кожен елемент містить:
  - startTime — час початку поливу;
  - holdTime — тривалість утримання води;
  - waterLevel — цільовий рівень води.

Таким чином, PlanSchema реалізує концепцію повторно використовуваних шаблонів, які можуть застосовуватися для різних запусків ферми.

2. Схема звітів вимірювань (ReportSchema)

Схема ReportSchema використовується для збереження історії сенсорних вимірювань, які надходять від мікроконтролера ESP32 під час роботи ферми.

Основні поля:

- _id — унікальний ідентифікатор вимірювання;
- cycleId — ідентифікатор циклу, в межах якого було отримано дані;
- temperature — виміряне значення температури;
- humidity — виміряне значення вологості;
- receivedAt — час отримання даних.

Дані цієї колекції використовуються для побудови графіків, формування звітів та аналізу стабільності параметрів середовища.

3. Схема циклу вирощування (CycleSchema)

Схема CycleSchema описує конкретний цикл роботи ферми, який створюється на основі вибраного плану вирощування.

Ключові поля схеми:

- _id — унікальний ідентифікатор циклу;
- planId — посилання на план, за яким створено цикл;
- isStarted, isFinished — прапорці стану циклу;
- result — результат циклу (наприклад, кількість або маса врожаю);
- resultComment — коментар користувача щодо результату;
- mode — режим роботи ферми (manual або scheduled).

Для реалізації автоматичного керування цикл містить три окремі розклади:

- lightSchedule — масив подій керування освітленням;
- pumpSchedule — масив подій подачі води із заданим рівнем;
- drainSchedule — масив подій зливу води.

Кожна подія має часову мітку performAt, що дозволяє серверу у реальному часі формувати керуючі команди для апаратної частини.

4. Схема ферми (FarmSchema)

Схема FarmSchema зберігає загальну інформацію про конкретну ферму та її поточний стан.

Основні поля:

- farmId — унікальний ідентифікатор ферми;
- name — назва ферми;
- description — опис ферми;
- currentCycleId — ідентифікатор активного циклу або null;
- manualState — стан модулів у ручному режимі, що включає параметри:
  - light — стан освітлення;
  - pump — стан насоса;
  - drain — стан дренажу.

Ця схема використовується сервером для формування відповідей у ручному режимі роботи.

5. Схема журналу подій (LogSchema)

Схема LogSchema призначена для фіксації системних подій, помилок та попереджень.

Поля схеми:

- _id — унікальний ідентифікатор події;
- timestamp — час виникнення події;
- type — тип повідомлення (error, warning тощо);
- farmId, cycleId — ідентифікатори пов’язаних об’єктів;
- event — опис події;
- value — додаткове значення або параметр;
- emailSent — прапорець, що вказує на факт надсилання сповіщення.

Логи використовуються для діагностики, тестування нештатних ситуацій та аналізу стабільності роботи системи.

Запропонована модель даних охоплює всі ключові аспекти функціонування системи - від планування та керування циклами до збереження вимірювань і фіксації помилок. Такий підхід забезпечує цілісність даних, гнучкість у керуванні фермою та можливість подальшого розширення функціональності серверної частини.

REST API та логіка керування системою

Серверна частина системи взаємодіє з мікроконтролером мікроферми та веб-інтерфейсом користувача через REST API, що забезпечує уніфікований і розширюваний обмін даними між компонентами системи. Усі запити та відповіді передаються у форматі JSON, що спрощує інтеграцію з апаратною частиною та клієнтським застосунком.

Для розробки, тестування та перевірки коректності роботи кінцевих точок використовується інструмент Postman, у якому створено окрему колекцію запитів. Для зручності тестування застосовується змінна середовища {{baseURL}}, що дозволяє швидко змінювати адресу сервера без необхідності редагування кожного запиту вручну. Це спрощує налагодження системи під час локальної розробки та після розгортання серверної частини у хмарному середовищі.

№	Ендпоінт	HTTP-метод	Призначення
1	/startCycle	POST	Запуск нового циклу вирощування на основі обраного плану з автоматичною генерацією розкладів
2	/finishCycle	POST	Завершення поточного циклу вирощування та фіксація результатів
3	/cycle	GET	Отримання детальної інформації про конкретний цикл
4	/getFinishedCycles	GET	Отримання списку завершених циклів вирощування
5	/data	POST	Передача виміряних сенсорних даних (температура, вологість) від ферми
6	/mode	GET	Отримання поточного режиму роботи ферми (manual / scheduled)
7	/changeCycleMode	POST	Зміна режиму роботи циклу (перехід між

№	Ендпоінт	HTTP-метод	Призначення
			manual і scheduled)
8	/scheduled-state	GET	Отримання керуючих команд для ферми в автоматичному режимі
9	/manual	POST	Керування модулями ферми в ручному режимі
10	/manual-state	GET	Отримання поточних команд керування в ручному режимі
11	/plan	POST	Створення нового плану пророщування
12	/plan	GET	Отримання плану пророщування за ідентифікатором
13	/plans	GET	Отримання переліку всіх збережених планів
14	/plansName	GET	Отримання списку назв планів без детальної інформації
15	/plan	PUT	Оновлення існуючого плану пророщування
16	/plan	DELETE	Видалення плану пророщування
17	/farms	GET	Отримання списку всіх ферм
18	/farm	GET	Отримання детальної інформації про конкретну ферму
19	/getReport	GET	Отримання історії сенсорних даних для побудови графіків
20	/timeline	GET	Отримання агрегованої часової інформації про події циклу
21	/logs	GET	Отримання журналу подій, помилок і попереджень системи

Серед усіх кінцевих точок серверної частини особливе значення мають три ендпоінти, які забезпечують безпосередню взаємодію між сервером і апаратною частиною мікроферми, а також реалізують основну логіку керування процесом вирощування. До них належать /startCycle, /scheduled-state та /data.

Ендпоінт /startCycle використовується для запуску нового циклу вирощування на основі плану, обраного користувачем через веб-інтерфейс. Запит містить ідентифікатори ферми та плану, які необхідні для формування нового циклу.

Після отримання запиту сервер виконує такі послідовні дії:

- перевіряє наявність ферми та плану у базі даних;
- на основі параметрів плану генерує розклади освітлення, подачі води та зливу;
- створює новий документ циклу у базі даних;
- зберігає ідентифікатор активного циклу у документі ферми.

Таким чином, план вирощування, що не містить конкретних календарних дат, перетворюється на цикл із фіксованими часовими мітками для кожної керуючої події.

Під час запуску нового циклу вирощування у межах ендпоінту POST /startCycle для формування розкладу освітлення використовується допоміжна функція generateLightSchedule. Її основним призначенням є перетворення узагальненого опису освітлення, заданого у плані вирощування, у послідовність конкретних подій із фіксованими часовими мітками.

Функція приймає як вхідні параметри загальну тривалість циклу вирощування у годинах та масив параметрів освітлення, визначених у плані. Початок циклу встановлюється з невеликим зсувом у часі, що дозволяє уникнути миттєвого старту виконання подій одразу після запуску циклу.

Для кожного елементу плану освітлення функція обчислює:

- момент увімкнення освітлення;
- момент вимкнення освітлення з урахуванням заданої тривалості.

Отримані події додаються до загального розкладу у вигляді впорядкованого масиву, де кожна подія містить булевий стан освітлення та часову мітку виконання. Додатково виконується фільтрація подій, що виходять за межі тривалості циклу, а також сортування за часом виконання.

Застосування окремої допоміжної функції для генерації розкладу освітлення дозволяє:

- відокремити логіку планування від обробки HTTP-запитів;
- забезпечити повторне використання алгоритму;

- спростити підтримку та розширення серверної частини системи.

Структура тіла запиту POST /startCycle наведена на рисунку 3.1, а приклад відповіді сервера у разі успішного запуску циклу - на рисунку 3.2.

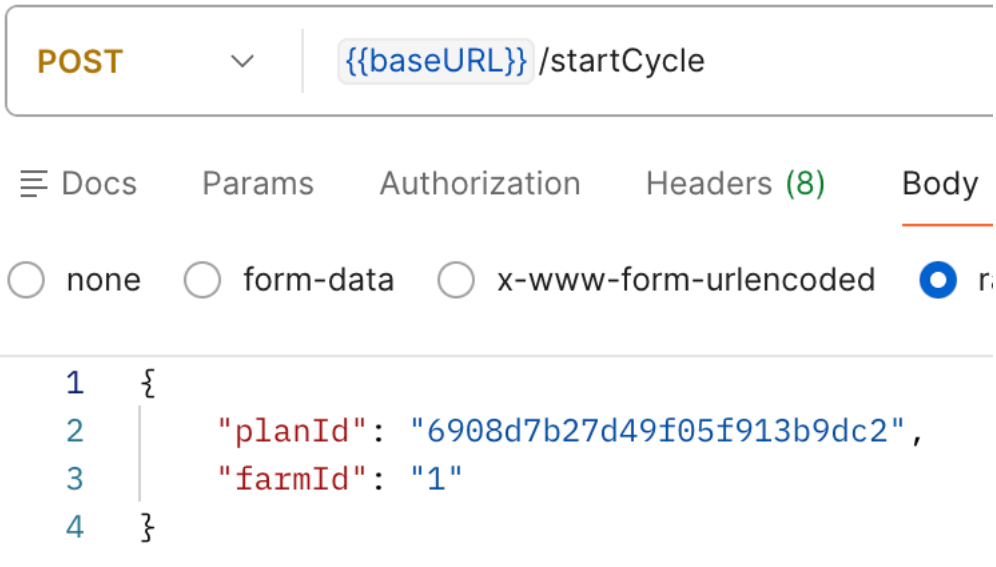

Рисунок 3.1 Структура тіла запиту /startCycle

Відповідь у випадку успіху буде виглядати наступним чином:

```
{
    "manualState": {
        "light": false,
        "pump": false,
        "drain": false
    },
    "_id": "6907a2b1c5895efa49c3bdbb",
    "farmId": "1",
    "description": "qweqweqwe",
    "currentCycleId": "692748b0bff79fb8f830abb0",
    "name": "Base Farm"
}
```

Рисунок 3.2 Структура відповіді для /startCycle

Ці приклади отримані під час тестування ендпоінту в середовищі Postman.

GET /scheduled-state — отримання команд у автоматичному режимі

Ендпоінт /scheduled-state використовується мікроконтролером ESP32 у режимі автоматичної роботи (scheduled). Ферма виконує цей запит періодично з фіксованим інтервалом для отримання актуальних команд керування.

Сервер формує відповідь на основі:

- активного циклу ферми;
- згенерованих розкладів освітлення, подачі води та зливу;
- поточного часу.

Для кожного модуля визначається його актуальний стан шляхом вибору останньої події, час виконання якої не перевищує поточний момент. У відповідь повертаються булеві значення для керування модулями, а також цільовий рівень води, необхідний для реалізації поливу методом затоплення.

У випадку, якщо цикл переведений у ручний режим, сервер повертає відповідне повідомлення, що сигналізує апаратній частині про зміну логіки роботи.

Приклад URL-запиту та структури відповіді для автоматичного режиму наведені на рисунку 3.3 та рисунку 3.4.

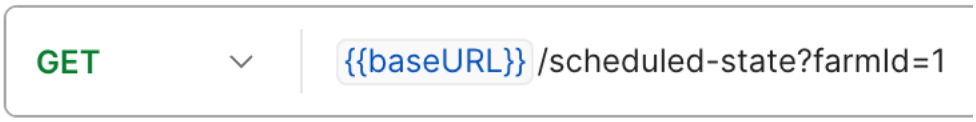

Рисунок 3.3 Приклад URL запиту для /scheduled-state

Відповідь у випадку успіху буде виглядати наступним чином:

Рисунок 3.4 Структура відповіді для /scheduled-state

Приклад відповіді у випадку ручного керування показано на рисунку 3.5.

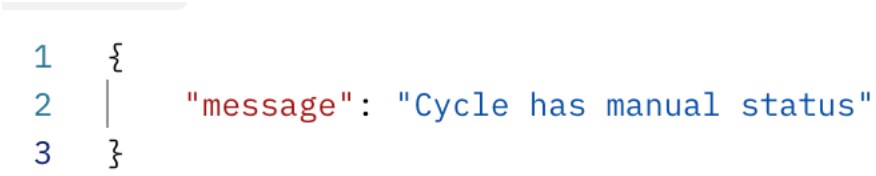

Рисунок 3.5 Структура відповіді для /scheduled-state якщо ферма має manual керування

POST /data — обробка сенсорних даних та моніторинг нештатних ситуацій

Ендпоінт /data використовується мікроконтролером для періодичної передачі виміряних параметрів мікроклімату, зокрема температури та вологості повітря. Запит містить поточні значення, отримані з датчиків, встановлених на мікрофермі.

На першому етапі сервер ідентифікує ферму за переданим параметром farmId та визначає активний цикл вирощування. Отримані значення температури та вологості зберігаються у базі даних у вигляді окремого запису, прив’язаного до поточного циклу або збереженого без прив’язки у випадку його відсутності.

Цей підхід дозволяє накопичувати повну історію вимірювань, яка використовується для побудови графіків, формування звітів та аналізу стабільності умов середовища.

Аналіз даних та класифікація подій

Після збереження сенсорних даних сервер виконує перевірку отриманих значень з метою виявлення нештатних ситуацій. Залежно від характеру відхилень дані класифікуються як:

- критична помилка (error) — у випадку отримання некоректних значень, що свідчать про несправність сенсора;
- попередження (warning) — у випадку виходу температури або вологості за рекомендовані межі.

Для кожної події формується уніфікований опис, що включає тип події, її ідентифікатор та текстове повідомлення.

Формування журналу подій

У разі виявлення помилки або попередження сервер створює відповідний запис у журналі подій. Лог містить інформацію про тип події, ідентифікатор ферми та циклу, часову мітку, а також значення параметрів, які спричинили виникнення події.

Журнал подій використовується для діагностики системи, аналізу стабільності роботи та відображення повідомлень у веб-інтерфейсі користувача.

Механізм надсилання сповіщень

Для критичних подій та попереджень передбачено механізм автоматичного надсилання повідомлень електронною поштою. З метою уникнення надмірної кількості сповіщень реалізовано обмеження частоти надсилання повідомлень одного типу для конкретної ферми.

Перед надсиланням нового повідомлення сервер перевіряє, чи надсилалося аналогічне сповіщення протягом визначеного проміжку часу. У разі відсутності нещодавніх повідомлень формується та надсилається електронний лист, а відповідний запис у журналі подій позначається як такий, що спричинив відправлення сповіщення.

Результат виконання запиту

Незалежно від наявності попереджень або помилок, у разі успішної обробки даних сервер повертає відповідь зі статусом ok. Це дозволяє апаратній частині системи продовжувати роботу без затримок та повторних спроб передачі даних.

Накопичені дані використовуються для формування звітів, побудови графіків температури та вологості, а також для аналізу умов вирощування в межах окремих циклів.

Структура тіла запиту та приклад відповіді ендпоінту POST /data наведені на рисунку 3.6, отриманому під час тестування в Postman.

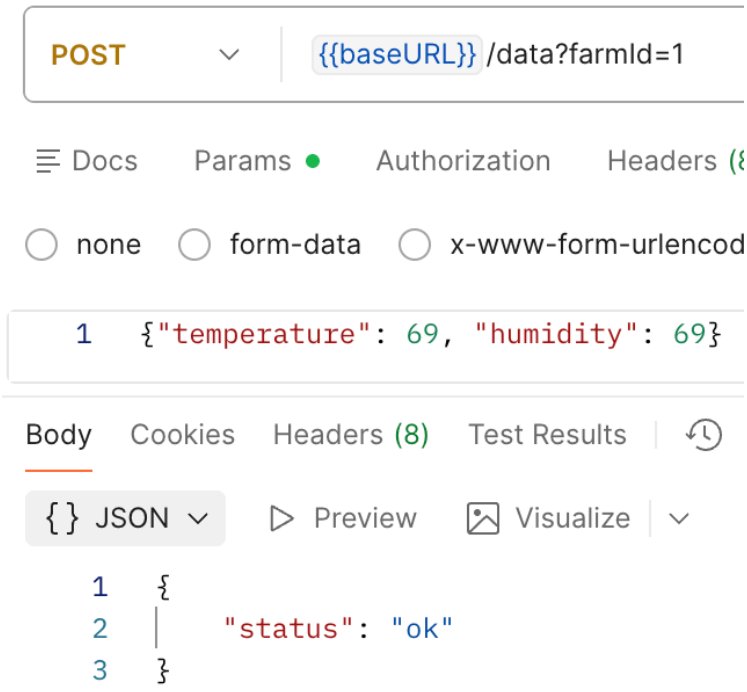

Рисунок 3.6 Структура тіла запиту та відповідь ендпоінту /data

Таким чином, серверна частина системи реалізує повний цикл обробки даних та керування мікрофермою: від прийому сенсорної інформації та її збереження до формування керуючих команд і сповіщення користувача про нештатні ситуації. Запропонована структура REST API, модульна організація коду та використання інструментів тестування дозволяють забезпечити стабільну взаємодію між апаратною частиною, веб-інтерфейсом і базою

даних, створюючи основу для подальшого розвитку та масштабування системи.

3.1.2 Розробка front-end частини

Клієнтська (front-end) частина системи призначена для забезпечення зручної та наочної взаємодії користувача з системою автоматизованого керування параметрами середовища мікроферми. Вона реалізує доступ до основних функцій керування, відображення поточного стану ферми, перегляд аналітичних даних та роботу з планами й циклами вирощування.

У межах даного підрозділу розглядається реалізація веб-інтерфейсу системи, його структура, основні сторінки та принципи взаємодії з серверною частиною через REST API. Особливу увагу приділено зручності користування, адаптивності інтерфейсу та логічному поділу функціональності між сторінками.

Для розробки клієнтської частини системи було обрано фреймворк Next.js версії 16.0.7, який надає сучасні засоби побудови веб-інтерфейсів, підтримує компонентний підхід та забезпечує ефективну взаємодію з серверною частиною через REST API. Обрана версія орієнтована на використання App Router як основного механізму маршрутизації, що відповідає актуальним рекомендаціям розробників Next.js.

App Router, який базується на директорії app та забезпечує більш структурований і гнучкий підхід до побудови веб-застосунків. Маршрути формуються на основі ієрархії папок, де файл page.tsx відповідає за відображення сторінки, а файл layout.tsx — за спільний макет для групи сторінок.

Типова структура застосунку з використанням App Router має вигляд:

app/

├── favicon.ico

├── globals.css

├── layout.tsx

├── page.tsx

Використання App Router дозволяє реалізувати вкладені маршрути, спільні макети та логічно відокремити функціональні частини інтерфейсу. У даному проєкті цей підхід використовується для організації сторінок керування фермою, роботи з планами пророщування, перегляду звітів та журналу подій.

Застосування App Router у поєднанні з компонентним підходом спрощує масштабування веб-інтерфейсу та підвищує зручність його підтримки у процесі подальшого розвитку системи.

Структура директорії app

- app/layout.tsx

Визначає спільний макет застосунку, який використовується для всіх сторінок. У макеті розміщено бокове меню навігації та загальні елементи інтерфейсу, що дозволяє забезпечити єдиний стиль та послідовну навігацію між розділами.

- app/page.tsx

Головна сторінка застосунку, яка слугує точкою входу до системи та перенаправляє користувача до основних функціональних розділів.

- app/control/page.tsx

Сторінка керування фермою, на якій користувач може:

  - переглядати поточний стан ферми;
  - перемикати режим роботи (manual / scheduled);
  - керувати модулями у ручному режимі;
  - запускати або завершувати цикли вирощування.

Дана сторінка є центральним елементом взаємодії користувача з системою.

- app/plans/page.tsx

Реалізує роботу з планами пророщування: створення нових планів, перегляд існуючих, редагування та видалення. Логіка цієї сторінки

безпосередньо пов’язана з підрозділом планування циклів на серверній частині.

- app/cycles/page.tsx

Сторінка перегляду інформації про завершені цикли вирощування, що дозволяє аналізувати результати попередніх запусків ферми.

- app/reports/page.tsx

Призначена для відображення звітів та аналітичних даних, зокрема історії температури та вологості. Дані отримуються з серверної частини та використовуються для побудови графіків і часових залежностей.

- app/logs/page.tsx

Реалізує перегляд журналу подій системи, включно з помилками та попередженнями. Для відображення таблиці логів використовуються допоміжні компоненти, що дозволяють сортування та фільтрацію даних.

Компонентна структура

Для повторного використання інтерфейсних елементів використовується директорія components, у якій зосереджено незалежні UI-компоненти:

- app-sidebar.tsx — бокове меню навігації;
- manual-controls.tsx — елементи керування модулями у ручному режимі;
- plan-selector.tsx — компонент вибору плану для запуску циклу;
- finish-cycle-modal.tsx — модальне вікно завершення циклу;
- timeline.tsx — візуалізація часової послідовності подій циклу;
- data-table.tsx, columns.tsx — компоненти табличного відображення даних.

Для побудови інтерфейсу активно використовується бібліотека shadcn/ui, яка надає готові стилізовані компоненти та сприяє збереженню єдиного візуального стилю застосунку.

Допоміжні каталоги

- hooks — містить кастомні React-хуки для роботи зі станом та серверними запитами;
- lib — допоміжні функції та сервіси, зокрема конфігурація HTTP-клієнта для взаємодії з бекендом;
- public — статичні ресурси;
- globals.css — глобальні стилі застосунку.

Використання бібліотеки shadcn/ui

Для побудови користувацького інтерфейсу у фронтенд-частині системи використовується бібліотека shadcn/ui, яка надає набір готових, стилістично узгоджених React-компонентів. Бібліотека базується на Tailwind CSS та дозволяє швидко створювати адаптивні й уніфіковані елементи інтерфейсу без необхідності розробки власних стилів з нуля.

Однією з переваг shadcn/ui є те, що компоненти інтегруються безпосередньо у код проєкту, що забезпечує повний контроль над їхньою поведінкою та стилізацією. Це особливо важливо для систем керування, де інтерфейс має бути простим, зрозумілим і легко розширюваним.

У межах даного проєкту компоненти shadcn/ui використовуються для реалізації таблиць, кнопок, форм введення, випадаючих списків та інших елементів взаємодії з користувачем. Наприклад, на сторінці перегляду журналу подій застосовано табличний компонент для відображення логів із можливістю фільтрації та посторінкової навігації.

Для прикладу у файлі data-table.tsx використано такі компоненти бібліотеки:

- Table, TableRow, TableCell — для відображення табличних даних;
- Select та Input — для реалізації фільтрів за типом події та ідентифікатором циклу;
- Button — для керування посторінковою навігацією.

Компоненти об’єднані у власний універсальний компонент таблиці, який приймає дані та конфігурацію через параметри. Такий підхід дозволяє повторно використовувати компонент для різних сторінок та спрощує підтримку коду.

Взаємодія клієнтської частини з сервером

Взаємодія фронтенд-частини системи з серверною реалізована за допомогою HTTP-запитів до REST API. Для виконання запитів використовується бібліотека axios, яка спрощує роботу з асинхронними запитами та обробку відповідей сервера. Адреса серверної частини задається через змінну середовища, що дозволяє легко змінювати конфігурацію без модифікації коду.

Обмін даними відбувається у форматі JSON, а отримані з сервера дані використовуються для динамічного оновлення стану користувацького інтерфейсу без перезавантаження сторінки.

Як приклад взаємодії з сервером розглянемо сторінку перегляду завершених циклів вирощування (cycles.tsx). Після завантаження сторінки клієнтська частина виконує запит до серверного ендпоінту для отримання списку звітів по завершених циклах.

Завантаження даних ініціюється у момент монтування компонента за допомогою хука useEffect, що дозволяє виконати асинхронний запит лише один раз при відкритті сторінки. Отримані дані зберігаються у локальному стані компонента.

Для керування станом інтерфейсу використовуються React-хуки:

- useState — для збереження отриманих з сервера даних;
- useEffect — для виконання побічних ефектів, зокрема HTTP-запитів.

```
  const [reports, setReports] = useState<CycleReport[]>([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchReports = async () => {
      try {
        const response = await
axios.get<CycleReport[]>(`${baseURL}/cycleReports`);
```

```
        setReports(response.data);
      } catch (err) {
        console.error(err);
      } finally {
        setLoading(false);
      }
    };

    fetchReports();
  }, []);
```

Під час завантаження даних використовується окремий стан loading, який дозволяє відобразити індикатор завантаження та запобігти відображенню неповних або некоректних даних.

Отримані з сервера дані зберігаються у структурованому вигляді та відображаються у табличній формі з використанням компонентів бібліотеки shadcn/ui.

```
<Card className="w-full">
      <CardTitle>Cycle Reports</CardTitle>
      <CardContent>
        <Table>
          <TableHeader>
            <TableRow>
              <TableCell>Plan</TableCell>
              <TableCell>Plant</TableCell>
              <TableCell>Plant Weight</TableCell>
              <TableCell>Result</TableCell>
              <TableCell>Comment</TableCell>
              <TableCell>Duration</TableCell>
            </TableRow>
          </TableHeader>
          <TableBody>
            {reports.map((report) => (
              <TableRow key={report._id}>
                <TableCell>{report.plan}</TableCell>
                <TableCell>{report.plant}</TableCell>
                <TableCell>{report.plantWeight}</TableCell>
                <TableCell>{report.result}</TableCell>
                <TableCell>{report.resultComment}</TableCell>
                <TableCell>{formatDuration(report.duration)}</TableCell>
              </TableRow>
            ))}
          </TableBody>
        </Table>
      </CardContent>
    </Card>
```

Додаткова логіка форматування (наприклад, перетворення тривалості циклу з хвилин у зручний для сприйняття формат) виконується безпосередньо на клієнтській стороні, що зменшує навантаження на сервер та підвищує гнучкість інтерфейсу.

Такий підхід дозволяє:

- відокремити логіку обробки даних від їх візуалізації;
- забезпечити швидке оновлення інтерфейсу;
- зберігати читабельність та структурованість коду.

Адаптивність користувацького інтерфейсу

Однією з важливих вимог до клієнтської частини системи є можливість зручного використання веб-інтерфейсу на різних пристроях, зокрема на смартфонах і планшетах. Для цього під час розробки інтерфейсу активно застосовуються можливості адаптивної верстки на основі Tailwind CSS.

Прикладом реалізації адаптивності є відображення списку планів пророщування на сторінці Plans, де кожен план представлений у вигляді елемента акордеону. Залежно від ширини екрана компонування елементів змінюється: на більших екранах інформація відображається у вигляді рядка з фіксованими колонками, тоді як на мобільних пристроях ті самі дані автоматично переходять у вертикальне розташування.

Використання адаптивних CSS-класів (зокрема md:flex-row, md:w-*, md:items-center) дозволяє без дублювання коду забезпечити коректне відображення інтерфейсу на різних роздільних здатностях. Детальна інформація про план приховується у згорнутому вигляді та відкривається лише за потреби, що покращує зручність користування та зменшує перевантаження інтерфейсу на малих екранах.

Таким чином, клієнтська частина системи забезпечує зручний доступ користувача до основних функцій керування мікрофермою, перегляду аналітичних даних та роботи з планами і циклами вирощування. Використання сучасного фреймворка Next.js, компонентного підходу та асинхронної взаємодії з серверною частиною дозволяє реалізувати адаптивний і наочний веб-інтерфейс.

3.2 Розробка апаратної частини

Апаратна частина системи реалізує функції зчитування параметрів середовища, виконання керуючих команд та взаємодії із серверною частиною

системи. У даній роботі апаратна складова розглядається не як завершена ферма, а як програмно-апаратний контролер, який виконує роль виконавчого вузла у системі автоматизованого керування параметрами мікроферми.

Як апаратну платформу було обрано мікроконтролер ESP32, який поєднує достатню обчислювальну потужність, вбудований Wi-Fi модуль та велику кількість GPIO-пінів. Це дозволяє одночасно працювати з сенсорами, виконавчими модулями та серверною частиною без використання додаткових модулів зв’язку.

Для реалізації функцій моніторингу використовуються:

- датчик температури та вологості DHT11;
- ультразвуковий датчик відстані HC-SR04P для визначення рівня води.

У демонстраційному макеті реальні виконавчі пристрої (насос, клапан, освітлення) замінено світлодіодами, що дозволяє наочно продемонструвати логіку керування без використання потужних електричних компонентів. Живлення системи здійснюється від джерела 5 В (USB або power bank).

Ініціалізація апаратних компонентів

Ініціалізація апаратної частини виконується у функції setup(). На цьому етапі налаштовується серійний інтерфейс для налагодження, ініціалізуються сенсори, встановлюється з’єднання з Wi-Fi мережею та задаються режими роботи GPIO-пінів.

Для керування виконавчими модулями відповідні піни переводяться у режим виходу та встановлюються у безпечний початковий стан:

```
pinMode(ledPin, OUTPUT);
pinMode(pumpPin, OUTPUT);
pinMode(drainPin, OUTPUT);

digitalWrite(ledPin, LOW);
digitalWrite(pumpPin, LOW);
digitalWrite(drainPin, LOW);
```

Такий підхід гарантує, що після запуску мікроконтролера жоден із модулів не буде активований до отримання відповідних команд від серверної частини.

Підключення до бездротової мережі Wi-Fi

Для забезпечення обміну даними з серверною частиною системи мікроконтролер ESP32 підключається до бездротової мережі Wi-Fi під час ініціалізації. Параметри мережі (SSID та пароль) задаються у вигляді констант, що дозволяє легко змінювати конфігурацію без модифікації логіки програми.

```
const char* ssid = "TP-Link_5BFC";
 const char* password = "178A-106";
```

Підключення до мережі виконується у функції setup() та супроводжується перевіркою статусу з'єднання. Мікроконтролер переходить до виконання основної логіки лише після успішного встановлення з'єднання з мережею, що гарантує готовність системи до обміну даними з сервером.

Фрагмент ініціалізації підключення до Wi-Fi:

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

Використання циклу очікування дозволяє уникнути ситуацій, коли система намагається виконувати HTTP-запити без активного мережевого з'єднання. У подальшій роботі статус Wi-Fi періодично перевіряється перед виконанням запитів до серверної частини, що підвищує стабільність роботи системи.

Зчитування параметрів середовища

Зчитування температури та вологості здійснюється з використанням датчика DHT11. У випадку, якщо сенсор повертає некоректні значення, використовуються резервні (fallback) значення, що дозволяє серверу виявити помилку сенсора.

```
#define DHTPIN 25
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
float h = dht.readHumidity();
      float t = dht.readTemperature();

      // Значення за замовчуванням, якщо сенсор не читається
      if (isnan(h) || isnan(t)) {
        Serial.println("Error reading DHT11! Sending fallback data...");

        h = 99;
        t = 99; }
```

Визначення рівня води виконується за допомогою ультразвукового датчика HC-SR04P шляхом вимірювання часу проходження ультразвукового імпульсу від передавача до відбивача та назад.

Передача сенсорних даних на сервер

Отримані значення температури та вологості передаються на серверну частину за допомогою HTTP POST-запиту з фіксованим інтервалом. Дані надсилаються у форматі JSON, що забезпечує простоту обробки на сервері та збереження у базі даних.

```
if (WiFi.status() == WL_CONNECTED) {
      HTTPClient http;
      http.begin(dataUrl);
      http.addHeader("Content-Type", "application/json");
String payload = "{\"temperature\":" + String(t) + ", \"humidity\":" + String(h) + "}";
      int code = http.POST(payload);

      Serial.print("Data POST code: ");
      Serial.println(code);

      http.end();
```

Отримання керуючих команд

ESP32 періодично виконує HTTP GET-запити до серверної частини для отримання поточного режиму роботи ферми та відповідних керуючих команд. Залежно від режиму (manual або scheduled) виконується різна логіка обробки відповіді.

Отримання поточного режиму ферми:

```
String getFarmMode() {
  if (WiFi.status() != WL_CONNECTED) return "";

  String url = String(serverBase) + "/mode?farmId=" + farmId;
  HTTPClient http;
  http.begin(url);

  int httpCode = http.GET();

  if (httpCode == 200) {
    String response = http.getString();
    response.trim();
    Serial.println("Mode: " + response);
    http.end();
    return response;
  } else {
    Serial.print("Mode request failed, code: ");
    Serial.println(httpCode);
    http.end();
    return "";
  }}
```

Усі рішення щодо увімкнення або вимкнення модулів приймаються сервером, тоді як мікроконтролер виконує роль виконавчого пристрою.

Логіка керування рівнем води та розрахунок realLevel

Для реалізації поливу методом затоплення необхідно контролювати фактичний рівень води у резервуарі. У системі використовується ультразвуковий датчик, який вимірює відстань від сенсора до поверхні води.

Під час ініціалізації задається параметр constantHeight, який відповідає відстані від датчика до дна резервуара у міліметрах. Значення може змінюватися залежно від способу монтажу датчика.

Рівень води обчислюється за формулою:

$$realLevel = constantHeight - distance$$

де:

- distance — виміряна відстань від датчика до поверхні води;
- realLevel — фактичний рівень води у резервуарі.

Фрагмент реалізації обчислення:

```
duration = pulseIn(echoPin, HIGH, 20000);
distance = duration / 5.82;
int realLevel = constantHeight - distance;
```

Отримане значення realLevel порівнюється з цільовим рівнем води, який надходить із серверної частини. Якщо рівень нижчий за необхідний — вмикається насос, якщо рівень досягнуто — насос вимикається. Аналогічно реалізовано керування зливом води.

Такий підхід дозволяє адаптувати систему до різних фізичних конфігурацій резервуара без зміни основної логіки керування.

Обробка помилок та безпечний режим

У разі виявлення некоректних показників датчика рівня води або втрати зв’язку з сервером система автоматично переводить виконавчі модулі у безпечний стан. Додатково формується повідомлення про помилку якщо це можливо, яке передається на сервер для журналювання та сповіщення користувача.

Апаратна частина системи реалізує функції збору даних та виконання керуючих команд, залишаючи всю логіку прийняття рішень на серверній стороні. Такий розподіл відповідальності забезпечує гнучкість, масштабованість та спрощує подальший розвиток системи автоматизованого керування параметрами середовища мікроферми.

3.3 Інеграція та тестування

Після реалізації програмної та апаратної частин системи наступним етапом є їх інтеграція та перевірка коректності спільної роботи. На цьому етапі особлива увага приділяється взаємодії між мікроконтролером, серверною частиною та веб-інтерфейсом, а також стабільності передачі даних і виконання керуючих команд.

У межах даного розділу розглядаються результати тестування основних сценаріїв використання системи, аналіз отриманих даних, їх візуалізація та перевірка роботи у нештатних ситуаціях.

3.3.1 Тестування взаємодії користувача з системою

Початкове налаштування апаратної частини

Перед запуском системи користувач виконує початкову конфігурацію мікроконтролера ESP32 у середовищі розробки Arduino IDE. На цьому етапі в коді задаються параметри підключення до бездротової мережі Wi-Fi:

```
const char* ssid = "TP-Link_5BFC";
const char* password = "178A-106";
```

Також, за потреби, коригується значення параметра constantHeight, яке визначає відстань від ультразвукового датчика до дна резервуара та використовується для обчислення рівня води:

```
long constantHeight = 65;
```

У випадку використання декількох ферм користувач додатково задає ідентифікатор ферми, який використовується для зв’язку з серверною частиною системи:

```
const char* farmId = "1";
```

Після внесення необхідних параметрів код завантажується на ESP32. У вікні Serial Monitor перевіряється успішність підключення до Wi-Fi мережі, що підтверджується відповідним повідомленням у логах(Рис 3.6).

```
Connecting to WiFi.......
Connected!
```

Рисунок 3.6 Повідомлення про успішність підключення до Wi-Fi мережі

Запуск та перевірка базової взаємодії

Після успішного підключення мікроконтролера до мережі користувач переходить до веб-інтерфейсу системи. Через сайт виконується створення плану пророщування, перевіряється можливість його редагування та видалення. Вигляд сторінки Plans зображено на рисунку 3.7.

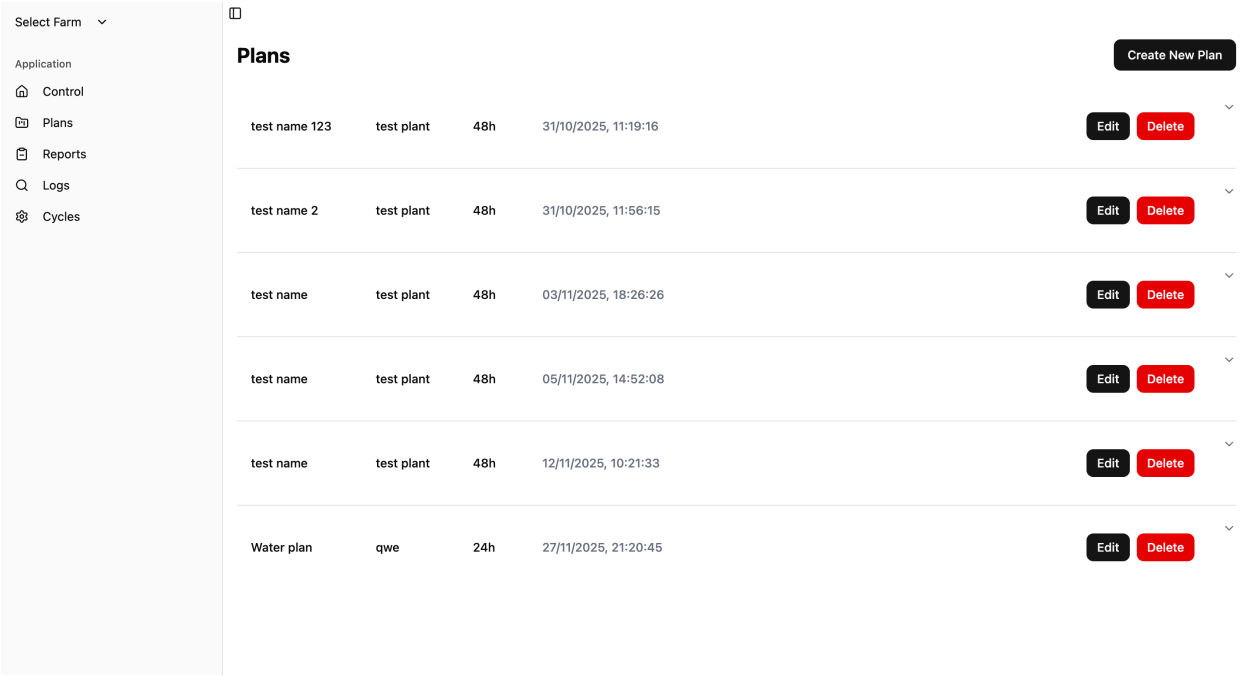

Рисунок 3.7 Сторінка Plans

Створення плану зображено на рисунку 3.8.

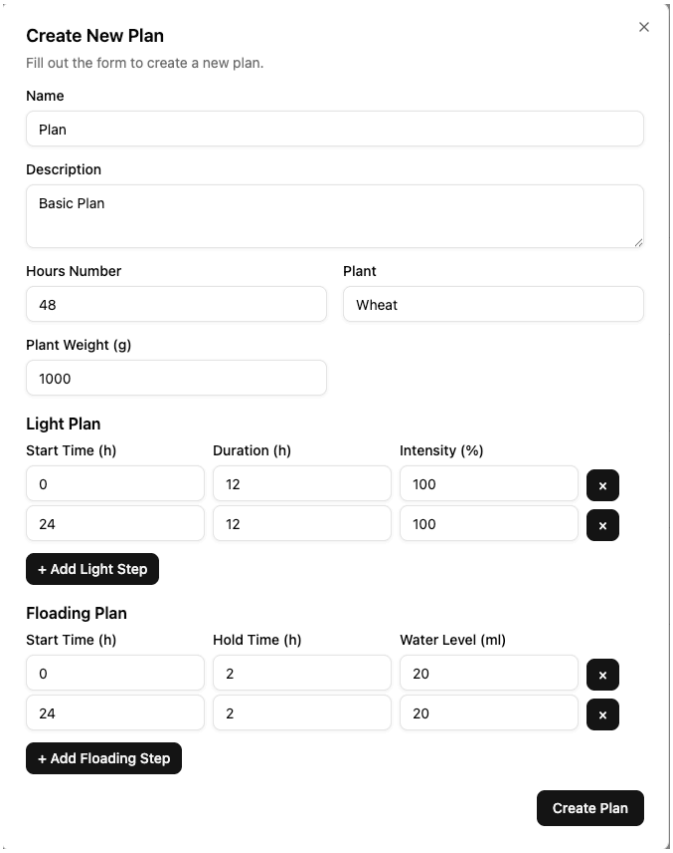

Рисунок 3.8 Модальне вікно створення плану

Створений план зображено на рисунку 3.9.

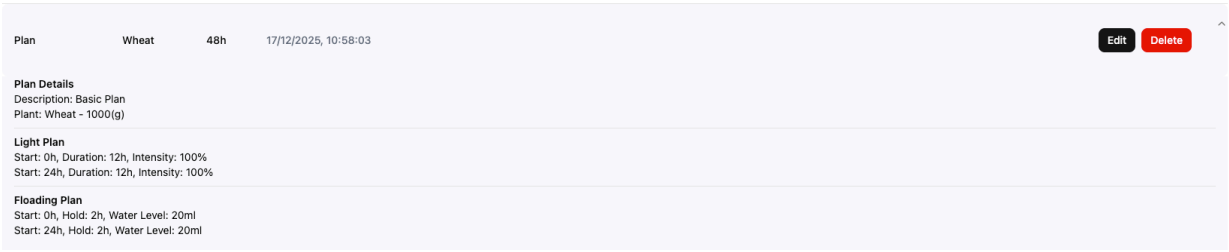

Рисунок 3.9 Створений план

Модальне вікно редагування зображено на рисунку 3.10.

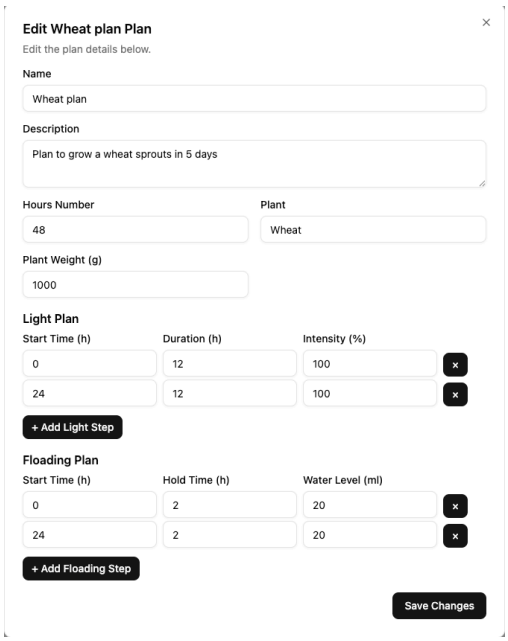

Рисунок 3.10 Модальне вікно редагування

Модальне вікно видалення зображено на рисунку 3.11.

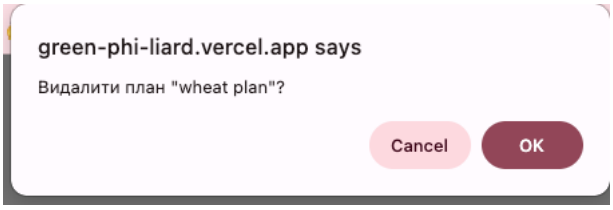

Рисунок 3.11 Модальне вікно видалення

Далі у боковому меню обирається відповідна ферма та створений план використовується для запуску нового циклу вирощування. Вибір ферми у боковому меню розбражено на рисунку 3.12.

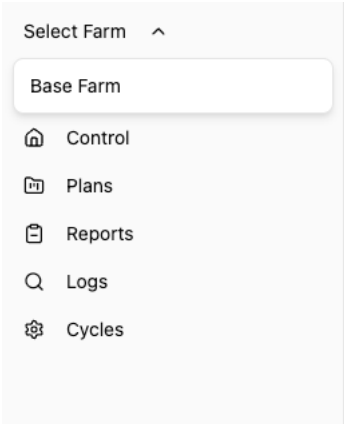

Рисунок 3.12 Бокове меню сторінки

Після вибору відповідної ферми у боковій панелі веб-інтерфейсу користувач призначає для неї план пророщування на сторінці Control(рис 3.13). Призначення плану є обов'язковою умовою для запуску автоматичного режиму роботи ферми.

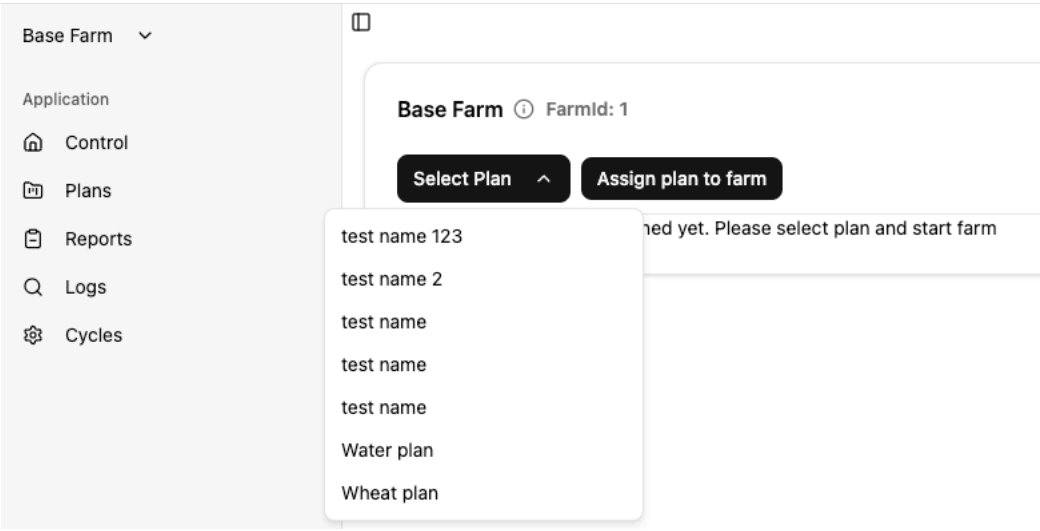

Рисунок 3.13 Процес вибору плану для ферми

Після підтвердження вибраного плану користувач ініціює запуск циклу вирощування. У цей момент серверна частина системи створює новий цикл, генерує розклади керування модулями відповідно до параметрів плану та прив'язує активний цикл до вибраної ферми. Мікроконтролер ESP32, виконуючи періодичні запити до сервера, отримує оновлені керуючі команди та переходить до виконання автоматичного режиму роботи. У вікні Serial Monitor можна перевірити що ESP32 успішно обробляє запити з серверу та отримує команди на викоринання(рис 3.14).

```
----------------------------------
Mode: scheduled
scheduled
Scheduled state: {"light":true,"pump":true,"drain":false,"waterLevel":20}
Sensor raw: 65  →  Water level mm: 0
Pump ON (water too low), expected level: 20
Drain OFF (server disabled)
----------------------------------
```

Рисунок 3.14 Вікно Serial Monitor під час циклу вирощування

Таким чином, запуск циклу забезпечує зв’язок між обраною фермою, планом пророщування та процесом збору й аналізу даних, що надалі використовуються для візуалізації та оцінки результатів вирощування.

У процесі роботи користувач має можливість переглядати інформацію про ферму, заплановані етапи поливу та освітлення, а також поточний стан системи(рис 3.15).

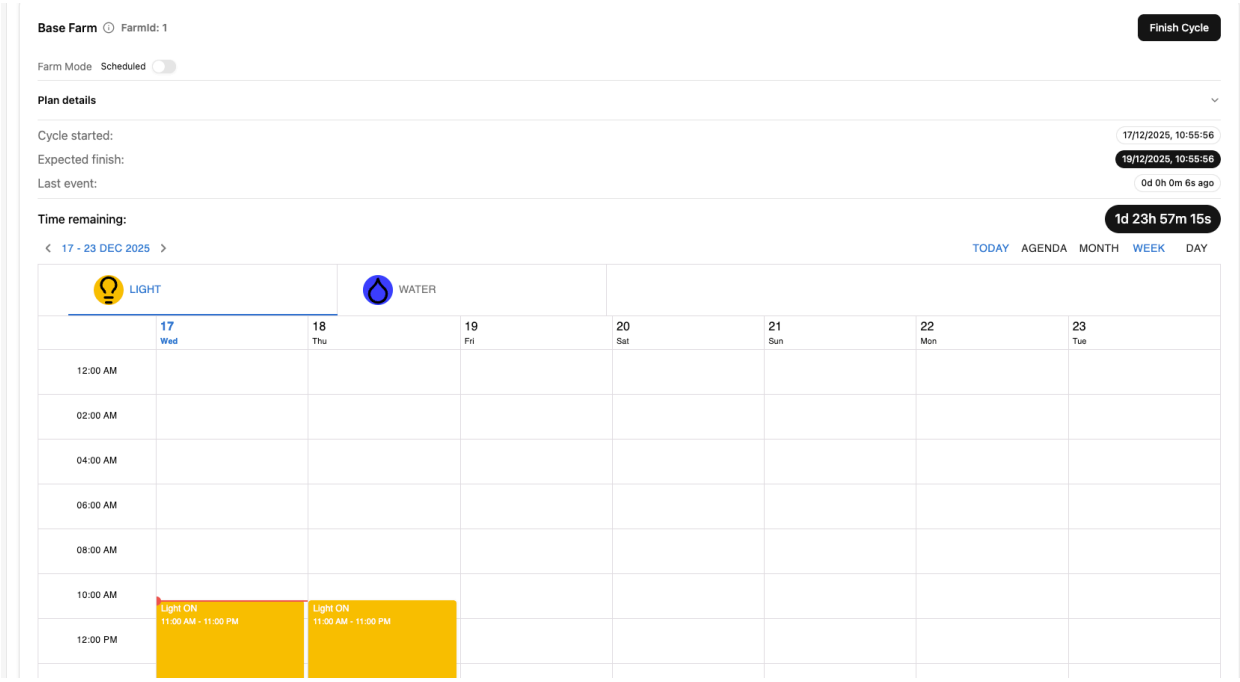

Рисунок 3.15 Інформація про актуальний цикл на сторінці Control

Можемо бачити що світлодіоди, які заміняють освітлення, насос і злив в макеті світяться відповідно до команд, які отримує ESP32(рис 3.16). Жовтний відповідає за освітлення, синів за полив і червоний за злив.

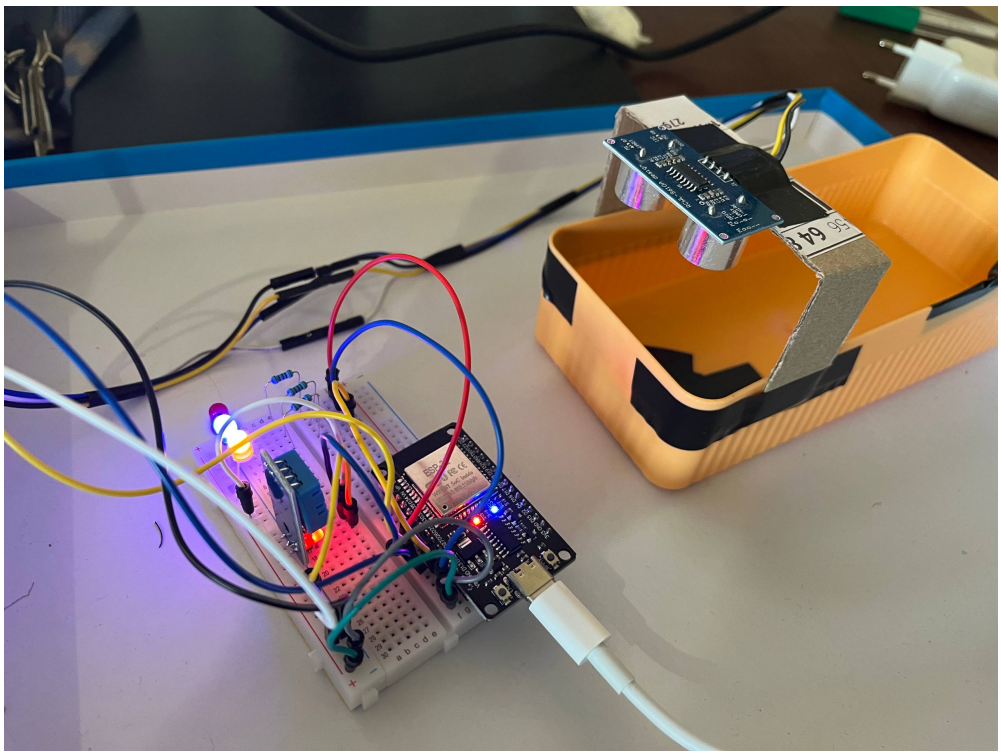

Рисунок 3.16 Демонстраційний макет апаратної частини з індикацією режимів освітлення, поливу та зливу води.

Імітуючи наповнення ємності можемо бачити як після досягнення потрібно рівня води насос вимикається(синій світлодіод). Демонстрація автоматичного вимкнення насосу після досягнення бажаного рівня води зображено на Рисунку 3.17

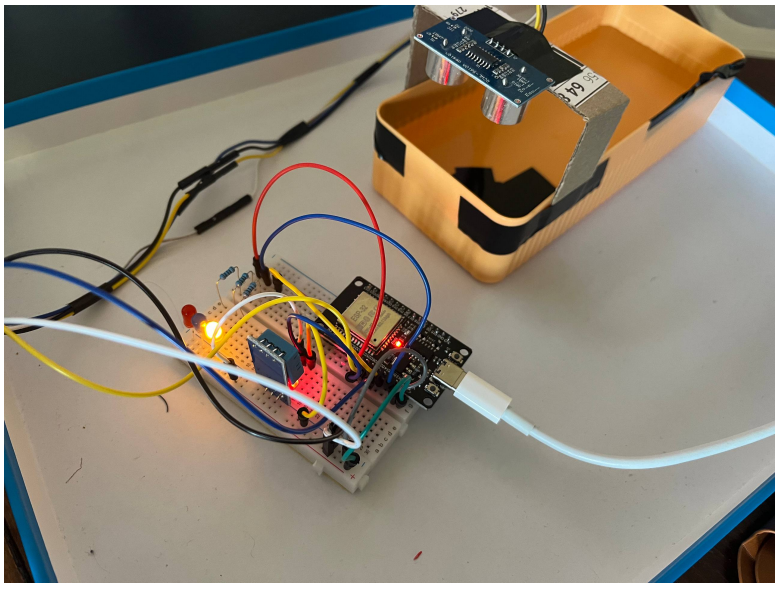

Рисунок 3.17 Демонстрація автоматичного вимкнення насосу після досягнення бажаного рівня води

Перейшовши в ручний режим ферми можемо керувати кожним компонентом окремо. Перехід ферми в ручний режим зображено на рисунку 3.18.

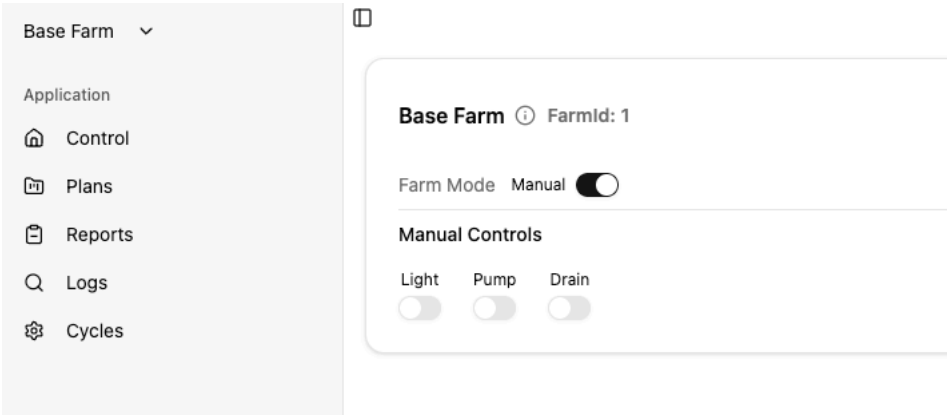

Рисунок 3.18 Перехід ферми в ручний режим

Включивши злив можемо перевірити спостерігати його роботу(червоний світлодіод). Демонстрація ручного керування зливом зображена на рисунку 3.19.

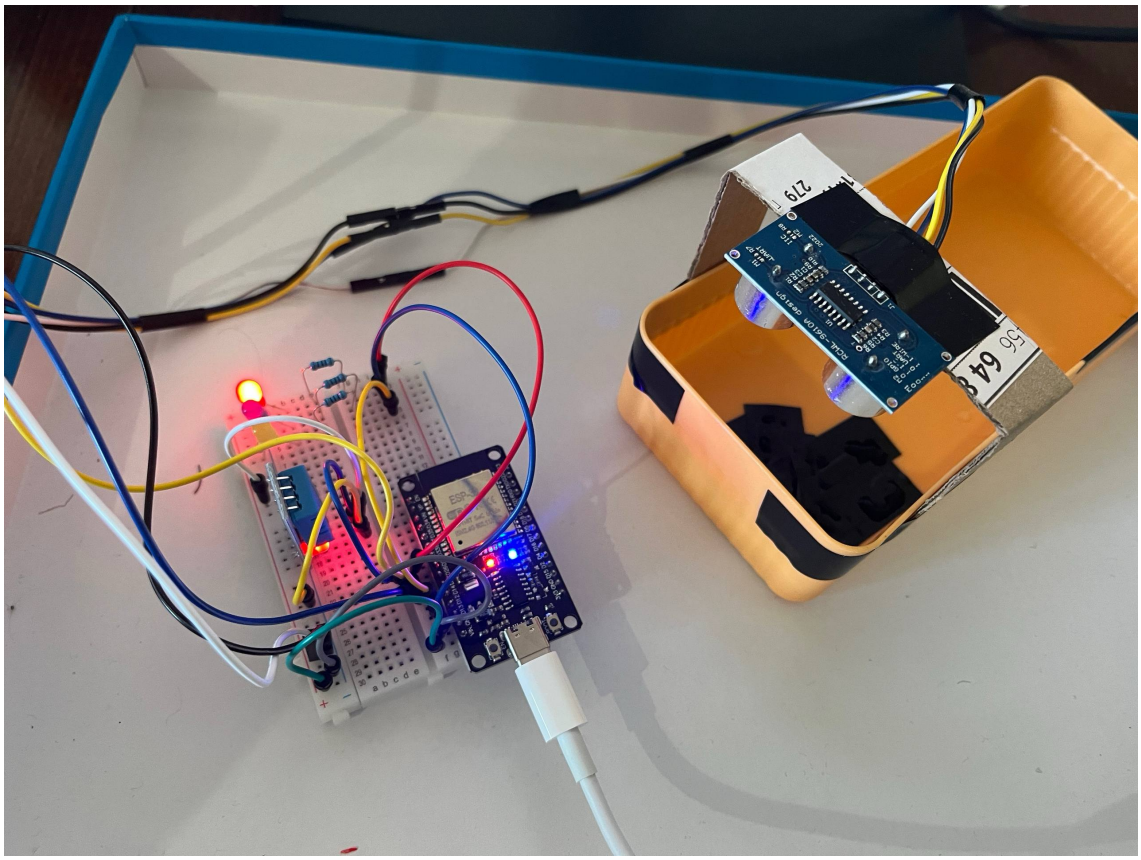

Рисунок 3.19 Демонстрація ручного керування зливом

Завершення циклу та аналіз результатів

Після завершення циклу вирощування користувач завершує цикл через веб-інтерфейс та заповнює модальне вікно з результатами врожаю(рис 3.20). Після цього ферма автоматично переходить у режим manual, у якому не виконуються жодні автоматичні керуючі дії, а система перебуває у стані очікування подальших команд користувача.

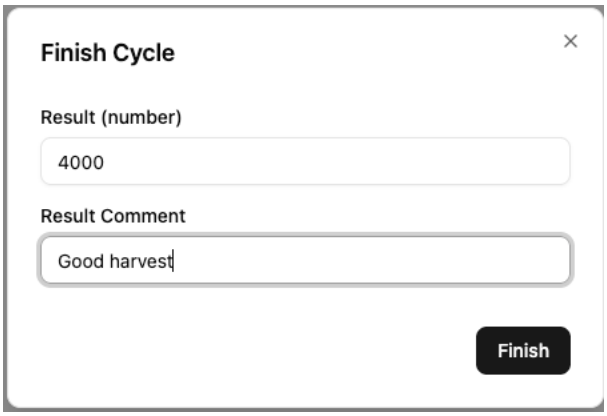

Рисунок 3.20 Завершення циклу

Проведене тестування показало, що користувач має повний контроль над процесом роботи мікроферми, від первинного налаштування апаратної частини до аналізу результатів завершених циклів. Взаємодія між веб-інтерфейсом, серверною частиною та мікроконтролером є стабільною, а отримані дані коректно зберігаються та відображаються у системі.

3.3.2 Аналіз отриманих даних та їх візуалізація

Однією з ключових переваг автоматизованих систем керування мікрофермами є можливість накопичення та аналізу даних, що характеризують умови вирощування. Збереження показників температури, вологості та інформації про перебіг циклів дозволяє не лише контролювати поточний стан ферми, а й аналізувати результати роботи з метою підвищення врожайності та зменшення витрат ресурсів. На основі зібраних даних користувач може коригувати плани пророщування, оптимізувати режими

поливу та освітлення, а також своєчасно виявляти відхилення від оптимальних умов.

Візуалізація температури та вологості протягом циклу

Для аналізу умов вирощування система надає можливість відображення температури та вологості у вигляді часових графіків на сторінці Reports. На рисунку 3.21 показано зміну параметрів середовища протягом усього активного циклу вирощування, який було розглянуто у попередньому підрозділі. Така візуалізація дозволяє оцінити стабільність умов та швидко виявити можливі відхилення від заданих значень.

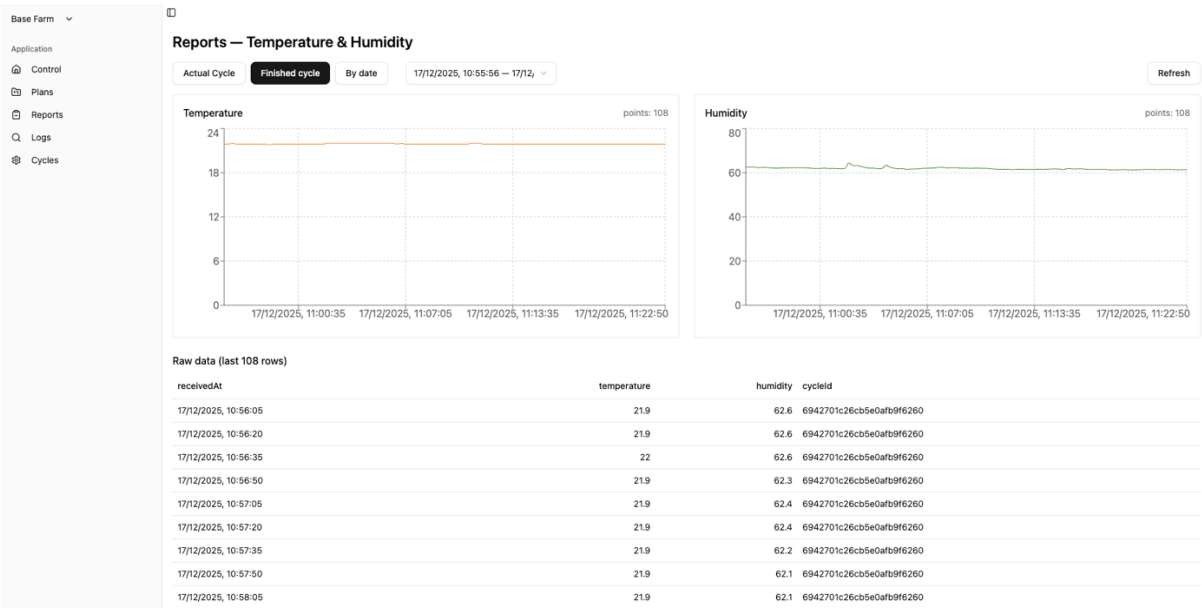

Рисунок 3.21 Сторінка Reports

Окрім перегляду даних за повний цикл, користувач може обрати конкретний часовий інтервал (from / to) для детальнішого аналізу. Це дає змогу зосередитись на окремих етапах вирощування, наприклад, у моменти активного поливу або зміни режимів освітлення. Діаграми вологості та температури за часовий інтервал зображено на рисунку 3.22

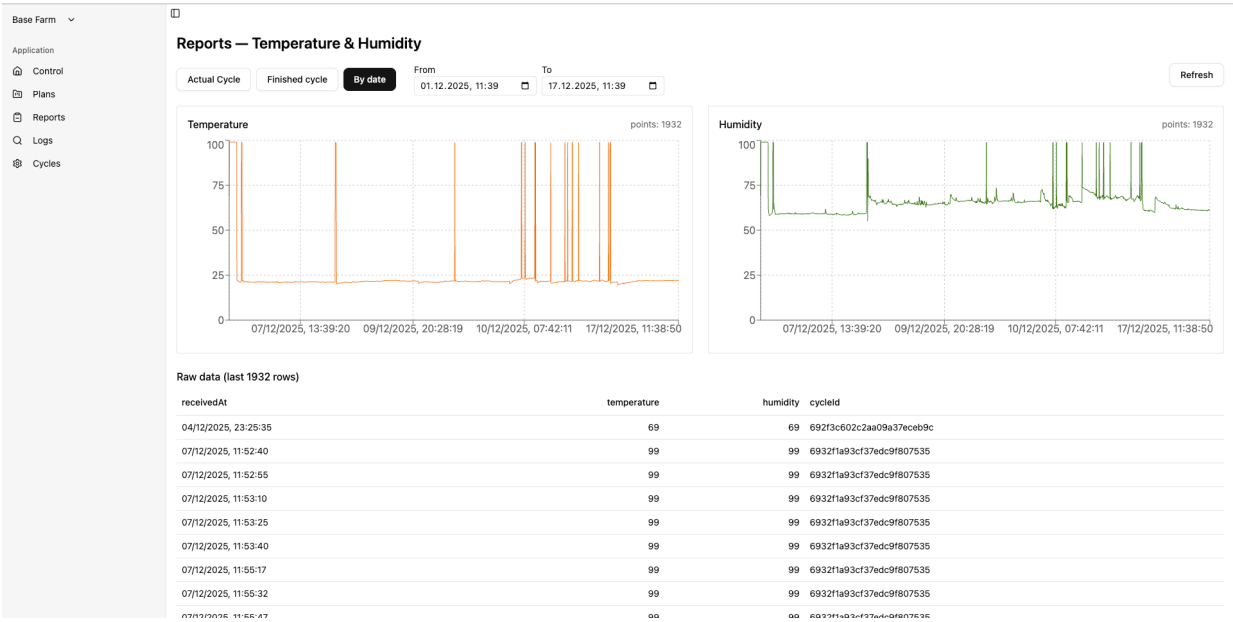

Рисунок 3.22 Діаграми вологості та температури за часовий інтервал

Аналіз журналу подій та помилок

Важливою складовою аналізу роботи системи є перегляд журналу подій. На сторінці Logs користувач має можливість переглядати всі зафіксовані повідомлення, відсортовані за типом події. Особливу увагу приділено відображенню помилок і попереджень, що дозволяє швидко ідентифікувати проблеми у роботі сенсорів або відхилення параметрів середовища.

Фільтрація логів за типом події спрощує аналіз нештатних ситуацій та дозволяє оцінити стабільність роботи системи протягом усього циклу вирощування. Сторінка Logs зображена на рисунку 3.23.

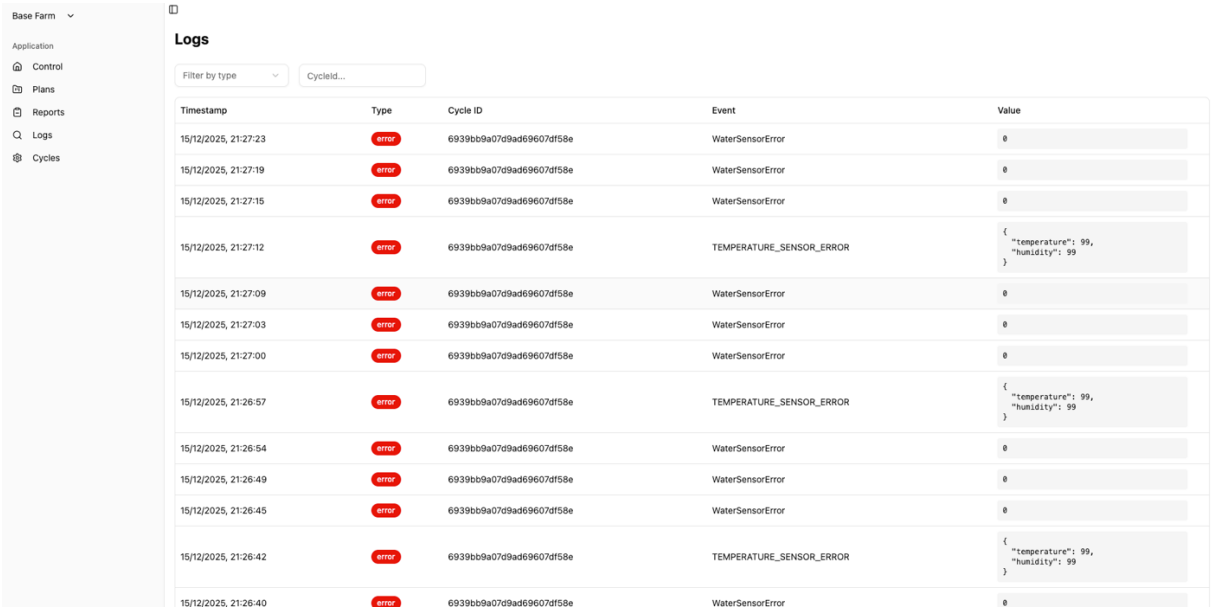

Рисунок 3.23 Сторінка Logs

Перегляд інформації про завершені цикли

Для комплексного аналізу роботи ферми система надає сторінку Cycles, де відображається інформація про всі завершені цикли вирощування. Користувач може переглянути використані плани, тривалість циклів, результати врожаю та коментарі, додані під час завершення циклу.

Наявність історії циклів дозволяє порівнювати різні плани пророщування між собою та оцінювати їх ефективність на основі фактичних результатів. Сторінка Cycles зображена на рисунку 3.24.

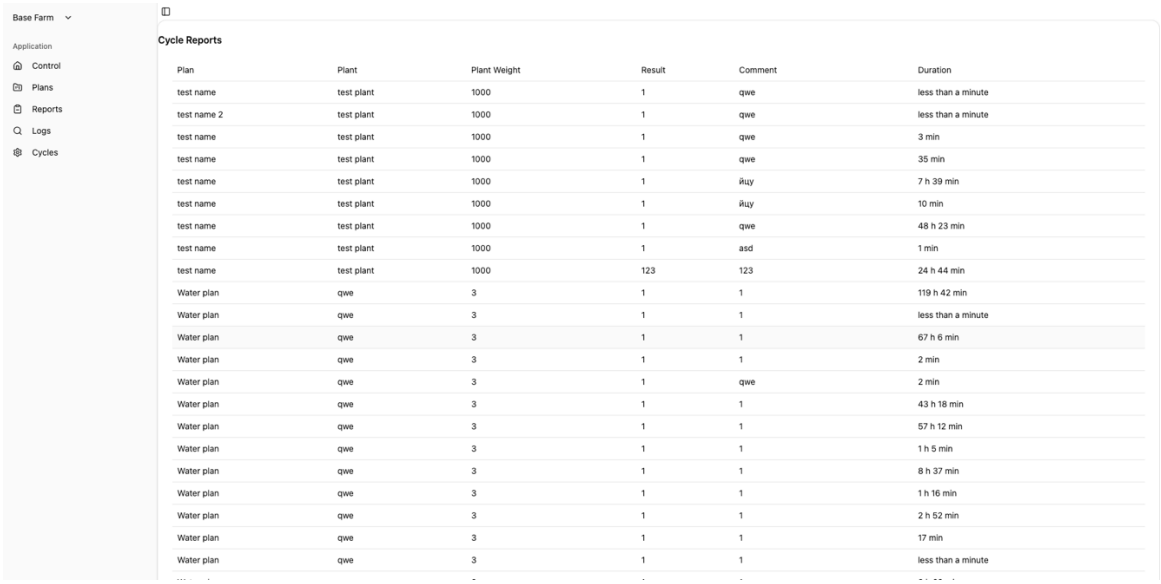

Рисунок 3.24 Сторінка Cycles

Проведений аналіз отриманих даних та їх візуалізація підтверджують, що розроблена система забезпечує не лише автоматизоване керування процесом вирощування, а й надає інструменти для глибшого аналізу умов роботи мікроферми. Використання зібраної інформації дозволяє приймати обґрунтовані рішення щодо оптимізації процесів, що сприяє підвищенню врожайності та зменшенню витрат на експлуатацію ферми.

3.3.3 Тестування нештатних ситуацій

Для перевірки надійності та стабільності роботи системи було проведено тестування у нештатних ситуаціях. Метою даного етапу є перевірка здатності системи коректно реагувати на помилки сенсорів і проблеми з мережею, забезпечуючи безпечний режим роботи та своєчасне інформування користувача.

Нештатна ситуація 1 — помилка зчитування температури та вологості

У першому тестовому сценарії імітується помилка зчитування даних із датчика температури та вологості шляхом від’єднання провідника сенсора. У результаті мікроконтролер передає на сервер резервні значення, які інтерпретуються як критична помилка сенсора.

Серверна частина системи фіксує помилку у журналі подій та надсилає сповіщення на електронну пошту користувача. Для запобігання надмірній кількості повідомлень реалізовано механізм обмеження частоти сповіщень: повторне повідомлення надсилається лише у випадку, якщо помилку було усунуто та вона з’явилась повторно не раніше ніж через 20 хвилин. Лист про помилку зображено на рисунку 3.25.

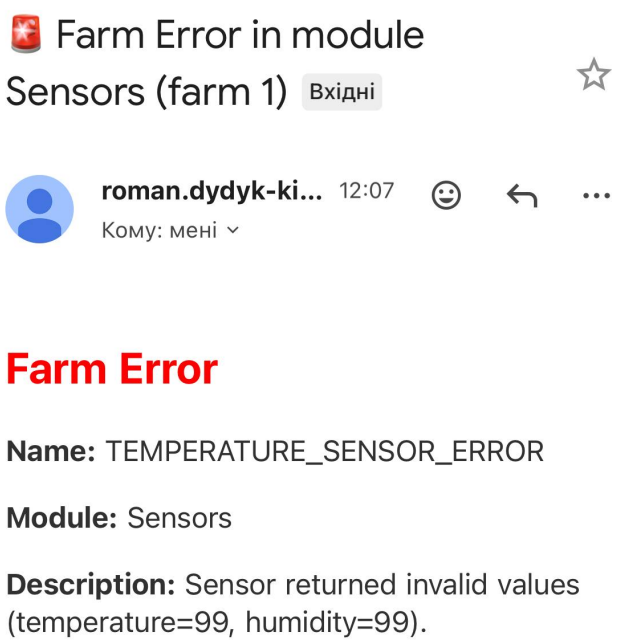

Рисунок 3.25 Лист про помилку зчинування датчику температури

Таким чином, користувач оперативно отримує інформацію про проблему, при цьому система не створює надлишкового навантаження у вигляді повторюваних повідомлень.

Нештатна ситуація 2 — помилка зчитування рівня води

У другому сценарії тестування імітується помилка зчитування рівня води шляхом від'єднання ультразвукового датчика. У разі виявлення некоректних показників система переходить у безпечний режим: подача води автоматично вимикається, незалежно від отриманих керуючих команд.

Інформація про помилку фіксується у журналі подій та надсилається користувачу електронною поштою. Насос залишається вимкненим до моменту усунення несправності, що виключає ризик неконтрольованої подачі води. Поведінка системи підтверджує коректну реалізацію захисної логіки та пріоритет безпеки над автоматизацією. Лист про помилку сенсора зображено на рисунку 3.26.

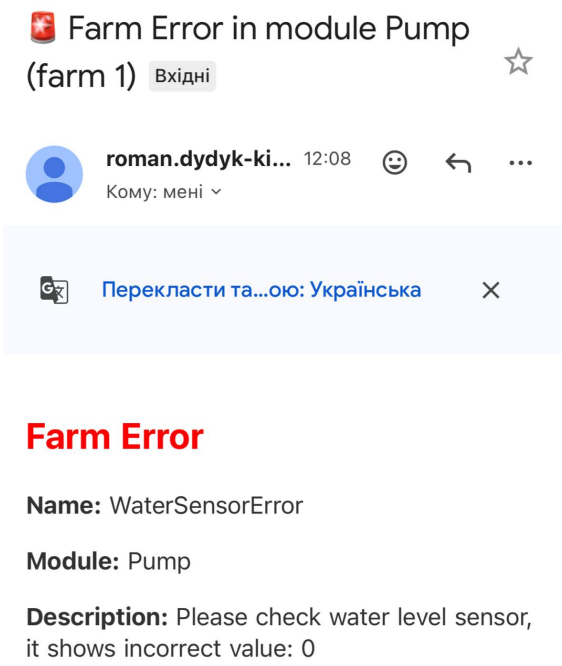

Рисунок 3.25 Лист про помилку зчинування сенсора рівня води

Нештатна ситуація 3 — втрата мережевого з'єднання

У третьому сценарії перевіряється поведінка системи у випадку втрати підключення до мережі Wi-Fi. За відсутності зв'язку з серверною частиною мікроконтролер автоматично вимикає всі виконавчі модулі та переходить у режим очікування.

Система не виконує жодних автоматичних дій до моменту відновлення з'єднання з мережею. Після повторного підключення мікроконтролер відновлює обмін даними з сервером та продовжує роботу відповідно до поточного режиму ферми. Такий підхід забезпечує стабільну та передбачувану поведінку системи навіть за умов нестабільного мережевого середовища. На рисунку 3.26 зображено повідомлення у вікні Serial Monitor про безпечний режим ферми.

```
No connection with server
```

Рисунок 3.26 повідомлення у вікні Serial Monitor про безпечний режим ферми

Результати тестування нештатних ситуацій підтверджують, що розроблена система здатна коректно реагувати на помилки сенсорів і втрату мережевого з’єднання. Реалізовані механізми логування, сповіщення та безпечного режиму роботи забезпечують високий рівень надійності та мінімізують ризики під час експлуатації мікроферми.

ВИСНОВКИ

У ході виконання магістерської роботи було розглянуто задачу автоматизованого керування параметрами середовища мікроферми, орієнтованої на вирощування соковитої зеленої кормової маси та мікрозелені в контрольованих умовах. Основна увага приділялася не створенню фізичної ферми як такої, а розробці універсальної програмно-апаратної системи керування, яка може бути інтегрована з виконавчими компонентами різної потужності та масштабу.

У роботі проаналізовано сучасні підходи та існуючі рішення у сфері автоматизованих систем вирощування, що дозволило виявити потребу у гнучких, модульних та доступних системах керування, орієнтованих на малі та середні фермерські господарства. Показано, що значна частина рішень на ринку або надмірно орієнтована на промислове використання, або не забезпечує достатнього рівня автоматизації та дистанційного контролю.

У результаті було спроєктовано та реалізовано архітектуру системи, яка поєднує апаратну частину на базі мікроконтролера ESP32, серверну частину з REST API та веб-інтерфейс для взаємодії з користувачем. Такий підхід забезпечує чіткий розподіл відповідальності між компонентами системи, підвищує гнучкість керування та спрощує подальше розширення функціональності.

Розроблена серверна частина реалізує логіку формування циклів вирощування, зберігання сенсорних даних, керування режимами роботи ферми та обробку нештатних ситуацій. Використання бази даних MongoDB дозволяє ефективно зберігати історію вимірювань, логів і результатів завершених циклів, що створює основу для подальшого аналізу та оптимізації процесів вирощування.

Клієнтська частина системи реалізована у вигляді адаптивного веб-інтерфейсу, який забезпечує зручне керування фермами, планами пророщування та циклами вирощування. Інтерфейс дозволяє користувачу мінімізувати час, витрачений на ручний догляд за фермою, та оперативно

отримувати інформацію про стан системи, результати вимірювань і можливі відхилення.

Апаратна частина системи реалізує функції збору параметрів середовища та виконання керуючих команд. У межах демонстраційного макета реальні виконавчі пристрої замінено індикаторами, що дозволило перевірити коректність логіки керування без використання потужних компонентів. Запропонований підхід робить систему універсальною та придатною для інтеграції з реальними насосами, освітленням і клапанами різної потужності.

Проведене тестування підтвердило працездатність системи в основних режимах роботи, а також її здатність коректно реагувати на нештатні ситуації, зокрема помилки сенсорів і втрату мережевого з'єднання. Реалізовані механізми логування, безпечного режиму та сповіщення користувача підвищують надійність системи та безпеку її експлуатації.

Збереження та візуалізація отриманих даних дозволяють аналізувати умови вирощування, порівнювати результати різних циклів і приймати обґрунтовані рішення щодо оптимізації процесів. Це створює передумови для підвищення врожайності та зменшення витрат ресурсів, що є особливо важливим для малих і середніх фермерських господарств.

Розроблена система є актуальним, гнучким та масштабованим рішенням, яке може бути використане як основа для подальшого розвитку. Перспективними напрямами вдосконалення є розширення набору сенсорів та їх точність, реалізація авторизації користувачів, підтримка керування кількома фермами, а також формування автоматизованих звітів у форматі PDF.

Таким чином, у межах магістерської роботи було досягнуто поставленої мети, розроблено систему автоматизованого керування параметрами середовища мікроферми, яка зменшує участь людини у процесі догляду, забезпечує стабільність умов вирощування та може бути адаптована для використання у різних аграрних сценаріях.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. React documentation [Електронний-ресурс].- URL: https://uk.reactjs.org/docs/getting-started.html
2. Next.js documentation [Електронний-ресурс].- URL: https://nextjs.org/docs
3. MUI documentation [Електронний-ресурс].- URL: https://ui.shadcn.com/docs
4. CSS MDN Web Docs [Електронний-ресурс].- URL: https://developer.mozilla.org/en-US/docs/Web/CSS
5. HTML MDN Web Docs [Електронний-ресурс].- URL: https://developer.mozilla.org/en-US/docs/Web/HTML
6. TypeScript Documentation [Електронний-ресурс].- URL: https://www.typescriptlang.org/docs/
7. MongoDB documentation [Електронний-ресурс].- URL: https://www.mongodb.com/docs/
8. Node.js documentation [Електронний-ресурс].- URL: https://nodejs.org/docs/latest/api/
9. Axios documentation [Електронний-ресурс].- URL: https://axios-http.com/docs/intro
10. Postman documentation [Електронний-ресурс].- URL: https://learning.postman.com/docs/introduction/overview/
11. Next.js Best Practices: Tips for Clean and Efficient Code Article [Електронний-ресурс].- URL: https://rohandalvii.medium.com/next-js-best-practices-tips-for-clean-and-efficient-code-99dd31a14797
12. What is Next.js and its benefits? Article [Електронний-ресурс].- URL: https://medium.com/@asiandigitalhub/what-is-next-js-and-its-benefits-8b13aab56bfd

13. Магістерська робота: методичні вказівки до змісту та оформлення для студентів спеціальності 123 - Комп'ютерна інженерія/ Мельничук С.І. - Івано-Франківськ: Видавництво ІФНТУНГ, 2023. – 29с.

14. What is REST? Article [Електронний-ресурс].- URL: https://restfulapi.net/

15. Industry-leading solutions to simplify and scale IT for industrial networks Article [Електронний-ресурс].- URL: https://www.cisco.com/site/us/en/solutions/networking/industrial-iot/index.html

16. Learn Everything About NextAuth.js v5 with Next.js, Shadcn UI & MongoDB [Електронний-ресурс].- URL: https://javascript.plainenglish.io/learn-everything-about-nextauth-js-v5-with-next-js-shadcn-ui-mongodb-5e2ed8ebd5c4

17. ESP32 Publish Sensor Readings [Електронний-ресурс].- URL:https://randomnerdtutorials.com/esp32-esp8266-publish-sensor-readings-to-google-sheets/

18. ESP32 with HC-SR04 Ultrasonic Sensor with Arduino IDE [Електронний-ресурс].- URL: https://randomnerdtutorials.com/esp32-hc-sr04-ultrasonic-arduino/

19. Ultrasonic hc-sr04 Sensor Interfacing with ESP32 [Електронний-ресурс].- URL: https://www.electronicwings.com/esp32/ultrasonic-hc-sr04-sensor-interfacing-with-esp32

20. DHT11/DHT22 Sensor Guide for Arduino [Електронний-ресурс].- URL: https://www.scribd.com/document/337029837/Complete-Guide-for-DHT11

21. How to use ArduinoJson with HTTPClient? [Електронний-ресурс].- URL: https://arduinojson.org/v6/how-to/use-arduinojson-with-httpclient

22. Mario's Ideas: LED Strips & Arduino – Understanding 5V and 12V Single-Color Strips [youtube відео] - URL: https://youtu.be/ms8_rbJWyoo?si=N2j_Rh8GWJRHquau

23. Core Electronics: How To Use A Water Pump with an Arduino [youtube відео] – URL: https://youtu.be/UEL9NR2Z5BY?si=9jFrrEzph-CUmztA

24. Barley Fodder Sprouting Trials continued: New Flood and Drain Tray System Installed [Електронний-ресурс] – URL: https://pacapride.wordpress.com/2012/06/08/barley-fodder-sprouting-trials-continued-new-flood-and-drain-tray-system-installed/

25. Li, Y., Dufault, R., & Leng, X. Automation and Monitoring Systems for Hydroponic Crop Production, Journal of Agricultural Machinery, 2020

26. Banks, A., Porcello, E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. O'Reilly Media, 2020.

27. Freeman, E., Robson, E. Head First JavaScript Programming. O'Reilly Media, 2014.

28. Wieruch, R. The Road to React. Leanpub, 2020.

29. Mesbah, A., Deursen, A. Software Architecture of Web Applications. Springer, 2017.

30. Next.js SEO for Developers – How to Build Highly Performant Apps with Next [Електронний-ресурс].- URL: https://www.freecodecamp.org/news/nextjs-seo/

31. Роберт Сесіл Мартін. Чиста архітектура 2019р.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема магістерської роботи: Вирішення задачі автоматизованого керування параметрами середовища мікроферми

Обсяг пояснювальної записки 87 аркушів.

\- таблиць;

36 рисунки;

\- додаток.

Дата завершення роботи *грудень 2025р*.

Підпис студента-дипломника _____

_____