ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система моніторінгу та керування IoT-пристроями на основі ESP»

на здобуття освітнього ступеня бакалавра

зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)

освітньо-професійної програми «Інформаційні системи та технології»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерел.

_____ Денис ПАНТЕЛЕЄВ

(*підпис*) *Ім'я ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти групи ІСД-41

Денис ПАНТЕЛЕЄВ
Ім'я ПРІЗВИЩЕ

Керівник: Владислав ЗАВАЦЬКИЙ
науковий ступінь, вчене звання Ім'я ПРІЗВИЩЕ

Рецензент: _____
науковий ступінь, вчене звання Ім'я ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інформаційні системи та технології

Ступень вищої освіти Бакалавр

Спеціалність 126 Інформаційні системи та технології

Освітньо-професійної програми «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІСТ
_____ Каміла СТОРЧАК

«_____» _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Пантелеєва Дениса Анатолійовича
(прізвище, ім'я, по батькові здобувач)

1. Тема кваліфікаційної роботи: «Розробка системи моніторінгу та керування IoT-пристроями на основі ESP»
   керівник кваліфікаційної роботи Владислав ЗАВАЦЬКИЙ
   (Ім'я ПРІЗВИЩЕ)
   затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.
2. Строк подання кваліфікаційної роботи «30» травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи:
   3.1. Технічна документація з описом мікроконтролерів сімейства ESP,
   3.2. Програмна документація мікроконтролерів сімейства ESP,
   3.3. Офіційна документація Python.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   4.1. Пошук та аналіз існуючих аналогічних рішень.
   4.2. Розробка IoT-пристроїв
   4.3. Розробка центральної системи керування
5. Перелік графічного матеріалу: *презентація*
6. Дата видачі завдання «24» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання	24.02-06.03.2025	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2025	
3	Розробка тестової моделі	14.03-21.03.2025	
4	Стандартизація програмних компонентів	22.03-31.03.2025	
5	Розробка системи	01.04-19.04.2025	
6	Тестування розробленої системи	20.04-25.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2025	
8	Розробка демонстраційних матеріалів	06.05-20.05.2025	

Здобувач вищої освіти _____ Денис ПАНТЕЛЕЄВ
(підпис) *(Ім'я ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Владислав ЗАВАЦЬКИЙ
(підпис) *(Ім'я ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 51 стор., 2 табл., 22 рис., 15 джерел.

Мета роботи – підвищення гнучкості та надійності IoT-систем шляхом реалізації мультицентричної архітектури керування.

Об'єкт дослідження – процес керування IoT-пристроями у розподіленому середовищі.

Предмет дослідження – програмне забезпечення для мультицентричного керування IoT-пристроями.

Короткий зміст роботи: В даній дипломній роботі було розроблено систему мультицентричного керування IoT-пристроїв. Для кожного IoT-пристрою була розроблена стандартизоване програмне забезпечення, що написане на мові програмування C++. Програмне забезпечення дозволяє пристрою працювати самостійно або з'єднуватися з терміналом керування для ручого контролю. Програмне забезпечення змінюється залежно від функціоналу пристрою. Термінал керування є розробленим інтерфейсом для зручного контролю кожного пристрою. Щоб спростити роботу з інтерфейсом термінала керування було додано лише необхідні функції для керування пристроями. Термінал керування був написаний на мові програмування Python за допомогою модуля графічних інтерфейсів PyQt5.

КЛЮЧОВІ СЛОВА: Керування IoT-пристроями, термінал керування, Python, модуль PyQt5, C++, мікроконтролери сімейства ESP.

ЗМІСТ

ВСТУП

Інтернет речей (Internet of Things, IoT) стрімко розвивається та відіграє дедалі важливішу роль у сучасному світі. Розумні будинки, автоматизовані виробництва, системи моніторингу навколишнього середовища — усе це приклади практичного впровадження IoT-рішень у повсякденне життя. Потреба в ефективному керуванні великою кількістю пристроїв зростає, що зумовлює необхідність розробки зручних, гнучких та надійних систем керування.

Традиційні підходи до керування IoT-пристроями здебільшого базуються на централізованих архітектурах, де один центральний контролер відповідає за збір та обробку інформації, а також за взаємодію з усіма пристроями. Однак такий підхід має низку обмежень: уразливість до відмови центрального вузла, труднощі з масштабуванням системи, а також залежність від наявності постійного підключення до мережі.

У відповідь на ці виклики було запропоновано концепцію мультицентричного керування, в якій кілька незалежних терміналів можуть одночасно здійснювати керування IoT-пристроями. Така архітектура підвищує надійність і гнучкість системи, дозволяє працювати з пристроями навіть при часткових збоях зв’язку та спрощує розгортання в умовах обмежених ресурсів.

У даній дипломній роботі було розроблено систему мультицентричного керування IoT-пристроями. Кожен пристрій отримав власне програмне забезпечення, написане мовою програмування C++, що дозволяє йому функціонувати як автономно, так і під контролем термінала. У якості терміналів керування було реалізовано інтерфейс користувача, створений з використанням Python та бібліотеки PyQt5. Інтерфейс містить лише необхідні функції, що забезпечують просте та інтуїтивно зрозуміле керування пристроями.

Об’єкт дослідження – процес керування IoT-пристроями у розподіленому середовищі.

Предмет дослідження – програмне забезпечення для мультицентричного керування IoT-пристроями.

Мета роботи – підвищення гнучкості та надійності IoT-систем шляхом реалізації мультицентричної архітектури керування.

Основними завданнями цієї роботи є**:**

1. Проаналізувати предметну область систем керування IoT-пристроями.
2. Дослідити переваги та недоліки існуючих систем керування IoT-пристроями.
3. Визначити вимоги до архітектури системи та програмного забезпечення IoT-пристроїв.
4. Розробити інтерфейс терміналу керування на основі PyQt5.
5. Реалізувати програмне забезпечення для IoT-пристроїв на C++.
6. Провести тестування системи на стабільність, зручність та функціональність.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ

1.1. Актуальність теми

Зараз, цифровізація охоплює дедалі більше сфер людської діяльності, IoT посідає особливе місце як одна з найперспективніших і найшвидше зростаючих напрямів. IoT уже давно перестав бути лише концепцією — сьогодні це цілком реальна інфраструктура, що об'єднує мільярди пристроїв по всьому світу. Його застосування є надзвичайно широким: від інтелектуальних побутових систем (розумні будинки, клімат-контроль, охоронні системи) до екологічного моніторингу, розумних міст (smart city), промислової автоматизації (IIoT) та навіть охорони здоров’я. Такий стрімкий розвиток викликає об’єктивну необхідність переосмислення підходів до побудови архітектури керування цими пристроями.

Головною перевагою IoT є здатність забезпечувати віддалений контроль, збір даних у реальному часі, моніторинг стану об’єктів і оперативне реагування на зміни в середовищі. Ці функції реалізуються через різноманітні інтерфейси — від комп’ютерних додатків до мобільних застосунків та веб-інтерфейсів. Проте більшість наявних рішень все ще базуються на централізованій моделі керування, де всі пристрої підпорядковані одному серверу або головному контролеру, який виконує функції посередника між користувачем і фізичними елементами системи.

Централізована архітектура, незважаючи на її зручність та простоту реалізації, має багато обмежень, які стають критичними при масштабуванні або збоях. До таких недоліків належать: висока вразливість системи до відмови центрального вузла, труднощі з розширенням мережі без істотних змін у структурі, обмежена гнучкість у керуванні та залежність від стабільного з’єднання з центральним контролером. Усе це знижує загальну надійність IoT-системи.

У контексті зростаючої кількості IoT-пристроїв, посилення вимог до безпеки, надійності та доступності систем, все більше уваги набуває концепція мультицентричного управління. Це підхід, за якого контроль над пристроями здійснюється не єдиним умовним центром, а кількома рівноправними терміналами.

Це можуть бути різні види клієнтських додатків — комп’ютерні додатки, мобільні застосунки, веб-інтерфейси — які мають змогу напряму або опосередковано взаємодіяти з пристроями, не потребуючи обов’язкової централізації керування.

Мультицентрична архітектура відкриває нові можливості в побудові адаптивних і стійких IoT-систем. Вона підвищує рівень гнучкості, дозволяє швидше реагувати на зміни в середовищі, розподіляє навантаження між кількома точками контролю, зменшує ризик збоїв та полегшує інтеграцію нових пристроїв. Це особливо актуально у випадках використання IoT у віддалених або важкодоступних регіонах, у мобільних або тимчасових інфраструктурах, а також у побутових умовах, де централізоване керування є недоцільним або економічно необґрунтованим.

Таким чином, розробка інноваційної системи управління IoT-пристроями, яка працює на принципі мультицентричності, передбачає створення універсального, стандартизованого програмного забезпечення для пристроїв та інтуїтивно зрозумілого, зручного у використанні інтерфейсу для користувача. Саме такий підхід дає змогу ефективно реалізовувати функції контролю, моніторингу та взаємодії, забезпечуючи при цьому високу адаптивність і стійкість системи.

1.2. Аналіз існуючих систем керування IoT-пристроями

Сьогодні існує багато різноманітних рішень, що забезпечують керування пристроями в рамках IoT. Вони можуть відрізнятися за архітектурою, принципом роботи, масштабом, призначенням та рівнем відкритості. В цілому їх можна можна розділити за такими критеріями: централізовані та децентралізовані (мультицентричні), комерційні та відкриті, локальні та хмарні.

Найпоширенішим типом залишаються централізовані хмарні платформи. До таких належать: Amazon AWS IoT, Microsoft Azure IoT Hub, Blynk та інші. Усі вони мають центральній серверний вузл, що приймає, обробляє та перенаправляє дані між пристроями та користувачами. Такі платформи забезпечують високу

масштабованість, багатий функціонал, зберігання даних, автоматизації та інтеграції з іншими сервісами.

Попри переваги, централізовані системи мають кілька суттєвих недоліків:

1) залежність від інтернет-з’єднання з хмарним сервером. У разі втрати зв’язку керування пристроями або моніторинг можуть стати недоступними.
2) Прив’язка до конкретного сервісу обмежує гнучкість у налаштуванні.
3) Обмеження автономності. Система без доступа до центрального вузла втрачає функціональність.

Альтернативою для централізованим платформ є відкриті системи з локальним використвнням. Таких як наприклад: Home Assistant — відкрита платформа для домашньої автоматизації з підтримкою великої кількості пристроїв. Такі системи забезпечують більший контроль над даними та зменшують залежність від зовнішніх серверів, однак потребують технічних знань для налаштування. Більшість із них, попри локальність, усе ще побудовані на централізованій архітектурі, що обмежує масштабованість.

Окрему нішу займають універсальні системи, які поєднують локальне керування з можливістю доступу через хмару. Наприклад: Mesh-мережі на базі Zigbee або Z-Wave, де пристрої взаємодіють напряму. Але поки такі підходи залишаються технічно складними та не надто поширеними в споживчому сегменті, хоча вони мають потенціал до створення більш стійких, масштабованих і автономних систем.

1.2.1. Amazon AWS IoT.

Amazon AWS IoT підтримує MQTT, HTTPS і WebSockets, тому найчастіше до неї підключають пристрої з підтримкою шифрування TLS та криптографічних сертифікатів. Це вимагає певної обчислювальної потужності, а отже, й відповідної

апаратної бази. Мікроконтролери сімейства *ESP* одні з найпопулярніших варіантів для розробки. Вони здатні підтримувати TLS-з'єднання і підключатися до Amazon AWS через MQTT запити. Також мінікомп'ютери *Raspberry Pi*, які використовується як повноцінний пристрій-клієнт, який не лише збирає дані, а й обробляє їх локально, надсилаючи вже оброблену інформацію в хмару Amazon AWS. Для дійсно складних проєктів зазвичай використовують *STM32* або інші контролер з FreeRTOS. Використовуються в промислових варіантах, часто з SDK від AWS IoT Device SDK.

Особливістю цих пристроїв є — необхідність зберігати ключі та сертифікати в безпечному середовищі, іноді за допомогою додаткових чіпів безпеки.

Amazon AWS IoT Core сам по собі немає графічного інтерфейсу або простих налаштувань для звичайного користувача. Amazon AWS надає повний контроль над пристроями, політиками безпеки, сертифікатами тощо, але потребує серйозної підготовки для використання. Робота здійснюється через інші сервіси AWS або сторонні програми: AWS IoT Console, Amazon CloudWatch, AWS Lambda, Alexa Smart Home API. Орієнтований переважно на досвідчених користувачів і інженерів.

AWS IoT Console призначена для розробників. Дозволяє бачити статус пристроїв, журнали повідомлень та створювати правила для обробки даних. *Amazon CloudWatch та AWS Lambda* використовуються для автоматизації подій та створення тригерів. Інтеграція *з Alexa Smart Home API* дозволяє створити голосовий інтерфейс управління IoT-пристроями. Є також можливість створити кастомні *веб-додатки* — зазвичай створюються розробником з використанням AWS Amplify, React/Vue.js і API Gateway.

1.2.2. Microsoft Azure IoT Hub.

Microsoft Azure IoT Hub надає доступ до великого функціоналу та добре інтегрується з іншими службами Microsoft, однак аналогічно до Amazon AWS

потребує глибокого технічного розуміння роботи хмарних сервісів і IoT-інфраструктури.

Microsoft Azure також підтримує MQTT, AMQP і HTTPS. Особливістю Azure є наявність IoT Plug and Play, що дозволяє створювати цифрові моделі пристроїв для автоматичної конфігурації. Для роботи з Microsoft Azure є *MXChip AZ3166* — фірмовий Azure-адаптований пристрій із вбудованими датчиками, Wi-Fi та підтримкою Azure SDK. Для початківцев чи інтузіастів є можливість використання мікронтролера *ESP32* з Azure IoT SDK — може бути використаний як бюджетна альтернатива, проте потребує ручної інтеграції. Також можна використати мінікомп'ютер *Raspberry Pi*. Він часто використовується як edge-пристрій або шлюз, що підключає простіші сенсори до Azure Hub через MQTT Bridge.

Microsoft Azure також не є готовими для користувача, проте має механізми для створення власних рішень: *Azure IoT Central* — платформа з графічним веб-інтерфейсом, де можна відображати телеметрію, налаштовувати керування та переглядати стан пристроїв. *Power BI* — може бути під'єднаний для візуалізації даних з Azure Інтеграція з *Microsoft Power Apps* — створення власних мобільних чи веб-застосунків з мінімумом коду. Голосові команди через *Azure Cognitive Services* або *Cortana* — дозволяють створити голосовий інтерфейс.

Azure часто використовується для складних сценаріїв аналітики, тому пристрої повинні забезпечувати стабільну передачу телеметрії та підтримку OTA.

1.2.3. Blynk.

Blynk є хмарною платформою, що має доволі зручний і візуально зрозумілий мобільний додаток та веб-інтерфейс. Вона дає змогу швидко створювати інтерфейси керування пристроями у вигляді кнопок, слайдерів, діаграм тощо. Blynk орієнтована на звичайних користувачів та ентузіастів і часто використовується для створення прототипів та невеликих проєктів. Але, вона має деякі обмеження в гнучкості налаштувань і вимагати платної підписки для розширених функцій.

Пристрої на базі мікроконтролерів сімейства *ESP* ідеально підходять для Blynk, оскільки вони легко інтегрується, і мають достатньо ресурсів для взаємодії з сервером у режимі реального часу. З більш простіших варіантів є можливіть використання мікроконтролерів сімейства *Arduino* з *ESP модулем* — бюджетне рішення для навчання або простих проєктів.

Особливість Blynk-пристроїв мінімальні вимоги до ресурсів, але залежність від постійного інтернет-з'єднання та хмарної авторизації через токени.

Blynk є однією з найзручніших платформ у контексті інтерфейсу керування, орієнтованою саме на користувача: Мобільний додаток *Blynk iOS* та *Android* головний інтерфейс. Користувачі можуть створювати панелі керування зі слайдерами, кнопками, графіками та індикаторами без програмування. Веб-додаток *Blynk.Console* дає змогу керувати пристроями з ПК, моніторити дані, створювати шаблони. *Push-сповіщення* дозволяють автоматизувати взаємодію з пристроями без сторонніх сервісів.

1.2.4. Home Assistant.

Home Assistant надає веб-інтерфейс із повною підтримкою локального розгортання та широкими можливостями кастомізації. Він дозволяє створювати складні автоматизації, персоналізовані панелі керування та інтеграції з різноманітними пристроями. Проте конфігурація системи часто потребує редагування YAML-файлів, що є складним для початківців. Також Home Assistant дозволяє працювати локально без хмари, тому до нього підключаються більшість пристрої з підтримкою локальних протоколів.

Готові пристрої типу *Shelly, Sonoff, Tuya* — ідеально інтегруються з Home Assistant через MQTT або локальні API. *Zigbee-пристрої* теж гарно під'єднуються — через Zigbee2MQTT або ZHA, підключаються через USB-шлюзи. Інтузіасти теж можуть під'єднувати пристрої на базі *ESP32/ESP8266 із* прошивкою *ESPHome* — дозволяє гнучко налаштовувати будь-які кастомні пристрої, сенсори, реле, індикатори.

Особливістю Home Assistant є повна автономність від хмари, можливість автоматизації, локальна інтеграція зі сценаріями та голосовими помічниками (Google Assistant, Alexa).

Також Home Assistant має потужний і зручний веб-інтерфейс (Dashboard), який також адаптований під мобільні пристрої: *Lovelace UI* — модульна система побудови панелей керування. Дозволяє створювати інтерактивні карти кімнат, графіки, кнопки. Для мобільних додатків *iOS ma Android* є синхронізація з основним сервером, має повний доступ до управління пристроями. Є інтеграція з голосовим помічником *Google Assistant, Alexa*, яка здійснюється через Home Assistant Cloud або налаштування локального доступу. Автоматизації через *YAML* або *UI* дозволяють створювати сценарії, які виконуються за подіями або тригерами.

1.3. Висновок аналізу існуючих рішень

Платформа	Архітектура	Спрямування	Складність використання
Amazon AWS IoT Core	Централізована, хмарна	Комерційна	Складна, орієнтована на розробників
Microsoft Azure IoT	Централізована, хмарна	Комерційна	Складна, вимагає знань DevOps
Blynk	Централізована (частково локальна)	Freemium	Зручна, але обмежена
Home Assistant	Локальна (можлива хмара)	Відкрита	Гнучка, але потребує багато налаштувань

Таблиця 1.1 Зведена таблиця з характеристиками існуючих IoT систем

У висновок можна скати, що існуючі системи управління IoT-пристроями демонструють значний прогрес та різноманіття, проте більшість із них є централізованими або суттєво залежні від центрального вузла керування. Це створює попит для подальших досліджень у напрямку мультицентричних, розподілених систем, які забезпечуватимуть: вищу стійкість, гнучкість у керуванні, зменшення залежності від хмари, можливість незалежного керування з кількох терміналів.

Окрім цього, попри багатофункціональність і потужність таких платформ, як AWS IoT, Azure IoT Hub, Home Assistant і Blynk, наразі відсутня справді проста та універсальна система з принципом *plug-and-play* — коли пристрій автоматично розпізнається системою, підключається до мережі та стає готовим до використання без додаткових технічних налаштувань. Усі наявні рішення вимагають хоча б базових знань у сфері мережевих технологій. Це створює явний бар’єр для широкомасштабного впровадження IoT у побутовій сфері серед простих користувачів. Таким чином, існує чітка потреба у створенні стандартизованої та інтуїтивної системи керування пристроями, яка могла б забезпечити зручність і водночас просту у використанні.

2. ПОСТАНОВКА ТЕХНІЧНОГО ЗАВДАННЯ І ВИБІР ЗАСОБІВ

Розробка починається з чіткого розуміння задач, які має виконувати система, розуміння побудови її архітектура та вибору засобів її реалізації. Головною задачею IoT системи звісно є моніторинг та керування під'єднаними пристроями. Система має бути реалізована такий чином, щоб користувач міг швидко та легко додавати нові пристрої та зручно керувати їми.

Оскільки ми розглядаємо створення мультицентричної системи моніторингу та керування IoT пристроями. Вона має декілька рівноправних пристроїв через, які можна здійснювати контроль над пристроями у мережі, тобто декілька терміналів керування мають рівні права доступу до пристроїв. Архітектура такої системи має бути побудована таким чином, щоб наявні термінали могли відправляти запити до під'єднаних пристроїв. Пристрій же в свою чергу має прийняти запит, обробити його, здійснити відповідну дію та відобразити результат її виконання. Побачивши результат термінал відобразить його користувачу. Результат може бути, як спрацювання дії так і не спрацювання. Так користувач матиме розуміння роботи під'єднаних пристроїв.

Тепер знаючи, що має робити система контролю IoT пристроїв виберемо засоби її реалізації. Термінал керування пристроями це девайс з встановленим спеціальним програмним забезпеченням. Саме з цього програмного забезпечення буде відбуватися відправка запитів підконтрольнім пристроям. Програмне забезпечення складається з двох частин це фронтенд та бекенд. Вибір засобу створення програмного забезпечення в першу чергу залежить від платформи терміналу керування.

Найуніверсальніший варіант – це Flutter, він має найбільшу кросплатформеність та підходить для пристроїв платформ: Android, iOS, Windows, Linux та macOS. Flutter також має можливість створення інтерактивного GUI, графіків, відео, звуку и має підтримку Bluetooth та MQTT.

Kotlin для Android та Swift iOS. Kotlin — офіційна мова розробки Android-додатків, має власне середовище розробки та підтримується Google. Має

повноцінний доступ до Android API: Bluetooth, Wi-Fi, камери та сенсорів. Підтримує багато бібліотек Bluetooth та MQTT. Swift — офіційна мова програмування від Apple, також має власну середу розробки та фреймворки для створення UI. Має простий синтаксис та доступ до сенсорів, Bluetooth, камери та навіть геолокації.

PyQt для Windows, Linux та macOS. PyQt — модуль мови програмування Python, використовується для створення кросплатформених GUI-додатків. Дає змогу будувати повноцінні віконні програми. Мова програмування Python має багато бібліотек та модулів для роботи з Bluetooth, MQTT, HTTP та WebSocket. Також має підтримку роботи з IoT пристроями.

Розібравши варіанти для створення програмного забезпечення, розглянемо способі реалізації підконтрольних IoT пристроїв. IoT пристрій також складається з двох частин це апаратне та програмне забезпечення. Апаратне забезпечення складається з головного контролера, сенсорів та механічних модулів. У якості головного контролера можна обрати мікроконтролери сімейства ESP, STM чи міні-комп’ютери сімейств Raspberry Pi, LattePanda, тощо. Програмне забезпечення залежить від обраного апаратного забезпечення.

Для виконання простих, конкретних задач: зчитування даних з сенсорів, керування реле, передачей даних по Bluetooth або Wi-Fi, краще використовувати мікроконтролери. Вони не мають операційної системи і програмуються зазвичай мовами програмування C, C++ чи MicroPython. Мікроконтролери є гарним вибором для створення недорогих, енергоефективних IoT-пристроїв, які працюють автономно або надсилаючи дані на зовнішній сервер чи шлюз.

Міні-комп’ютери натомість є повноцінними комп’ютерами на одній платі. Вони працюють під управлінням Linux і дозволяють запускати багатозадачні програми, сервіси, інтерфейси, бази даних, а також обробляти відео та графіку. Використовуються для більш складних задач, які не здатен виконувати мікроконтролер.

2.1. Визначення загальних вимог до системи

Оскільки, одною з головних цілей мультицентричної системи є відмова від центрального модуля керування. Основна логіка роботи, обробка даних та прийняття рішень керуються безпосередньо IoT-пристроям в подальшому модулям, що мають діяти автономно. Це дозволяє суттєво зменшити навантаження на центральний інтерфейс, мінімізувати ризики пов’язані з відмовами центрального вузла та в цілому забезпечити гнучкість системи.

Для досягнення цього необхідно зробити наступні кроки у проєктуванні IoT-системи: типізація пристроїв, уніфіковані команди, автономність у прийнятті рішень, стандартизована система передачі даних та останнє, але не менш важливе просте підключення до мережі.

Типізація пристроїв: усі модули повинні бути класифіковані та поділені за типом по функціоналу. Наприклад, автоматичні жалюзі, автополив рослин, розумне освітлення, вентиляція тощо. Це дозволить стандартизувати їх, спростить додавання нових модулів у систему та зробить керування простішим. Створивши чітку структуру, в якій кожен тип модулів має передбачувану поведінку та набір параметрів. Завдяки цьому можна заздалегідь визначити логіку обробки подій для кожної категорії, не змінюючи архітектуру при розширенні чи оновленні системи. Наприклад, модулі типу «освітлення» працюють за спільними шаблонами, незалежно від того, чи це лампа, світлодіодна стрічка або зовнішній прожектор. Програмний інтерфейс буде взаємодіяти не з конкретними пристроями, а з абстрактними типами дозволяючи створювати сценарії автоматизації. Крім того, типізація створює основу для майбутньої аналізу даних. Дані, зібрані від пристроїв одного типу, можна групувати, порівнювати та аналізувати для виявлення закономірностей чи проблем.

Уніфіковані команди: кожен тип модулю повинен мати власний набір стандартизованих команд спрацювання або відповіді. Це є продовженням стандартизації пристроїв підводячи їх до простого застосування за принципом plug-and-play, де не потрібно кожного разу налаштовувати велику кількість унікальних

налаштувань для нового модулю. Наявність чітко визначених команд дозволяє забезпечити однакову взаємодію між різними пристроями. Наприклад, модулі освітлення можуть підтримувати спільні команди ввімкнення, вимкнення, зміни яскравості чи кольору, а модулі клімат-контролю — команди зміни температури, активації вентиляції або підтримки режиму енергозбереження. Завдяки цьому будь-який новий модуль, доданий у систему, миттєво вписується у загальну логіку керування без потреби в розробці окремих драйверів чи адаптерів. Уніфікованість команд також спрощує автоматизацію сценаріїв та взаємодію між пристроями. Наприклад, модуль сенсору руху може надсилати універсальний сигнал активації, який можуть обробити будь-які інші модулі, які мають реагувати на цю подію — вмикати світло, активувати камери або навіть виклик поліції. Це забезпечує гнучкість системи, яка дозволить будувати складні сценарії роботи IoT-молулів з простих компонентів, без постійного переписування коду.

Автономність у прийнятті рішень: кожен IoT-модуль повинен мати можливість самостійно виконувати задану логіку дій у відповідь на зовнішні або внутрішні сигнали. Наприклад, гідропонна система вирощування рослин має працювати самостійно. Термінал керування потрібен лише для внесення коректив у роботі модулів чи їх ручного керування. Такий підхід дозволяє системі не лише продовжувати роботу у випадку втрати зв’язку з центром керування, а й оперативніше реагувати на зміни середовища. Наприклад, модуль температурного контролю при виявлені перегріву може одразу активувати охолодження, без необхідності чекати команди ззовні. Це особливо важливо в критичних або віддалених умовах, де стабільність інтернет-з’єднання не гарантована. Крім того, розподілення логіки дозволяє масштабувати систему без створення додаткового навантаження на термінали керування. Кожен новий модуль додається як незалежна пристрій з власним набором правил і логіки, що знижує ризик конфліктів і збоїв.

Підключення до мережі: кожен модуль має підтримувати спосіб швидкого та безпечного підключення до мережі, включно з налаштуванням Wi-Fi, Ethernet або іншого каналу зв’язку по типу Bluetooth чи BLE. Для зручності користувача процес

приєднання повинен бути максимально простим. В ідеалі, модуль при першому запуску повинен самостійно перейти в режим ініціалізації, сигналізуючи про свою готовність до з’єднання через індикатор або простий веб-інтерфейс. Користувач, зі свого боку, має мати змогу підключитися до пристрою без потреби в спеціалізованому програмному забезпеченні — наприклад, через стандартний браузер або мобільний додаток. Після підключення пристрій повинен автоматично виявляти доступні мережі, запитувати облікові дані та завершувати конфігурацію без додаткових технічних дій.

Створення системи на основі вище зазначених параметрів дозволить розподілити головну логіку між IoT-модулями, що відкриває шлях до побудови дійсно адаптивної, масштабованої та відмовостійкої інфраструктури, де складна технічна архітектура прихована за простим і зрозумілим для користувача інтерфейсом.

Характеристика	Опис реалізації в системі
Типізація пристроїв	Усі пристрої класифікуються за функціональним призначенням: освітлення, вентиляція, автополив, жалюзі тощо. Це спрощує стандартизацію команд, керування та додавання нових модулів.
Уніфіковані команди	Кожен тип пристрою має фіксований набір команд, що відповідають його функціоналу
Автономність у прийнятті рішень	Модулі працюють незалежно від центрального вузла, виконуючи локальні сценарії дій. Наприклад, система автополиву активується за показником вологості без участі користувача. Але користувач може змінити налаштування з сенсора на полив по таймеру
Просте підключення до мережі	Кожен модуль підтримує автоматизоване підключення до мережі (Wi-Fi, Ethernet, Bluetooth/BLE). Налаштування зводиться до вказання адреси та паролю без складних конфігурацій.

Таблиця 1.2 Зведена таблиця з характеристиками для IoT модулів.

2.2 Визначення загальних вимог до графічного інтерфейсу.

Оскільки однією з основних цілей мультицентричної системи є надання можливості пристроям працювати автономно без централізованого контролю, виникає потреба у створенні простого, але надійного способу взаємодії з ними у випадках, коли потрібне ручне втручання. Такий спосіб реалізується у вигляді графічного інтерфейсу термінала керування, який дозволяє швидко підключатись до пристрою, передавати команди, переглядати поточний стан і, за потреби, вносити зміни. Основне завдання цього інтерфейсу — не перевантажувати користувача складними налаштуваннями, а навпаки — забезпечити зрозумілу і стабільну роботу з різними типами IoT-модулів.

Головне вікно інтерфейсу: головне вікно виконує функцію панелі спостереження, де в реальному часі відображаються всі під’єднані модулі. Всі модулі мають відображатися простими та зрозумілими елементами, які дозволять користувачу швидко та просто орієнтуватися в інтерфейсі. Кожен модуль має мати власну назву, поруч з назвою має показується його тип та статус.

Пошук та фільтри: у межах головного екрана інтерфейсу термінала має бути реалізовано не лише базове відображення підключених пристроїв, а й додаткові інструменти для полегшення навігації в системах з великою кількістю під’єднаних модулів. Одним із ключових елементів має бути пошуковий рядок, який дозволить за назвою швидко знаходити необхідні модулі. Окрім пошуку, інтерфейс також включає систему фільтрації, яка дозволяє групувати пристрої за типом чи групою. Наприклад, користувач може вибрати відображення лише сенсорів температури, лише виконавчих модулів або ж побачити всі модулі, що належать до певної групи наприклад, "Кабінет 234" чи "Поверх 5". Це особливо корисно, коли мережа включає десятки IoT-модулів з різними функціями.

Створення груп пристроїв: у межах головного екрану також має бути кнопки створення та видалення груп модулів. Кнопка створення групи відкриватиме просте вікно, де користувач зможе вказати назву нової групи. Після підтвердження група з’являється в загальному списку, і до неї можна буде додавати будь-які

пристрої. Функція видалення групи має бути доступна через відповідну кнопку. При натиснені на кнопку має відкриватися прости вікно з наявними групами, в якому користувач зможе вибрати групи для видалення. Це особливо корисно у великих мережах, де модулі розділяються за приміщеннями чи напрямками роботи.

Журналу подій: дії користувача, відповіді від модулів чи помилки системи мають зберігаються у зручному вигляді. Тому головне вікно також має містити кнопку журналу подій. При натисненні на неї має відкриватися вікно зі зведеною таблицею подій, в ній має бути вказаний час запису, опис записаної події та елемент з яким відбулася подія. Це дозволить при потребі повернутися до історії подій, проаналізувати її та знайти причину збою.

Додавання нових модулів: для додавання нових модулів на головному вікні передбачено окрему кнопку додавання нового модуля. Вона відкриває вікно, у якому користувач може ввести основні параметри модулю, який підключається до системи. Для спрощення процесу у вікні користовачу має бути потрібно лише ввести назву модуля, його ір адресу, тип та групу до якої пристрій буде віднесений. Залежно від типу модулю має додаватися рядок кількість пристроїв. Це необхідно для пристроїв типу автоматичних жалюзей, коли вікон які треба закривати може бути декілька. Після заповнення форми нового модуля він має з’явитися у головному вікні та стати доступним для моніторингу і керування.

Моніторинг та керування модулями: після вибору іконки модуля у головному вікні має відкриватися вікно контролю, яке автоматично формує набір інструментів для взаємодії в залежності від функціоналу вибраного модуля. Від простого зчитування даних до надсилання команд чи налаштувань. Це дозволить просто та без затримок перемикатися між модулями.

У підсумку, реалізований графічний інтерфейс термінала керування відіграє ключову роль у забезпеченні зручності, надійності та гнучкості мультицентричної IoT-системи. Завдяки продуманій структурі головного вікна користувач має змогу швидко орієнтуватися в мережі пристроїв, використовувати пошук і фільтри для точного вибору потрібного модуля, а також впорядковувати пристрої за логічними групами. Інструменти для створення та видалення груп забезпечують підтримку

структурованого підходу до організації мережі, що є особливо важливим у системах з великою кількістю елементів.

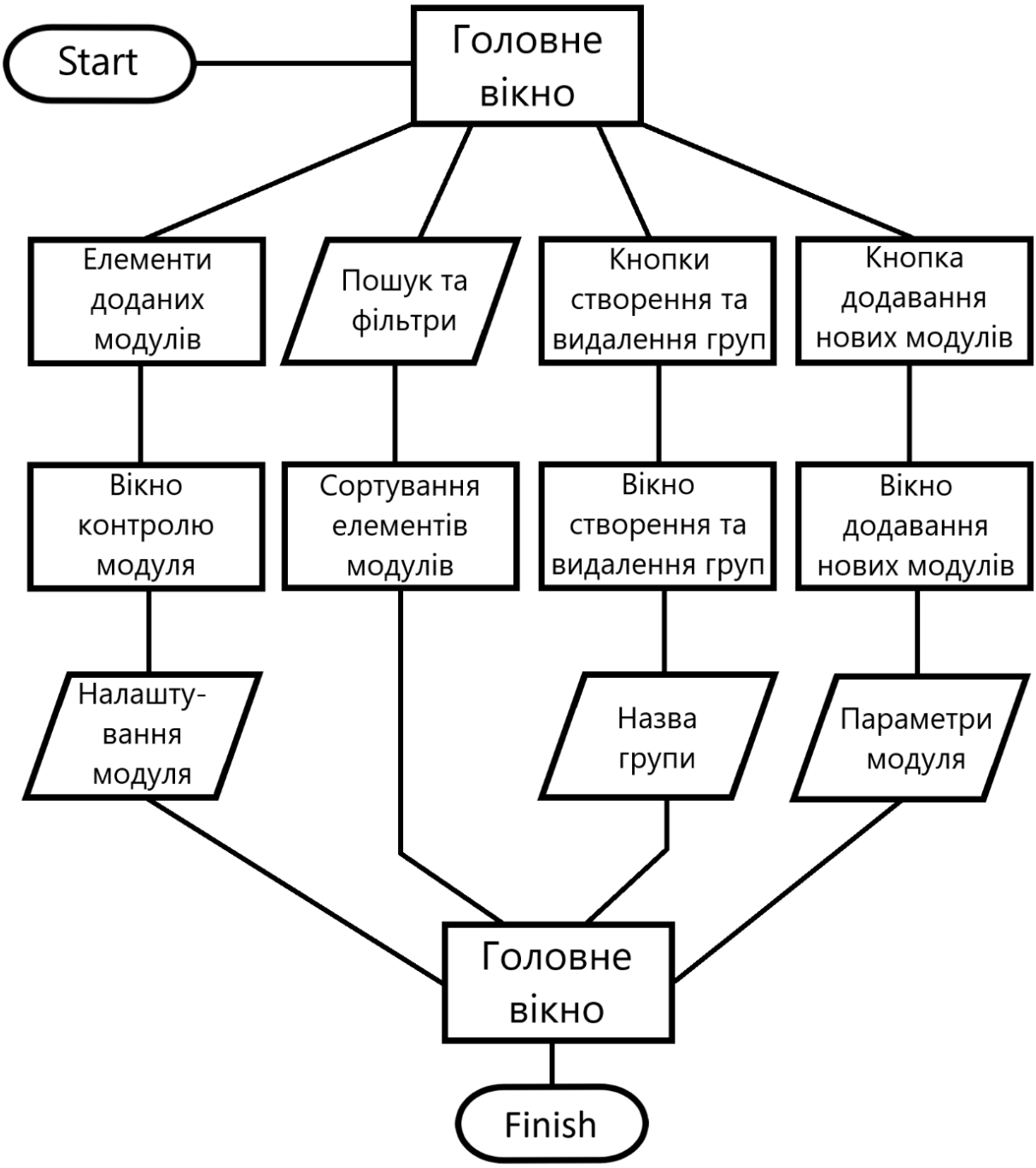


Рис. 2.1. блок-схема графічного інтерфейсу термінала керування.

2.3. Визначення загальних вимог до IoT-модулів

IoT-пристрої, що використовуються у системі мультицентричного керування, повинні забезпечувати стабільну автономну роботу, з можливістю ручного керування через спеціально розроблений інтерфейс. Такий підхід дозволить використовувати пристрої як незалежні модулі, що самостійно виконують задані функції, так і як елементи централізованого управління. Пристрої повинні бути здатні до ефективної комунікації з терміналом керування за допомогою надійного мережевого протоколу, що забезпечить точну передачу даних.

Загальні функціональні вимоги: IoT-пристрої, що входять до складу системи мультицентричного керування, повинні забезпечувати стабільну та надійну функціональність в умовах автономного виконання завдань. Кожен пристрій має бути здатним працювати незалежно від керуючого термінала, виконуючи основні функції відповідно до закладеної програми.

Взаємодія з терміналом керування: Забезпечення ефективної комунікації між IoT-пристроями та терміналом керування є ключовим чинником стабільного функціонування системи мультицентричного керування. Передача даних повинна здійснюватися з використанням надійних мережевих протоколів, що забезпечують мінімальні затримки, високу пропускну здатність, а також захист від втрати або пошкодження інформації. Доцільним є використання таких протоколів, як MQTT, TCP/IP або подібних, які адаптовані до роботи в умовах обмежених ресурсів та нестабільного мережевого середовища, що є типовим для IoT-систем. IoT-модулі повинні мати стандартизований програмний інтерфейс API, який дозволить проводити уніфікований обмін повідомленнями між пристроями та терміналом. Такий API повинен забезпечувати однозначне трактування команд управління, зворотну передачу даних про стан пристрою, та, за необхідності, підтримку подій, що ініціюються самим пристроєм. Наприклад, аварійні сповіщення або повідомлення про досягнення порогових значень. Обмін даними має здійснюватися

у заздалегідь визначеному форматі, що дозволяє уникнути неоднозначностей під час обробки запитів або відповідей.

Апаратне забезмечення IoT-модулів: Апаратне забезпечення IoT-пристроїв, що функціонують у складі системи мультицентричного керування, має відповідати низці вимог, спрямованих на забезпечення стабільної, енергоефективної та гнучкої роботи. Основною вимогою до апаратної платформи є підтримка виконання програмного забезпечення, що обумовлює необхідність використання мікроконтролерів чи міні-комп'ютерів із достатніми обчислювальними ресурсами та об'ємом оперативної пам'яті. Модулі повинні мати набір цифрових та аналогових входів/виходів: GPIO, ADC, PWM тощо, для взаємодії з сенсорами, виконавчими елементами або периферійними пристроями. Залежно від задач модуля наявність інтерфейсів зв'язку типу UART, I2C чи SPI може бути обов'язковою для забезпечення розширення функціоналу та підключення додаткових модулів. Таких як модулі пам'яті, дисплеї, мережеві адаптери чи інші спеціалізовані компоненти. Особливу увагу слід приділяти енергоспоживанню пристроїв, особливо у випадках автономного живлення від батарей чи акумуляторів. Мікроконтролер повинен підтримувати режими енергозбереження, які дозволяють зменшити споживання енергії у періоди простою або очікування подій. Конструктивне рішення має також передбачати можливість швидкого виходу з режиму сну для оперативного реагування на зовнішні сигнали або команди керування. Апаратна реалізація повинна бути достатньо компактною та модульною, щоб забезпечити легкість інтеграції пристроїв у різні середовища експлуатації. У разі потреби, пристрої повинні бути захищені від впливу факторів зовнішнього середовища.

2.4. Вибір апаратних засобів реалізації

Для реалізації системи мультицентричного керування потрібно обрати апараті засоби, здатні забезпечити стабільну роботу IoT-модулів в умовах автономного функціонування та взаємодії з центральним терміналом керування.

Основним критерієм вибору апаратної платформи була наявність вбудованих засобів бездротової комунікації, достатня обчислювальна потужність, підтримка необхідних периферійних інтерфейсів, а також енергоефективність.

ESP32: у якості апаратної основи IoT-пристроїв було обрано мікроконтролери серії ESP32, які відзначаються високою продуктивністю при збереженні компактних розмірів та низького енергоспоживання. ESP32 оснащений двоядерним процесором, інтегрованими модулями Wi-Fi та Bluetooth, що дає змогу створювати пристрої з повною підтримкою бездротової взаємодії без потреби в додаткових комунікаційних компонентах. Наявність великої кількості цифрових та аналогових входів та виходів дозволяє підключати широкий спектр датчиків і виконавчих елементів. Для реалізації вимог гнучкості та розширюваності система повинна підтримувати підключення додаткових модулів через стандартні апаратні інтерфейси, такі як UART, I2C, SPI, що також доступні на платформах ESP32. Завдяки цьому до складу пристрою можуть бути легко інтегровані дисплеї, реле, сенсори температури, вологості, освітлення, руху тощо, залежно від функціонального призначення кожного модуля. Крім того, ESP32 підтримує режими енергозбереження, що дозволяє оптимізувати витрати енергії у пристроях з автономним живленням. Такий підхід особливо актуальний у випадках використання акумуляторів або джерел енергії з обмеженою ємністю. Конструктивно пристрої можуть бути реалізовані у компактному форматі з можливістю монтажу в умовах обмеженого простору, а при необхідності — у захищених корпусах, стійких до впливу зовнішніх факторів таких як пил, волога, температурні зміни.

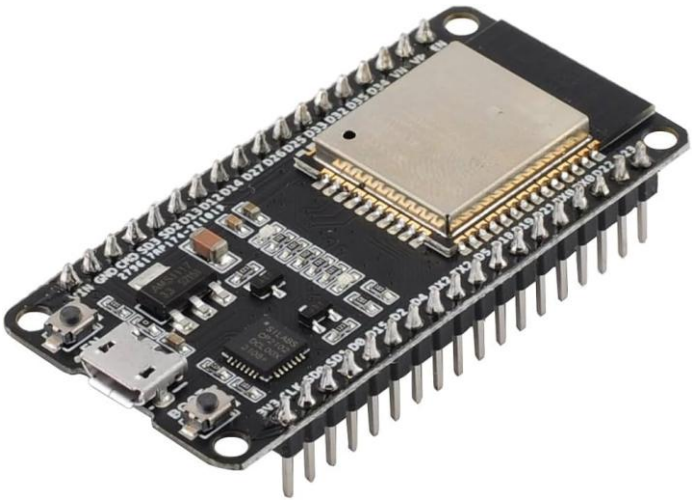


Рис. 2.2. Мікроконтролер ESP32.

STM32: окрім ESP32, як альтернативну апаратну платформу також було розглянуто мікроконтролери серії STM32, що належать до сімейства ARM Cortex-M виробництва компанії STMicroelectronics. Ці контролери вирізняються широким модельним рядом, який дозволяє підібрати оптимальний варіант відповідно до вимог конкретного пристрою — від надзвичайно компактних та енергоефективних моделей до високопродуктивних з розширеними можливостями обробки даних. Мікроконтролери STM32 підтримують велику кількість периферійних інтерфейсів, включаючи ADC, DAC, SPI, I2C, UART, CAN, USB та інші, що робить їх надзвичайно гнучкими у проєктуванні складних пристроїв з широким спектром функцій. Важливою перевагою STM32 є наявність апаратних модулів криптографії (у певних серіях), що дає змогу реалізовувати засоби захисту на апаратному рівні без суттєвого навантаження на основний процесор. Ще однією сильною стороною STM32 є висока точність аналогових вимірювань, що робить їх доцільними для проєктів, де пріоритетом є якість зчитування з аналогових сенсорів, наприклад, у вимірювальних або медичних пристроях. Крім того, пристрої STM32 добре масштабуються у складних системах, особливо у випадках, коли необхідна чітка багаторівнева структура комунікації між підсистемами. Однак, на відміну від ESP32, мікроконтролери STM32 не мають вбудованих засобів бездротового зв’язку. У разі потреби Wi-Fi або Bluetooth з’єднання необхідно додатково інтегрувати відповідні модулі ESP8266, HC-05, модулі LoRa тощо, що ускладнює

конструкцію пристрою, збільшує його габарити та загальну вартість реалізації. У результаті, STM32 є доцільним вибором у тих випадках, коли ключовими є точність, надійність, розширена обробка даних, а також високий ступінь захищеності. У той же час, для компактних, мобільних IoT-рішень, що вимагають вбудованого бездротового зв'язку, більш доцільною залишається платформа ESP32.

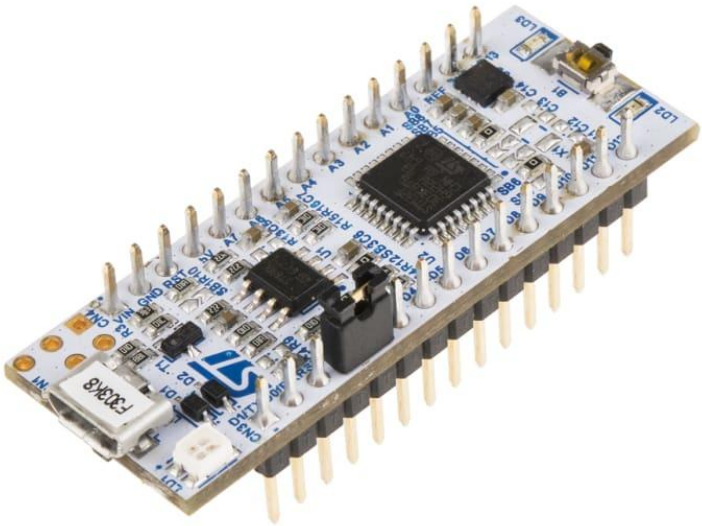


Рис. 2.3. Мікроконтролер STM32.

Raspberry Pi: ще однією альтернативною платформою для реалізації IoT-пристроїв є одноплатний комп'ютер Raspberry Pi, зокрема моделі Raspberry Pi 3, 4, та 5 які поєднують у собі повноцінну обчислювальну систему на базі Linux та вбудовані засоби бездротового зв'язку. Raspberry Pi забезпечує набагато вищу обчислювальну потужність у порівнянні з мікроконтролерами ESP32 та STM32, завдяки чому може використовуватись у системах, де необхідна обробка зображень, робота з відео, машинне навчання або складна логіка взаємодії між пристроями. Завдяки наявності повноцінної операційної системи, Raspberry Pi OS, дана платформа забезпечує доступ до широкого спектру програмного забезпечення, бібліотек, інструментів відлагодження та мережевих сервісів. Це значно спрощує інтеграцію з хмарними сервісами, базами даних, а також дозволяє реалізовувати локальні веб-сервери, шлюзи або вузли обробки даних всередині

IoT-системи. Raspberry Pi має вбудовані модулі Wi-Fi та Bluetooth, а також широкий набір апаратних інтерфейсів GPIO, I2C, SPI, UART, що дозволяє підключати зовнішні сенсори, реле, камери, дисплеї та інші периферійні пристрої. Проте, у порівнянні з мікроконтролерами, ця платформа має й низку обмежень, зокрема — вищий рівень енергоспоживання, що робить її менш придатною для автономних систем із живленням від батарей. Також запуск повноцінної ОС потребує більшого часу завантаження та вимагає відносно складнішого адміністрування. Raspberry Pi доцільно використовувати у якості центрального вузла обробки в системі мультицентричного керування — наприклад, для збору даних від кількох IoT-пристроїв, прийняття рішень на основі аналізу або забезпечення інтерфейсу взаємодії з користувачем. Водночас для периферійних виконавчих або сенсорних модулів доцільніше застосовувати мікроконтролери ESP32 або STM32, які є енергоефективнішими і простішими в апаратній реалізації.

Рис. 2.4. Міні-комп’ютер Raspberry Pi 5.

2.5. Вибір мови програмування.

Вибір мови програмування для реалізації компонентів системи мультицентричного керування здійснювався з урахуванням особливостей цільових

пристроїв, вимог до ефективності, простоти розробки, підтримки апаратного рівня та подальшої масштабованості. У даному проєкті застосовано комбінований підхід із використанням мов C++ та Python, кожна з яких відіграє важливу роль на різних рівнях архітектури системи. Для розробки прошивки IoT-пристроїв основним інструментом виступає мова **C++**, яка є традиційною для мікроконтролерного програмування. Вона забезпечує низькорівневий доступ до апаратних ресурсів, високу продуктивність, ефективне управління пам'яттю, а також сумісність із широким спектром драйверів та бібліотек, необхідних для роботи з периферією. Завдяки цьому C++ є оптимальним вибором для реалізації критичних за часом реакції компонентів, таких як обробка сигналів або взаємодія з сенсорами в реальному часі.

Водночас, для спрощення та пришвидшення розробки, а також у навчальних або прототипних цілях, можливо використовувати MicroPython — полегшену версію Python, адаптовану для роботи на мікроконтролерах, зокрема ESP32. MicroPython дозволяє писати зрозумілий, компактний код, зменшуючи поріг входу для розробників-початківців, а також прискорює тестування функціоналу. Він підтримує роботу з GPIO, PWM, I2C, SPI, а також має бібліотеки для роботи з мережевими інтерфейсами, що дозволяє реалізовувати повноцінні IoT-пристрої без необхідності писати низькорівневий код. Керуючий термінал, призначений для моніторингу та ручного управління IoT-пристроями, реалізовано мовою Python, яка завдяки своїй універсальності та багатству бібліотек ідеально підходить для розробки графічних інтерфейсів, мережевої логіки та інтеграції з базами даних чи хмарними сервісами. У цьому проєкті для створення GUI використано бібліотеку PyQt5, яка забезпечує створення функціонального, сучасного та кросплатформенного інтерфейсу користувача. Таким чином, використання як C++, так і Python/MicroPython у різних частинах системи дозволяє гнучко поєднувати продуктивність та низькорівневий контроль з швидкістю розробки та зручністю у створенні інтерфейсів, що є особливо важливим у багаторівневих IoT-системах.

2.6. Вибір середовища розробки.

Середовище розробки програмного забезпечення є одним із ключових інструментів у процесі створення, налагодження та підтримки системи мультицентричного керування IoT-модулями. Вибір конкретного середовища визначався такими критеріями, як зручність написання коду, підтримка цільової апаратної платформи, наявність засобів відлагодження, керування бібліотеками та інтеграція з системами контролю версій.

Середа розробки для програмування IoT-модулів: У розробці програмного забезпечення для мікроконтролерів системи мультицентричного керування було прийнято рішення використовувати PlatformIO — кросплатформенне середовище розробки, спеціально оптимізоване для проєктування вбудованих систем. Цей інструмент працює у зв'язці з редактором Visual Studio Code і надає розробникам доступ до широкого спектру функцій, що істотно підвищують продуктивність розробки. Однією з ключових переваг PlatformIO є його інтегрована система керування бібліотеками, яка дозволяє автоматизовано підключати зовнішні залежності без ручного завантаження або конфігурування. Крім того, середовище підтримує багатопроєктну структуру, що дає можливість паралельно розробляти прошивки для різних пристроїв у межах єдиного проєкту. Це особливо важливо в умовах мультикомпонентної IoT-системи, де кожен пристрій виконує унікальну функцію. PlatformIO забезпечує глибоку інтеграцію з системами контролю версій по типу Git, автоматичну генерацію конфігурацій прошивок, а також можливість завантаження коду до пристрою безпосередньо з інтерфейсу середовища. Наявність розширених можливостей налагодження, підтримки тестування, а також гнучка система налаштування середовища розробки роблять його зручним інструментом для реалізації комплексних рішень на основі мікроконтролерів ESP32, STM32 та інших. Таким чином, використання PlatformIO дозволяє розробляти ефективне, масштабоване та структуровано організоване вбудоване програмне забезпечення, що відповідає сучасним стандартам інженерного проєктування в IoT-сфері.

Паралельно з основним середовищем розробки, для задач швидкого тестування функціоналу пристроїв та початкового створення прототипів використовувалося Arduino IDE — мінімалістичне, але широко популярне середовище для розробки мікроконтролерного ПЗ. Arduino IDE добре підходить для створення компактних програм, де ключовими є оперативність розгортання, простота інтерфейсу та доступність бібліотек. Arduino IDE підтримує мікроконтролери ESP32 через інсталяцію відповідного набору плат, що дозволяє програмувати їх так само легко, як і традиційні контролери серії AVR. Простіше кажучи, це середовище одразу готове до використання: воно не потребує складної конфігурації та дозволяє миттєво завантажити код на пристрій. Ця властивість робить Arduino IDE надзвичайно привабливим для оперативної перевірки працездатності сенсорів, модулів зв'язку або окремих фрагментів алгоритмів. Хоча Arduino IDE поступається PlatformIO в плані масштабованості, структурованості проєктів і розширених функцій налагодження, воно зберігає популярність серед як початківців, так і досвідчених розробників — передусім через інтуїтивно зрозумілий інтерфейс і величезну екосистему готових прикладів. Отже, Arduino IDE доцільно використовувати у тих випадках, коли необхідно швидко розгорнути базову прошивку, провести первинне тестування пристрою або підготувати демонстраційний прототип із мінімальними витратами часу на налаштування середовища.

Середа розробки для програмування графічного інтерфейсу термінала керування: для створення інтерфейсу керування для системи мультицентричного керування IoT-пристроями було обрано середовище розробки PyCharm, що спеціалізується на програмуванні мовою Python. Вибір даного інструменту був зумовлений низкою переваг, які забезпечують ефективну розробку, структурування коду та зручне налагодження клієнтських застосунків. Інтерфейс керування розроблявся з використанням бібліотеки PyQt5, яка дозволяє створювати функціональні та адаптивні графічні інтерфейси. PyCharm надає повну інтеграцію з цією бібліотекою, що забезпечує зручне середовище для побудови структури проєкту, автоматичну генерацію коду за допомогою дизайнерів форм, а

також підтримку інтелектуального автодоповнення та перевірки синтаксису. Окрім того, PyCharm дозволяє легко керувати віртуальними середовищами виконання venv, що є важливим для організації залежностей та забезпечення переносимості проєкту. Середовище також забезпечує інтеграцію з системами контролю версій зокрема, Git, що є важливим для підтримки проєкту у командній розробці або при довготривалому впровадженні оновлень. Засоби рефакторингу, налагодження та запуску окремих модулів у візуальному режимі значно пришвидшують процес розробки і дозволяють швидко виявляти та усувати логічні помилки. Особливо важливою перевагою є можливість створення структурованого проєкту, у якому логіка керування пристроями, мережевий обмін даними, обробка сигналів та графічне представлення функцій інтерфейсу реалізуються у вигляді окремих модулів. Це підвищує читабельність коду, спрощує модифікацію та подальше розширення функціоналу системи. Таким чином, використання PyCharm у поєднанні з PyQt5 дозволило реалізувати гнучкий, зручний у використанні та масштабований інтерфейс керування для системи IoT-пристроїв, що повністю відповідає вимогам до сучасних графічних клієнтських застосунків.

3. ОПИС ФУНКЦІОНУВАННЯ ПЛАТФОРМИ ТА ТЕСТУВАННЯ

Даний розділ надає повну характеристику та огляд розробленої системи мультицентричного керування IoT-пристроями. У межах цього розділу детально розглядаються ключові компоненти системи, зокрема апаратне та програмне забезпечення IoT-модулів, архітектура взаємодії між пристроями, а також функціональні можливості термінала керування, який забезпечує зручний доступ до контролю та моніторингу. Особлива увага приділяється аспектам зручності користування, масштабованості рішення, стабільності комунікації між елементами та придатності системи до розширення.

Крім того, у цьому розділі буде наведено результати тестування, яке проводилося з метою перевірки працездатності системи у реальних умовах експлуатації. У якості прикладного сценарію реалізації обрано гідропонну систему вирощування рослин, яка дозволяє продемонструвати практичне застосування мультицентричної моделі керування, ефективність автоматизованого контролю параметрів середовища таких як вологість повітря, температура повітря, рівень освітлення, pH розчину тощо. Для початку розглянемо структурну схему функцій, які виконує систеиа.

3.1. Карта системи та порядок роботи з нею.

Робота системи мультицентричного керування починається з додавання нового IoT-модулю до термінала керування. Користувач відкриває графічний інтерфейс системи й переходить у вікно додавання нового пристрою. У цьому вікні необхідно ввести базові параметри пристрою: його назву, мережеву адресу, тип пристрою, а також групу, до якої він належить. Після введення цих даних система зберігає конфігурацію та ініціює підключення до відповідного IoT-модуля.

Після успішного додавання пристрою, користувач може взаємодіяти з ним через інтерфейс термінала керування. Зокрема, в режимі ручного управління надається можливість надсилати команди на вмикання або вимикання конкретних

функцій — наприклад, активувати насос для подачі води, увімкнути світло або запустити вентиляцію. Надіслані з термінала команди передаються до пристрою по локальній мережі WiFi. IoT-модуль, отримавши команду, її обробляє, виконує відповідну дію й формує відповідь про статус виконання, яка відображається у відповідному вікні інтерфейсу.

Крім прямого управління, пристрій постійно передає на термінал дані з вбудованих сенсорів. У випадку гідропонної системи це, зокрема, температура повітря, температура розчину, рівень вологості та інші показники, залежно від конфігурації. Ці дані збираються у реальному часі, і термінал відображає їх у вигляді числових значень.

Таким чином, користувач має змогу оперативно слідкувати за станом середовища, отримувати зворотний зв’язок щодо виконаних команд, а також реагувати на будь-які відхилення від заданих параметрів.

Завдяки цій структурі система забезпечує повноцінну, гнучку взаємодію між людиною й технічними компонентами.

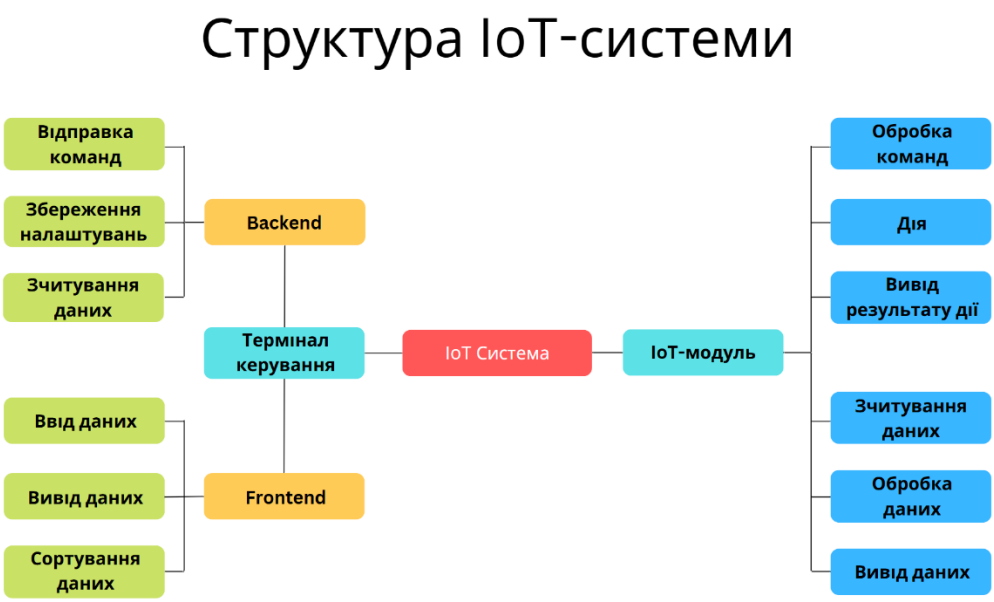


Рис. 3.1. Структурна схема функцій IoT-системи.

3.1.1. Карта роботи з інтерфейсом термінала керування

Головна сторінка, як і в більшості програмних систем з графічним інтерфейсом, виконує функцію стартової точки взаємодії користувача з програмним забезпеченням. У контексті розробленої системи мультицентричного керування IoT-пристроями головна сторінка є першим елементом, з якого починається робота з терміналом керування. На цій сторінці реалізовано зручну навігацію до основних розділів — зокрема, додавання нового пристрою, перегляд підключених модулів, перехід до керування гідропонною системою. На малюнку 3.2. представлено зовнішній вигляд головної сторінки інтерфейсу розробленої системи.

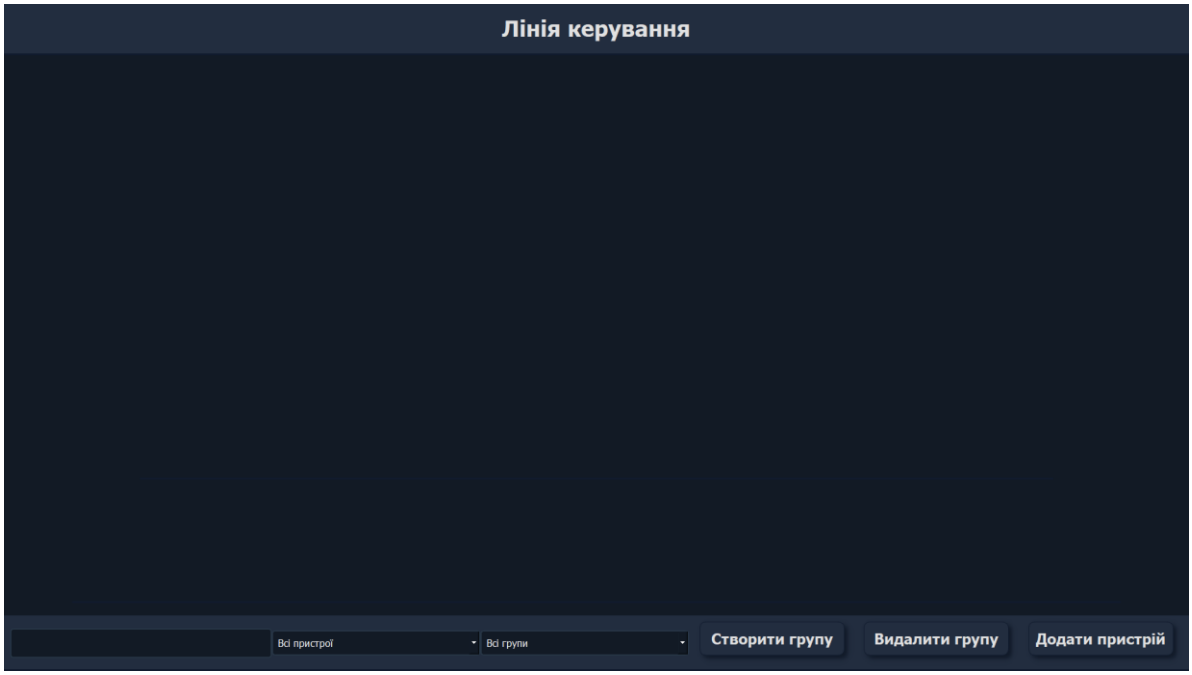


Рис. 3.1.1. Головне вікно графічного інтерфейсу термінала керування.

Після ознайомлення з головною сторінкою інтерфейсу користувач переходить до наступного етапу — організації пристроїв за допомогою створення груп. У інтерфейсі термінала керування передбачено кнопку «Створити групу», натискання якої відкриває відповідне діалогове вікно для введення параметрів нової групи. Цей функціонал дозволяє структурувати велику кількість

підключених модулів відповідно до їх призначення, розташування або вирощуваної культури, що значно спрощує подальшу навігацію та керування системою. У діалоговому вікні користувач вводить назву групи, наприклад «Томати», яка відображатиметься в інтерфейсі як окрема категорія. Група «Томати» у цьому випадку може об’єднувати один або кілька гідропонних модулів, що відповідають за вирощування помідорів. Після підтвердження введених даних група зберігається в структурі системи та з’являється у списку для сортування модулів.

Надалі користувач може додавати нові пристрої безпосередньо до створеної групи. Завдяки цьому інтерфейс стає більш впорядкованим: усі дії — як додавання, так і керування — виконуються в межах конкретної групи, що дозволяє чітко розмежувати модулі за функціональним або логічним принципом. Такий підхід забезпечує зручність при роботі з великою кількістю пристроїв і дозволяє швидко ідентифікувати потрібні модулі в інтерфейсі термінала. На малюнку 3.1.2. представлено зовнішній вигляд вікна створення групи.

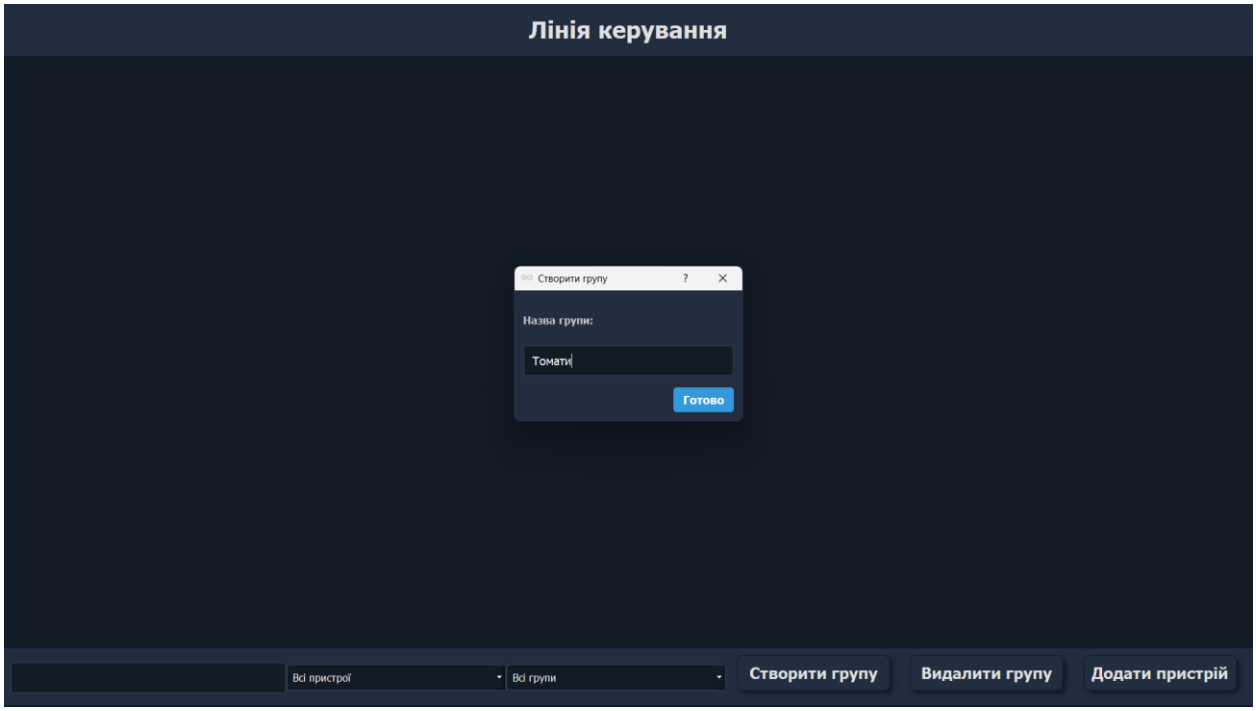


Рис. 3.1.2. Вікно створення групи пристроїв.

Наступним кроком після створення групи є додавання нового пристрою, який буде безпосередньо взаємодіяти з системою керування. У графічному інтерфейсі термінала для цього передбачено окрему кнопку — «Додати пристрій». Натискання на неї відкриває спеціальне діалогове вікно, у якому користувач вводить необхідні дані для реєстрації пристрою в системі.

У вікні додавання пристрою обов’язково вказується назва пристрою — унікальний або описовий ідентифікатор, який дозволяє легко розпізнати пристрій у списку. Далі вводиться мережева адреса, яка є ключовим параметром для встановлення зв’язку з фізичним модулем. Також користувач обирає тип пристрою зі списку доступних, в нашому випадку це гідропонна система. Останнім кроком є вибір групи, до якої буде віднесено пристрій. Це може бути раніше створена група «Томати», або можна взагалі не прив’язувати пристрій до групи. На малюнку 3.1.3. представлено зовнішній вигляд вікна додавання пристроїв.

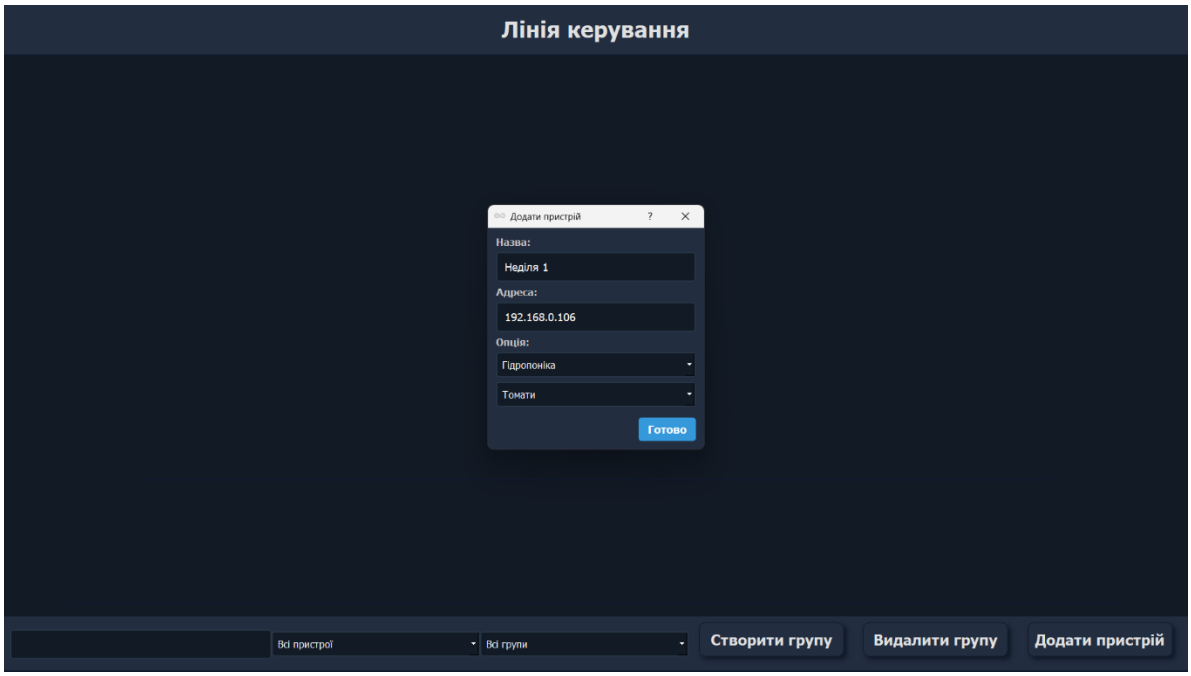


Рис. 3.1.3. Вікно додавання пристроїв.

Після заповнення усіх полів і підтвердження дії, пристрій зберігається в системі, автоматично додається до обраної групи, і стає доступним для керування через інтерфейс термінала. Завдяки такому підходу забезпечується логічна і

технічна прив’язка між елементами системи, що дозволяє централізовано керувати великою кількістю модулів із чітким розмежуванням за призначенням або місцем застосування. На малюнку 3.1.4. представлено зовнішній вигляд головного вікна з доданим пристроєм.

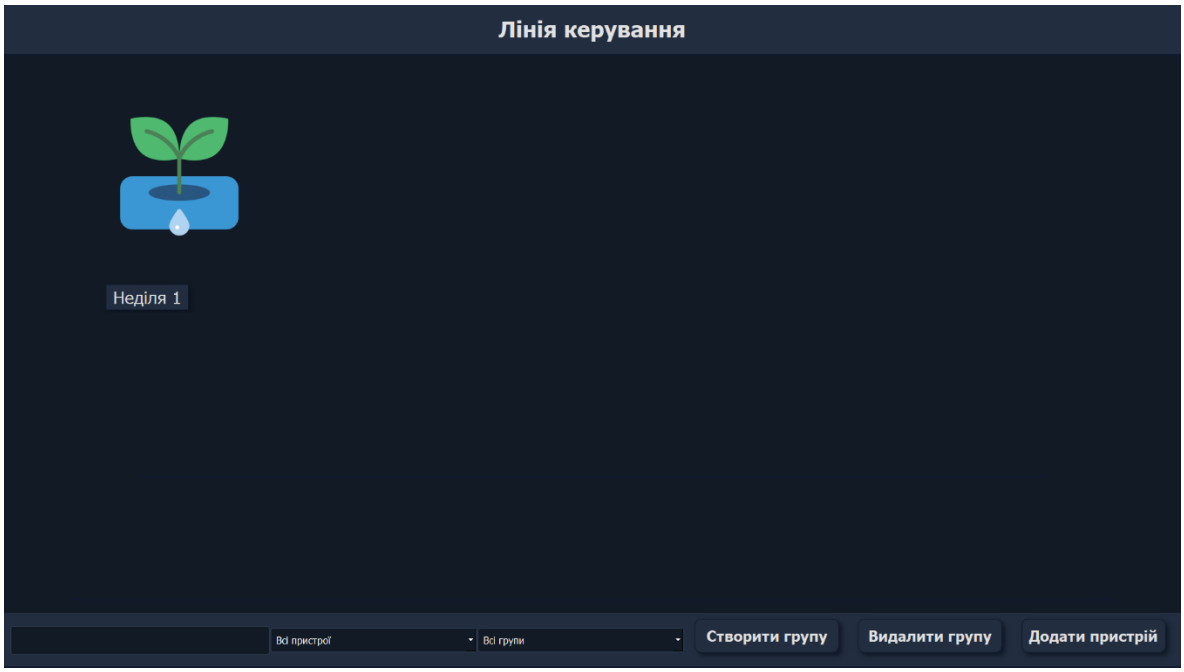


Рис. 3.1.4. Головне вікно з доданим пристроєм.

Після додавання пристрою до системи та його прив’язки до відповідної групи, користувач отримує можливість повноцінно взаємодіяти з ним через інтерфейс термінала. Для цього достатньо натиснути на іконку пристрою у головному вікні. У результаті відкривається вікно контролю пристрою, що надає доступ до повного набору функцій керування та моніторингу.

Оскільки тип обраного пристрою — гідропонна система, інтерфейс вікна адаптовано під характерні функціональні компоненти такого модуля. У верхній частині відображаються назва пристрою та його мережева адреса, що дозволяє користувачу швидко ідентифікувати пристрій, особливо при наявності великої кількості підключених модулів.

Центральна частина вікна містить елементи керування, які дозволяють вручну активувати або зупинити окремі складові гідропонної установки. Зокрема,

реалізовані кнопки для керування роботою насосу, освітлення та вентиляції. Кожна з цих кнопок виконує миттєве надсилання відповідної команди до модуля, після чого користувач отримує зворотний зв’язок про результат виконання. Окрім індивідуального керування, у вікні передбачено також дві функціональні кнопки загального призначення: одна для повного запуску всіх основних компонентів (насос, світло, вентиляція), інша — для їх одночасного вимкнення. Це дає змогу швидко перевести пристрій у робочий або неактивний стан без необхідності поетапного керування кожною системою окремо.

Частина вікна відведена для відображення даних з сенсорів, підключених до гідропонної установки. У режимі реального часу користувач бачить актуальні значення температури повітря, вологи повітря, температури, кислотності (pH) та солоності (EC) води. Дані виводяться у зручному форматі, що дозволяє миттєво оцінювати стан середовища та приймати рішення щодо втручання у роботу системи.

Завдяки такій структурі, вікно контролю пристрою поєднує в собі як засоби прямого керування, так і інструменти моніторингу, що забезпечує повноцінну взаємодію з гідропонним модулем у межах мультицентричної системи керування. На малюнку 3.1.5. представлено зовнішній вигляд вікна контролю доданим пристроєм.

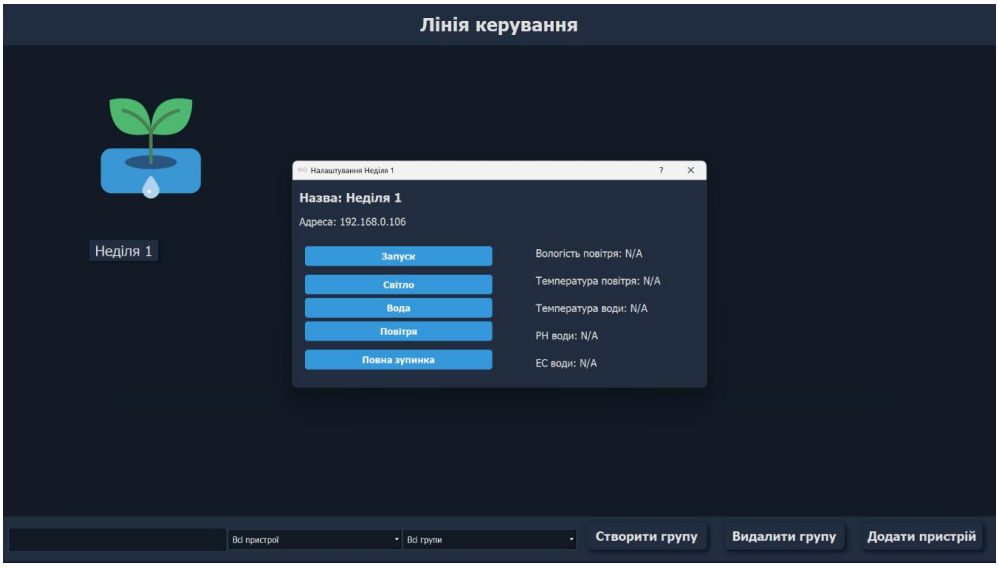


Рис. 3.1.5. Вікно контролю доданого модуля.

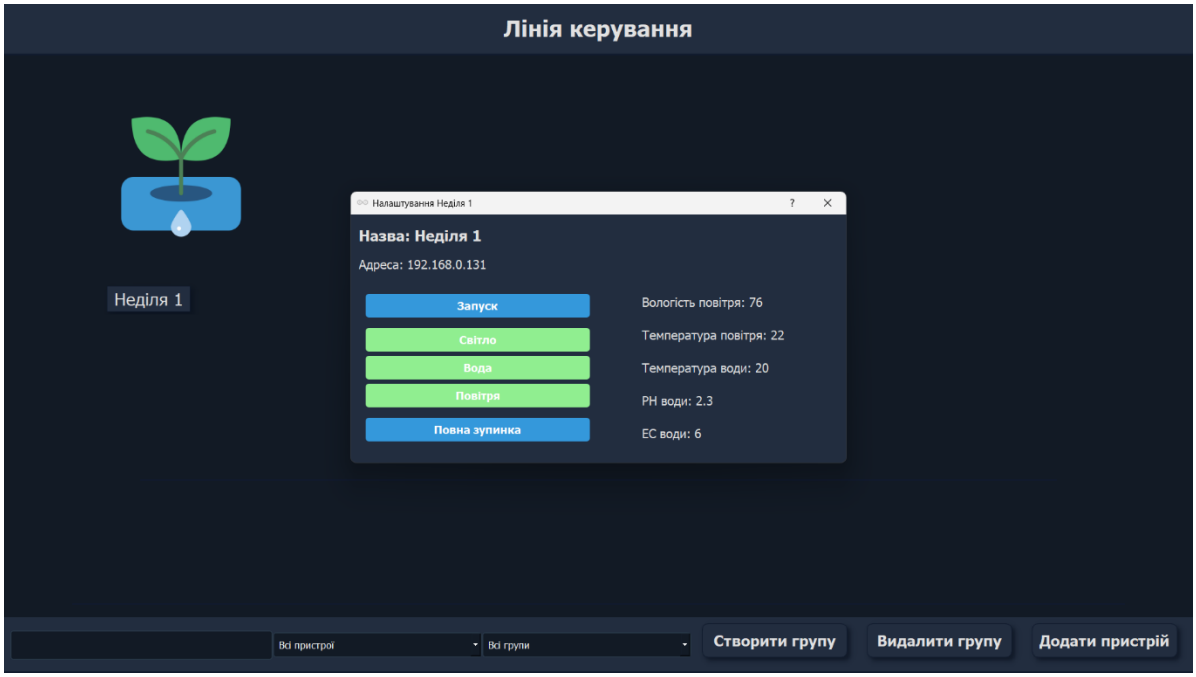


Рис. 3.1.6. Відображення виконаної дії у вікні контролю модулю.

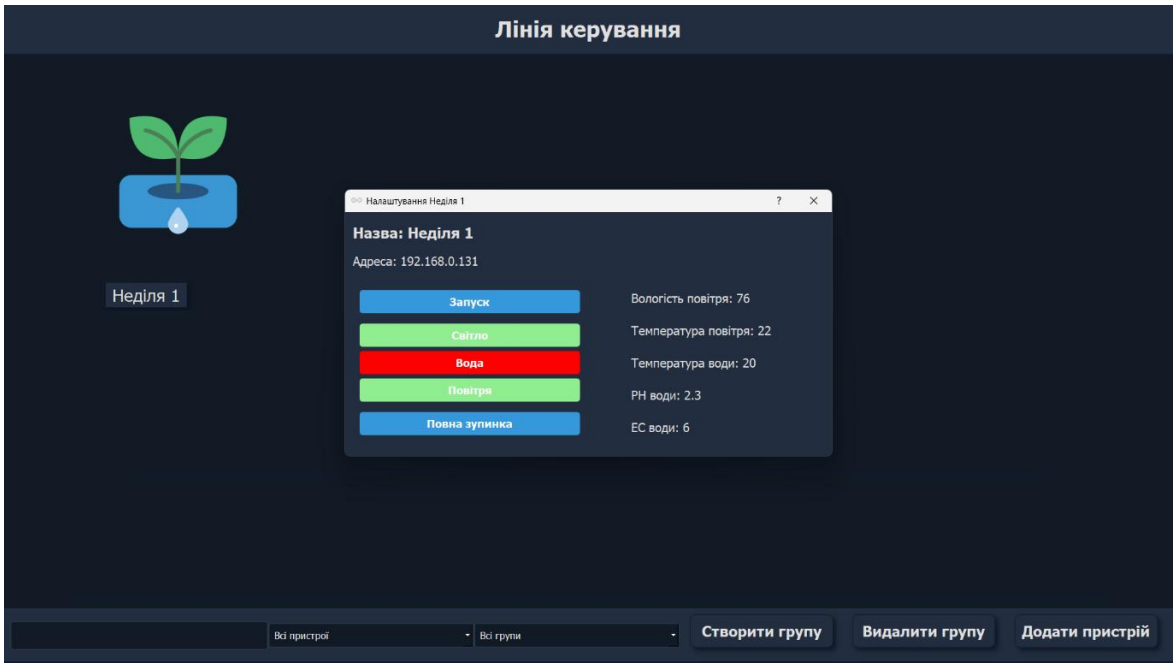


Рис. 3.1.7. Відображення виконаної дії у вікні контролю модулю.

У процесі розширення системи та підключення великої кількості пристроїв, користувачу надається можливість швидко орієнтуватися в інтерфейсі за допомогою функції пошуку. Відповідне текстове поле розташоване у верхній частині головного вікна термінала керування. Пошук здійснюється за назвою

пристрою або частиною введеного тексту, а результати автоматично оновлюються у міру введення символів. Це дозволяє без зволікань знайти конкретний модуль серед десятків або сотень підключених елементів.

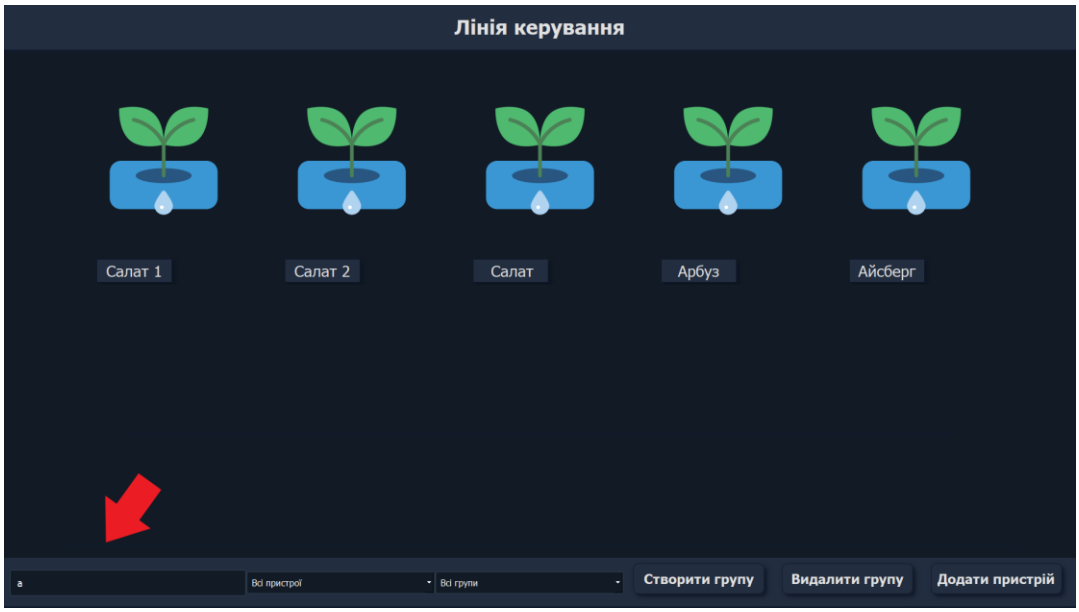


Рис. 3.1.8. Рядок пошуку головного вікна.

Додатково інтерфейс підтримує сортування пристроїв за типом. Кожному модулю під час додавання до системи призначається тип, який відображає його функціональне призначення. За допомогою відповідного фільтра користувач може відобразити лише ті пристрої, які належать до певного типу, що значно спрощує навігацію, коли йдеться про керування великою кількістю технічно різнорідних модулів.

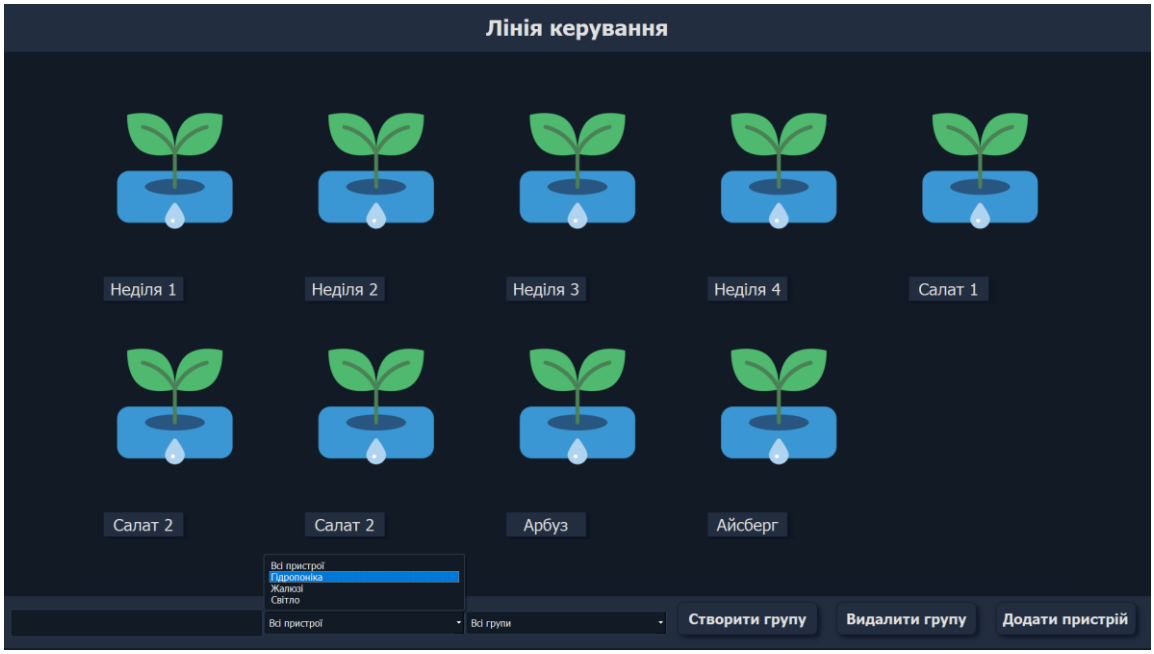


Рис. 3.1.9. Фільтр за типом модуля головного вікна.

Окрім цього, реалізовано можливість сортування пристроїв за групами. Усі пристрої, додані до певної логічної групи. Наприклад, «Томати», «Огірки», «Теплиця 1», можуть бути відфільтровані й відображені окремо. Це дозволяє працювати з конкретною культурою або об’єктом, не відволікаючись на інші елементи системи. Такий підхід забезпечує не лише зручність у щоденному використанні, а й логічну структуризацію пристроїв у масштабованій системі.

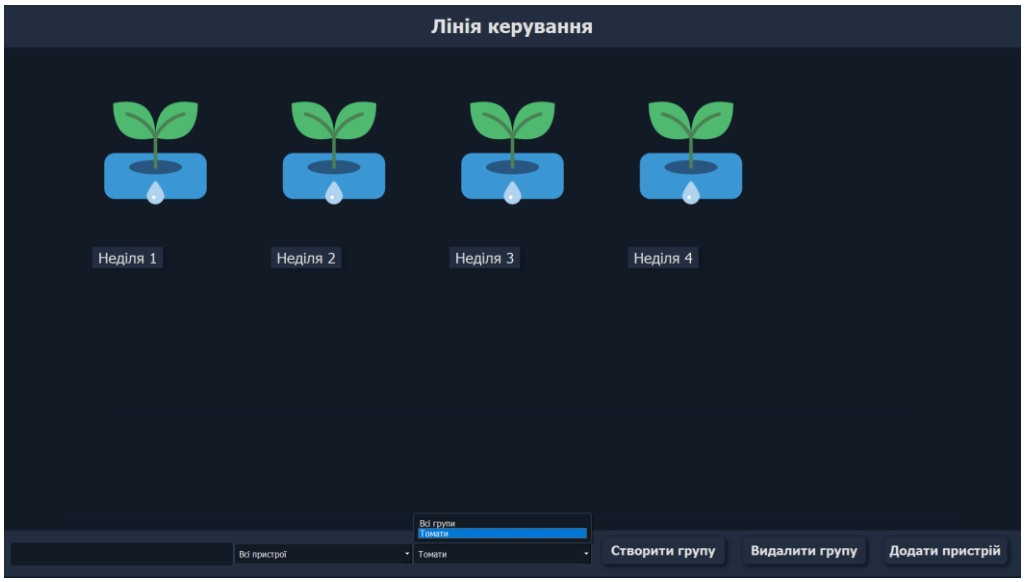


Рис. 3.1.10. Фільтр за групою модуля головного вікна.

У разі, якщо користувач бажає видалити одну з наявних груп пристроїв, для цього в головному вікні термінала керування передбачено відповідну функціональну кнопку. Натискання на неї відкриває вікно видалення груп, у якому відображається повний список усіх створених груп, присутніх у системі. Кожна група подана у вигляді окремого елемента списку з зазначенням її назви, що дає змогу легко ідентифікувати потрібну. Користувач може обрати одну або кілька груп для видалення. Видалена група автоматично зникає з інтерфейсу.

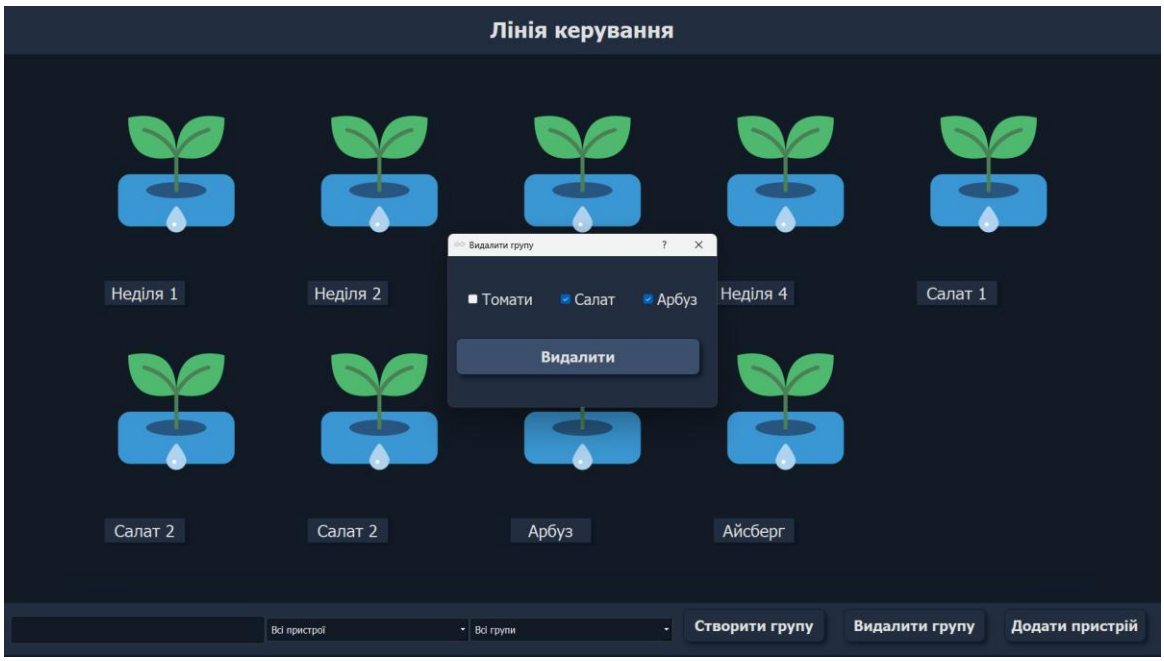


Рис. 3.1.11. Вікно видалення груп модулів.

Таким чином, інтерфейс термінала керування системою мультицентричного керування IoT-пристроями реалізовано з урахуванням зручності, гнучкості та масштабованості. Починаючи з головної сторінки, користувач має доступ до основних функцій — створення груп, додавання нових пристроїв, пошуку та сортування модулів. Інтуїтивна структура дозволяє ефективно організовувати роботу з великою кількістю пристроїв, розподіляти їх за логічними групами, керувати кожним модулем окремо або у складі групи. Наявність динамічного вікна контролю, яке адаптується до стану підключення пристрою та виводить як кнопки управління, так і сенсорні дані в реальному часі, забезпечує повний цикл взаємодії користувача з системою. У сукупності ці можливості створюють просте та

функціонально повне середовище керування, що відповідає сучасним вимогам до управління розподіленими IoT-системами.

3.1.3. Карта роботи апаратного забезпечення модулю IoT-системи.

Апаратне забезпечення IoT-модуля, що є складовою частиною мультицентричної системи керування, побудоване на основі мікроконтролера ESP32 з підтримкою бездротової комунікації, та включає у свій склад комплекс сенсорів і виконавчих елементів. Кожен модуль функціонує як автономний вузол, здатний самостійно зчитувати дані з сенсорів, приймати керуючі команди з термінала та здійснювати відповідні дії.

ESP32: у якості центрального обчислювального елементу IoT-модуля було обрано мікроконтролер ESP32, розроблений компанією Espressif Systems. Цей мікроконтролер є сучасним, енергоефективним і багатофункціональним рішенням, спеціально орієнтованим на використання в інтернеті речей, автоматизованих системах керування та вбудованих рішеннях, що потребують як продуктивності, так і бездротової комунікації.

Однією з ключових переваг ESP32 є наявність двоядерного 32-бітного процесора Tensilica Xtensa LX6, який працює на частоті до 240 МГц. Це дозволяє виконувати паралельні задачі — наприклад, одночасну обробку сенсорних даних та підтримку мережевого з’єднання. Крім того, ESP32 має вбудовані модулі Wi-Fi та Bluetooth, що усуває необхідність у додаткових бездротових модулях і значно спрощує апаратну реалізацію IoT-пристрою. Контролер підтримує широкий набір інтерфейсів: GPIO, ADC, DAC, SPI, I2C, UART, PWM, що забезпечує гнучкість у підключенні сенсорів, виконавчих пристроїв та периферії. Наприклад, для гідропонного модуля на базі ESP32 можна безпосередньо підключати температурні сенсори, датчики вологості, реле керування насосом чи освітленням, а також дисплеї або індикатори. Ще однією важливою перевагою ESP32 є підтримка режимів енергозбереження, що дозволяє мінімізувати споживання енергії у випадках автономного живлення. Це особливо актуально для систем, які працюють

у віддалених або обмежених за ресурсами умовах. У режимі глибокого сну ESP32 може споживати менше ніж 10 мкА, що робить його конкурентоспроможним рішенням для автономних IoT-модулів. Також слід зазначити високу популярність ESP32 серед розробників, активну підтримку спільноти та доступність великої кількості бібліотек, прикладів коду та документації. Це значно пришвидшує процес розробки, тестування та інтеграції пристрою в загальну систему.

Таким чином, вибір ESP32 як основи для IoT-модуля є технічно обґрунтованим з огляду на поєднання високої продуктивності, гнучких інтерфейсів, бездротового зв'язку та енергоефективності, що робить його оптимальним для побудови масштабованої IoT-системи.

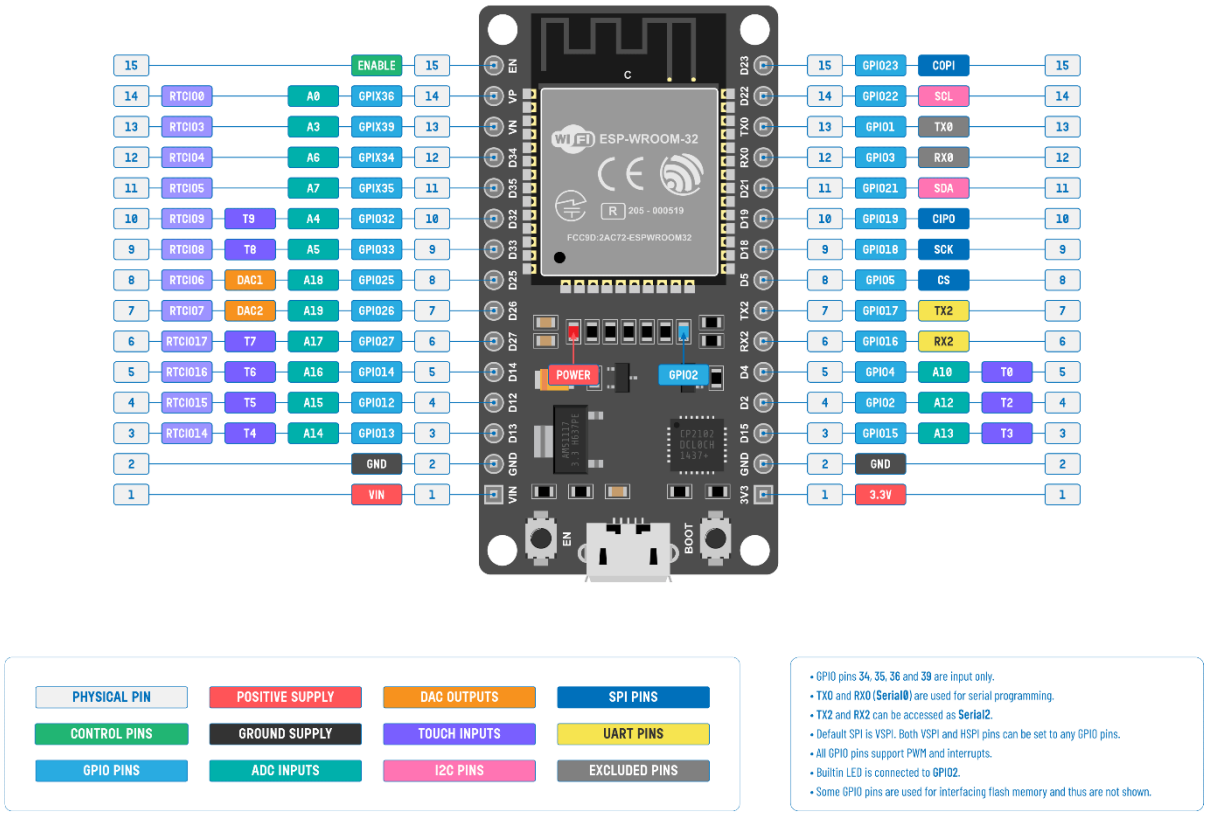


Рис. 3.2.1. Розпіновка мікроконтролера ESP32.

DHT11: для вимірювання таких ключових параметрів навколишнього середовища, як температура повітря та відносна вологість, у складі IoT-модуля

використано сенсор DHT11. Цей датчик є недорогим, енергоефективним і широко застосовується в системах автоматизованого моніторингу та управління, зокрема в проектах на базі мікроконтролерів ESP32. Сенсор DHT11 поєднує в собі цифровий датчик температури та ємнісний датчик вологості, а також має вбудований АЦП і мікроконтролер, що дозволяє здійснювати зчитування даних у цифровому форматі. Обмін із основним контролером (у цьому випадку ESP32) відбувається через однопровідний цифровий інтерфейс, що мінімізує кількість необхідних з'єднань і спрощує підключення.

Типові характеристики DHT11 включають діапазон вимірювання температури від 0 до +50 °C з точністю ±2 °C та діапазон вимірювання вологості від 20 до 90 % з точністю ±5 %. Хоча ці параметри є базовими порівняно з більш дорогими аналогами, як DHT22 або BME280, DHT11 є цілком достатнім для побутових, навчальних і низькобюджетних проєктів, таких як гідропонні установки початкового рівня або тестові IoT-системи.

Завдяки компактним розмірам, низькому енергоспоживанню та широкій підтримці у програмному середовищі DHT11 був обраний як оптимальний варіант для базового контролю кліматичних параметрів у рамках даного прикладу IoT-системи.

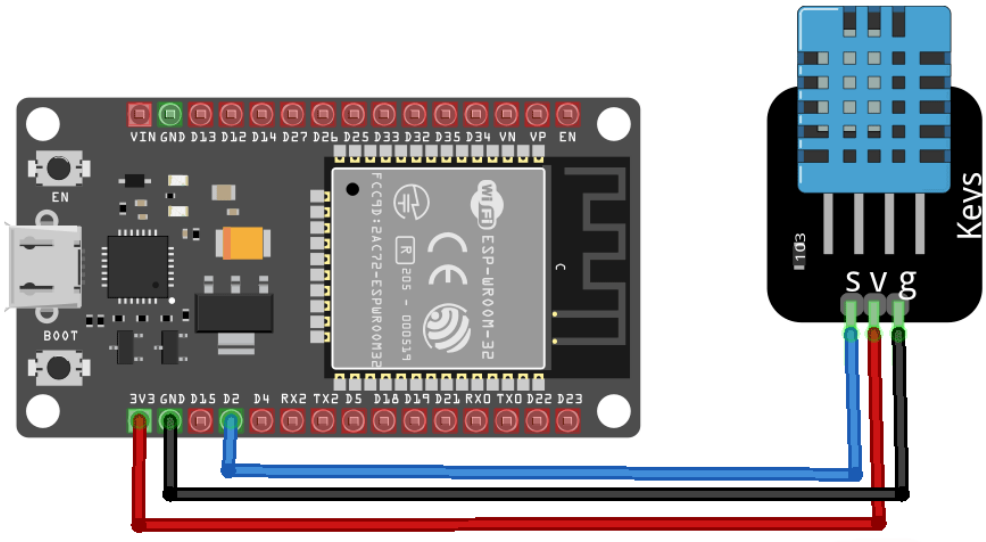


Рис. 3.2.1. Схема підключення сенсора DHT11 до ESP32.

DS18B20: для вимірювання температури живильного розчину у гідропонній системі до складу IoT-модуля включено цифровий температурний сенсор DS18B20 у водонепроникному виконанні. Цей датчик є популярним рішенням для точного температурного моніторингу в умовах підвищеної вологості, занурення у рідину або агресивного середовища, що робить його ідеальним варіантом для контролю температури води в сільськогосподарських IoT-системах. DS18B20 забезпечує високу точність вимірювань у робочому діапазоні від –55 до +125 °C та має роздільну здатність до 12 біт, яка може бути налаштована програмно. Датчик передає дані у цифровому форматі за допомогою однопровідного інтерфейсу, що дозволяє підключати кілька таких сенсорів до одного піну мікроконтролера, використовуючи лише один сигнальний провід. Це значно спрощує електричну схему пристрою та зменшує кількість необхідних з'єднань.

Важливою особливістю DS18B20 є його водозахищений корпус, який дозволяє занурення у рідину без додаткових заходів ізоляції. Саме завдяки цій властивості датчик широко використовується у системах поливу, акваріумних контролерах, теплицях та, зокрема, у гідропонних установках, де точне вимірювання температури розчину є критично важливим для підтримання стабільного середовища для росту рослин.

Завдяки цифровому інтерфейсу, стійкості до зовнішніх умов та високій точності, DS18B20 був обраний як основний сенсор для контролю температури води у складі IoT-модуля гідропонної системи. Його використання дозволяє забезпечити надійність і довготривалу експлуатацію навіть у складних умовах.

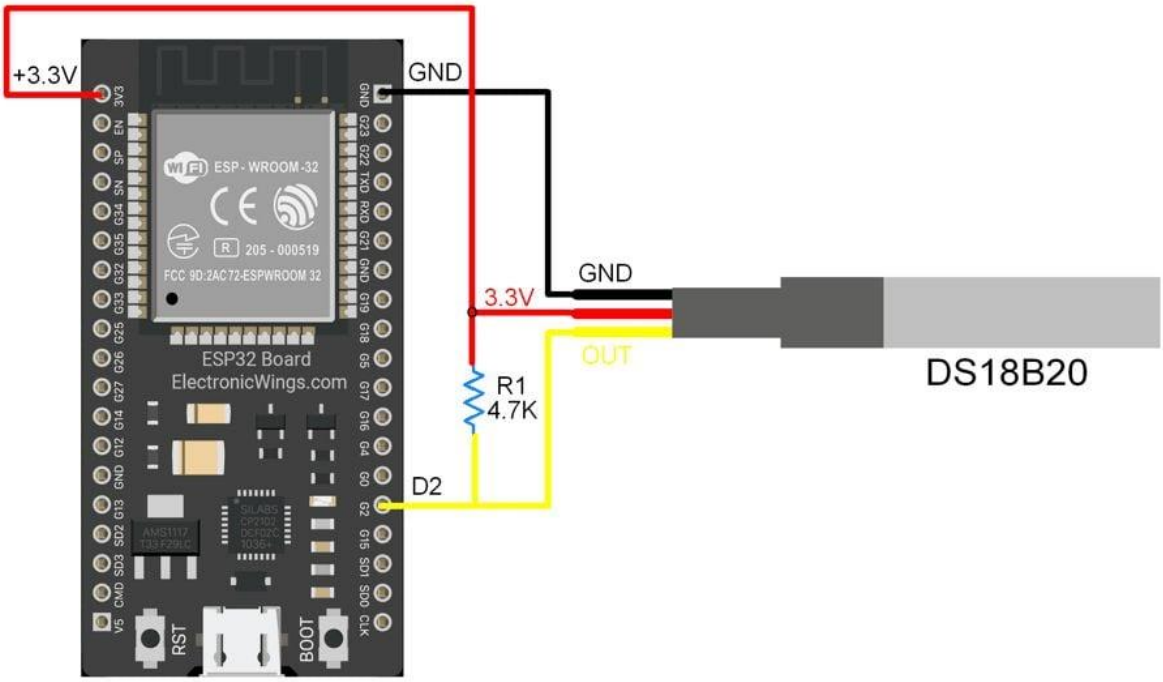


Рис. 3.2.2. Схема підключення сенсора DS18B20 до ESP32.

Сенсор кислотності: для контролю хімічного складу живильного середовища у гідропонній системі до складу IoT-модуля включено сенсор кислотності pH. Контроль pH є критично важливим у процесі вирощування рослин у безґрунтових умовах, оскільки від рівня кислотності залежить доступність мінералів і мікроелементів, засвоюваних кореневою системою. Сенсор кислотності pH складається зі скляного електрода та еталонного зонда, підключених до підсилювального модуля. Сенсор зчитує електрохімічну напругу, що виникає при зануренні у розчин, і пропорційно перетворює її у значення рівня pH. Цей аналоговий сигнал подається на вхід АЦП мікроконтролера ESP32, де здійснюється подальша цифрова обробка.

Типовий діапазон вимірювання pH-сенсора становить від 0 до 14 одиниць pH з точністю до ±0,1–0,3 pH, що є цілком достатнім для потреб сільськогосподарських установок. Оптимальний рівень кислотності для більшості гідропонних культур перебуває в межах 5,5–6,5, тому система може бути запрограмована на сповіщення при відхиленні значень за встановлені межі.

Таким чином, використання pH-сенсора забезпечує постійний моніторинг кислотності розчину добрив, що дозволяє підтримувати стабільні умови для росту рослин, запобігати стресу культури та своєчасно виявляти відхилення від допустимих агрономічних параметрів.

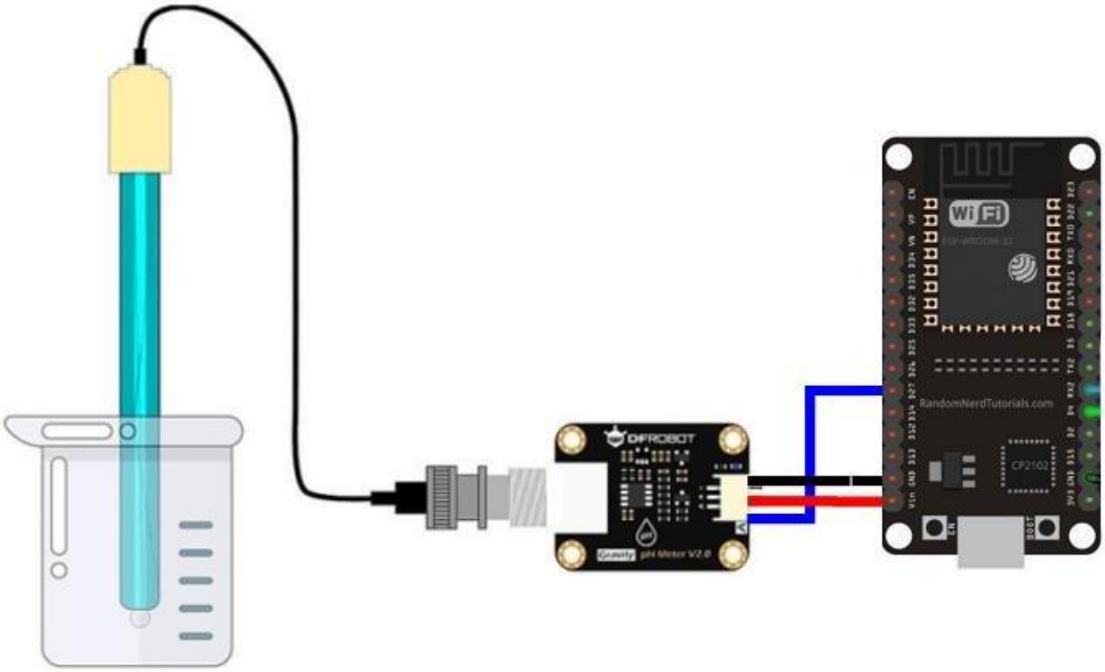


Рис. 3.2.3. Схема підключення ph сенсора до ESP32.

Для забезпечення ефективного контролю якості живильного розчину в гідропонній системі важливим є не лише моніторинг кислотності, а й оцінка концентрації розчинених солей, що безпосередньо впливає на доступність поживних речовин для рослин. З цією метою до складу IoT-модуля інтегровано сенсор електропровідності EC-сенсор, який вимірює електричну провідність розчину як індикатор загального мінерального складу.

EC-сенсор функціонує за принципом вимірювання електричного опору між двома або трьома електродами, зануреними у рідину. Оскільки провідність розчину прямо пропорційна вмісту іонів, пристрій дозволяє робити висновки про концентрацію таких речовин, як калій, натрій, кальцій, нітрати тощо. Значення вимірювань зазвичай подаються у мікросіменсах на сантиметр µS/cm або

мілісіменсах mS/cm, і охоплюють діапазон від 0 до 5000 µS/cm або більше залежно від моделі сенсора.

Сигнал, що надходить з датчика, зазвичай є аналоговим, тому для його зчитування використовується вбудований АЦП мікроконтролера ESP32. У більш точних рішеннях можливе застосування попереднього підсилення або використання спеціалізованого модуля інтерфейсу для компенсації температурної залежності, оскільки провідність розчину змінюється з температурою.

В межах розробленої IoT-системи EC-сенсор використовується для постійного моніторингу концентрації поживних речовин у розчині, з можливістю налаштування допустимих меж. У разі перевищення або зниження порогових значень користувач отримує попередження через термінал керування, що дає змогу своєчасно скоригувати склад розчину або вжити інших агротехнічних заходів.

Таким чином, використання EC-сенсора у складі IoT-модуля дозволяє автоматизовано стежити за критично важливим параметром живильного середовища — концентрацією солей, забезпечуючи оптимальні умови для росту гідропонних культур і мінімізуючи ризики пов’язані з недо- або перенасиченням середовища.

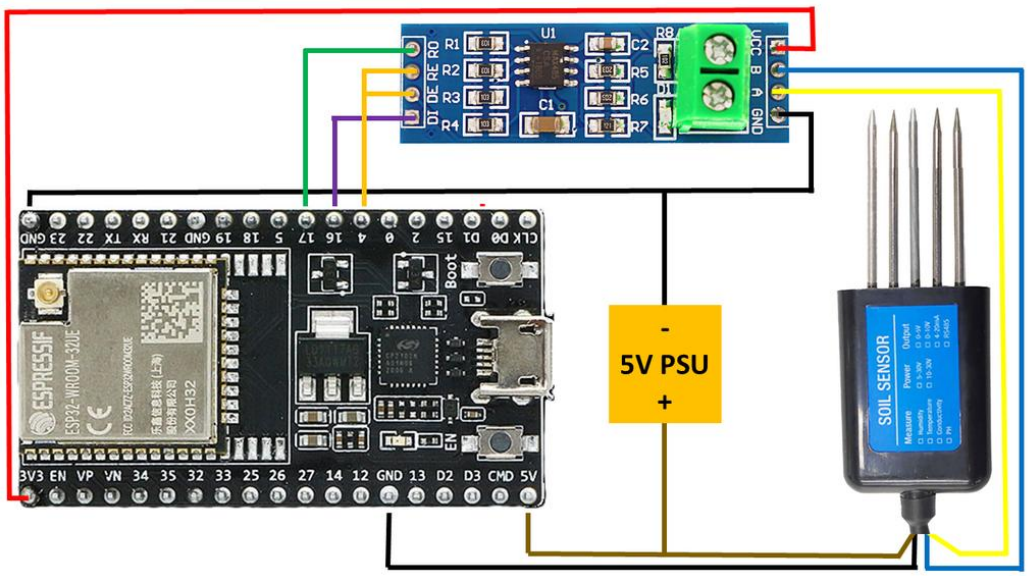


Рис. 3.2.4. Схема підключення EC сенсора до ESP32.

Реле: у складі IoT-модуля, який виконує функції автоматизованого керування гідропонною системою, важливу роль відіграють виконавчі компоненти, зокрема електромеханічні або твердотільні реле. Їх призначення полягає у забезпеченні фізичного керування високострумовими навантаженнями, такими як насос, система освітлення та вентиляційний блок, через сигнали низького рівня, що подаються з мікроконтролера.

У реалізованій системі використано релейні модулі, сумісні з мікроконтролером ESP32, які дозволяють комутувати змінну або постійну напругу живлення виконавчих пристроїв. Реле керується цифровим виходом мікроконтролера: при активації піну ESP32 на модулі замикається або розмикається відповідний контакт, що викликає увімкнення або вимкнення підключеного обладнання. Кожне реле має ізоляцію від логічної частини, що підвищує безпеку й надійність у роботі з електричними навантаженнями.

Основними перевагами використання реле є простота підключення, надійність комутації та гнучкість в управлінні різними типами навантажень. Такі модулі можуть комутувати напругу 220 В для живлення ламп або насосів, а також працювати з 12/24 В у низьковольтних системах. Крім того, можливість використання мультирелейних модулів. Наприклад, 4-канальні або 8-канальні блоки, що дозволяє компактно керувати кількома пристроями одночасно з одного контролера.

У контексті гідропонної системи, кожне реле відповідає за окремий функціональний елемент: одне — за насос подачі розчину, друге — за освітлення рослин, третє — за вентиляцію. Команди з термінала передаються до ESP32, який активує відповідне реле згідно з поточним станом або користувацькою командою. Таким чином, релейна частина забезпечує фізичну реалізацію керування середовищем, що є ключовим у процесах автоматизації.

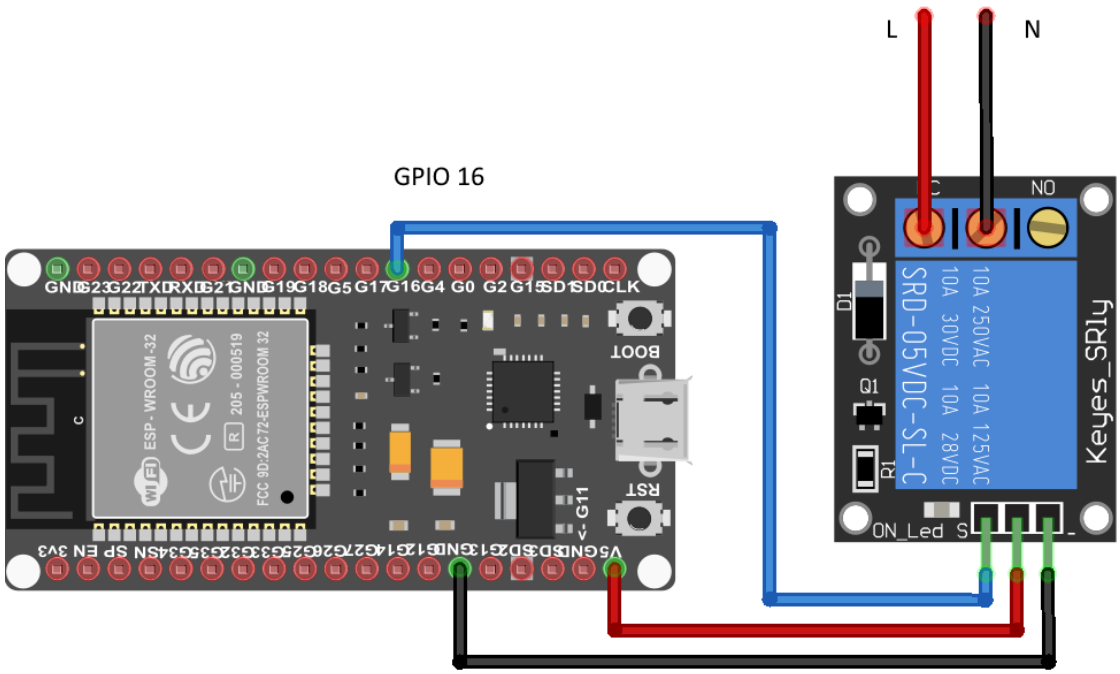


Рис. 3.2.5. Схема підключення модуля реле до ESP32.

У результаті аналізу й практичного впровадження було сформовано оптимальну апаратну конфігурацію IoT-модуля, призначеного для роботи в складі мультицентричної системи керування. Центральним елементом обрано мікроконтролер ESP32, який поєднує високу обчислювальну продуктивність, підтримку бездротових технологій (Wi-Fi та Bluetooth), широкий набір периферійних інтерфейсів та енергоефективність. Саме ці властивості забезпечують гнучкість і масштабованість системи.

Склад сенсорних компонентів модуля підібрано з урахуванням специфіки гідропонного застосування. Для вимірювання температури та вологості повітря використано сенсор DHT11, як надійне й просте у підключенні рішення. Для контролю температури води інтегровано водонепроникний цифровий датчик DS18B20, який демонструє високу точність та стабільність у вологому середовищі. Моніторинг кислотності pH забезпечується за допомогою електрохімічного pH-сенсора з аналоговим виходом, а оцінка концентрації поживних речовин — через сенсор електропровідності EC, що дозволяє стежити за загальним мінеральним складом живильного розчину.

Виконавчі елементи, зокрема реле, забезпечують керування насосом, освітленням і вентиляцією, що в сукупності формує повноцінний автоматизований гідропонний модуль. Завдяки структурі, що передбачає циклічний обмін даними між сенсорами, контролером і терміналом, а також можливість прямого реагування на зміну умов, система демонструє стабільну, адаптивну та технічно обґрунтовану архітектуру для сучасного IoT-застосування.

ВИСНОВКИ

У ході розробки системи мультицентричного керування IoT-пристроями було реалізовано повнофункціональне рішення, що охоплює як інтерактивний інтерфейс керування, так і апаратну частину інтелектуального модуля, здатного до автономної роботи. Стартовою точкою взаємодії користувача є головне вікно термінала, де представлена загальна інформація про систему та надається доступ до основних дій: створення груп, додавання пристроїв, перегляд активних модулів.

Функціональність інтерфейсу передбачає створення логічних груп пристроїв, що дозволяє впорядковувати IoT-модулі за призначенням або місцем розташування. Після створення групи користувач може додавати нові пристрої, вказуючи для кожного з них назву, адресу, тип і належність до певної групи. Всі пристрої відображаються в головному вікні, де доступні засоби пошуку та сортування — за назвою, типом або групою. Це суттєво підвищує зручність управління у випадках масштабування системи.

Після додавання пристрою до системи користувач має змогу відкривати вікно контролю модуля, яке адаптується до вибраного типу пристрою. У випадку гідропонного модуля вікно містить елементи керування насосом, освітленням і вентиляцією, а також кнопки для одночасного запуску або зупинки всіх компонентів. При активному підключенні пристрою інтерфейс динамічно відображає поточний стан виконавчих елементів та виводить актуальні дані з сенсорів: температуру повітря й води, вологість, кислотність і солоність розчину. Для гнучкості управління передбачена також можливість видалення груп пристроїв, що дозволяє оперативно змінювати структуру системи.

Апаратна частина IoT-модуля реалізована на базі мікроконтролера ESP32, який забезпечує високу обчислювальну здатність, підтримку бездротової комунікації та широкий набір інтерфейсів. До складу модуля включено необхідні сенсори: DHT11 для температури та вологості повітря, DS18B20 у водозахищеному корпусі для вимірювання температури розчину, pH-сенсор для моніторингу кислотності та EC-сенсор для контролю концентрації поживних речовин.

Управління фізичними компонентами, зокрема насосом, лампами та вентиляцією, здійснюється через релейні модулі, які дозволяють безпечно комутувати високострумові навантаження з рівня логіки мікроконтролера.

У сукупності, реалізована система поєднує гнучкість інтерфейсу, точність апаратного моніторингу та надійність керування, що дозволяє використовувати її як базову платформу для побудови інтелектуальних аграрних рішень. Побудована архітектура є масштабованою, адаптованою до різних типів пристроїв і готовою до подальшого розвитку з урахуванням нових вимог або сценаріїв використання.

ПЕРЕЛІК ПОСТЛАНЬ

1. Design and Implementation of ESP32-Based IoT Devices. URL: https://www.mdpi.com/1424-8220/23/15/6739.
2. Developing IoT Projects with ESP32: Discover the IoT Development Ecosystem with ESP32 to Create Production-Grade Smart Devices.
3. Getting Started with MicroPython on ESP32 and ESP8266. URL: https://randomnerdtutorials.com/getting-started-micropython-esp32-esp8266/
4. Fitzpatrick M. PyQt5 Tutorial 2025, Create Python GUIs with Qt. *Python GUIs*. URL: https://www.pythonguis.com/pyqt5-tutorial/
5. BLE Communication with PyQt5 Based GUI URL: https://medium.com/%40imlent/ble-communication-with-pyqt5-based-gui-6d3c2f3ac96f
6. Characterization and Performance Evaluation of ESP32 for Real-Time Sensor Networks
7. Omdahl J. Energy Efficiency Improvements in Smart Grid Components. Scitus Academics LLC, 2017.
8. Chang E. Fresher PyQt5: A Beginner's Guide to PyQt5. Independently Published, 2020.
9. Cameron N. ESP32 Formats and Communication. Berkeley, CA: Apress, 2023. URL: https://doi.org/10.1007/978-1-4842-9376-8
10. Delsing J. Iot Automation. Taylor & Francis Group, 2020.
11. Minev S. Exploring the Power of PyQt5: Building Interactive GUI Applications. *Medium*. URL: https://medium.com/@stefanminev_54009/exploring-the-power-of-pyqt5-building-interactive-gui-applications-4718529a902e.
12. Bits of Analytics - Getting started with PyQt5 for Python GUI development. *Bits of Analytics - Tutorials, notes, and ramblings on analytics*.

URL: https://bitsofanalytics.org/posts/pyqt5-getting-started/pyqt5_getting_started.

13. Industrial IoT / ed. by I. Butun. Cham: Springer International Publishing, 2020. URL: https://doi.org/10.1007/978-3-030-42500-5
14. IoT Editorial Office I. Acknowledgment to the Reviewers of IoT in 2022. *IoT*. 2023. Vol. 4, no. 1. P. 56. URL: https://doi.org/10.3390/iot4010003
15. Okunishi T. IoT (IoT Supporting our Life and Industry). *Nippon Shokuhin Kagaku Kogaku Kaishi*. 2017. Vol. 64, no. 5. P. 283. URL: https://doi.org/10.3136/nskkk.64.283

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Система моніторінгу та керування IoT-пристроями на основі ESP

Виконав студент 4 курсу групи ІСД-41
Пантелеєв Денис Анатолійович
Керівник роботи
Завацький Владислав Олександрович

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – підвищення гнучкості та надійності IoT-систем шляхом реалізації мультицентричної архітектури керування.
- **Об'єкт дослідження** – процес керування IoT-пристроями у розподіленому середовищі.
- **Предмет дослідження** – програмне забезпечення для мультицентричного керування IoT-пристроями.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати предметну область систем керування IoT-пристроями.
2. Дослідити переваги та недоліки існуючих систем керування IoT-пристроями.
3. Визначити вимоги до архітектури системи та програмного забезпечення IoT-пристроїв.
4. Розробити інтерфейс терміналу керування на основі PyQt5.
5. Реалізувати програмне забезпечення для IoT-пристроїв на C++.
6. Провести тестування системи на стабільність, зручність та функціональність.

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Платформа	Архітектура	Спрямування	Складність використання
Amazon AWS IoT Core	Централізована, хмарна	Комерційна	Складна, орієнтована на розробників
Microsoft Azure IoT	Централізована, хмарна	Комерційна	Складна, вимагає знань DevOps
Blynk	Централізована (частково локальна)	Freemium	Зручна, але обмежена
Home Assistant	Локальна (можлива хмара)	Відкрита	Гнучка, але потребує багато налаштувань

ЗАГАЛЬНІ ВИМОГИ ДО СИСТЕМИ

Вимоги до IoT-модулів

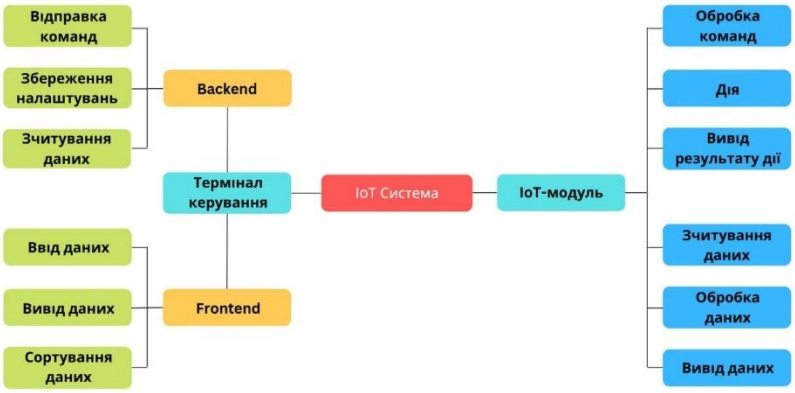
1. Типізовані пристрої та уніфіковані команди.
2. Автономність у прийнятті рішень.
3. Просте підключення до мережі.
4. Апаратне забезмечення IoT-модулів

Вимоги до графічного інтерфейсу термінала керування

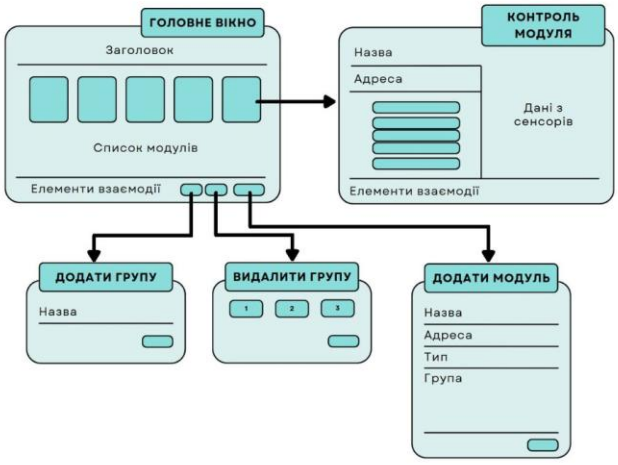
1. Пошук та фільтри.
2. Створення груп пристроїв
3. Журналу подій.
4. Додавання нових модулів.
5. Моніторинг та керування модулями

ЗАСОБИ РЕАЛІЗАЦІЇ

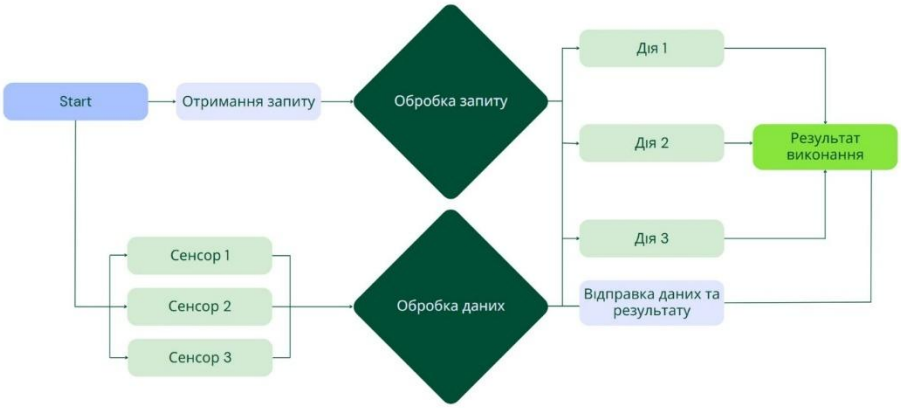
СТРУКТУРА IOT-СИСТЕМИ



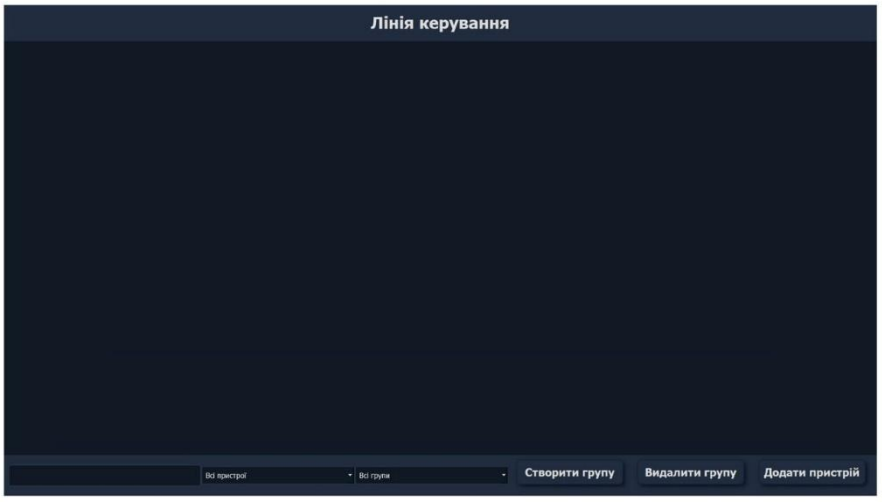
СТРУКТУРА ГРАФІЧНОГО ІНТЕРФЕЙСУ



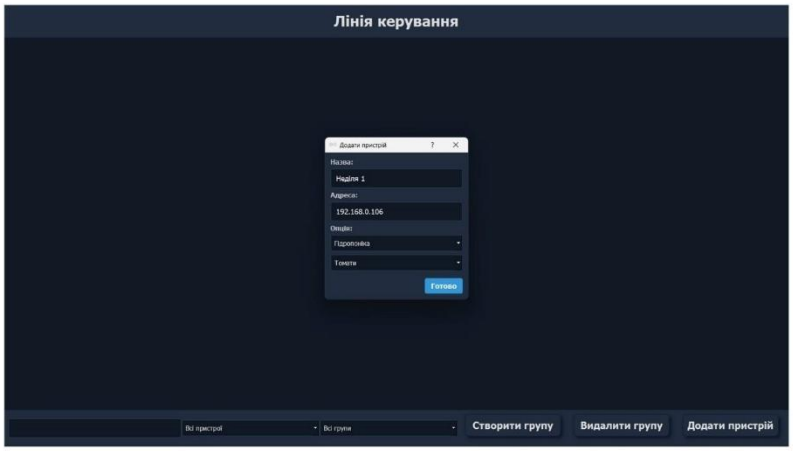
АРХІТЕКТУРА IOT-МОДУЛІВ



ГРАФІЧНИЙ ІНТЕРФЕЙС

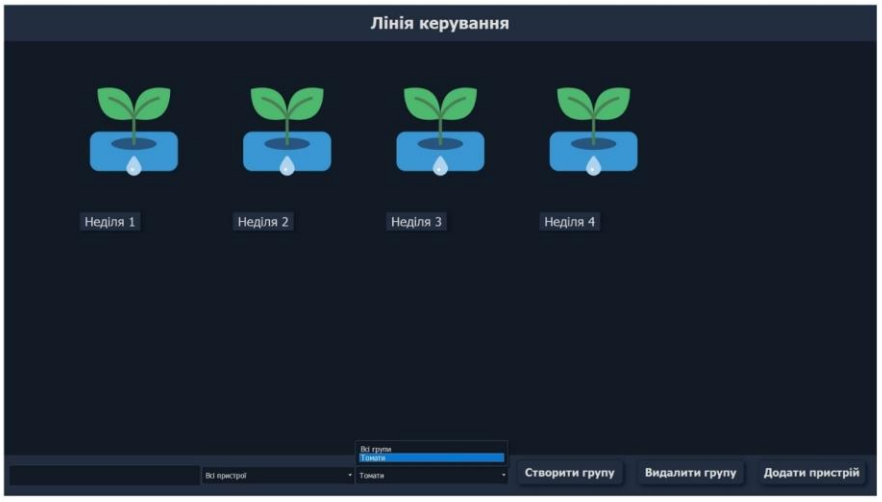


ГРАФІЧНИЙ ІНТЕРФЕЙС



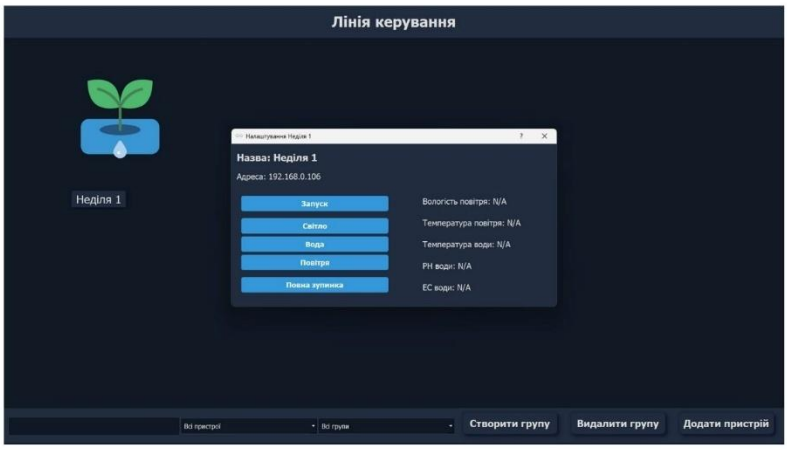
10

ГРАФІЧНИЙ ІНТЕРФЕЙС

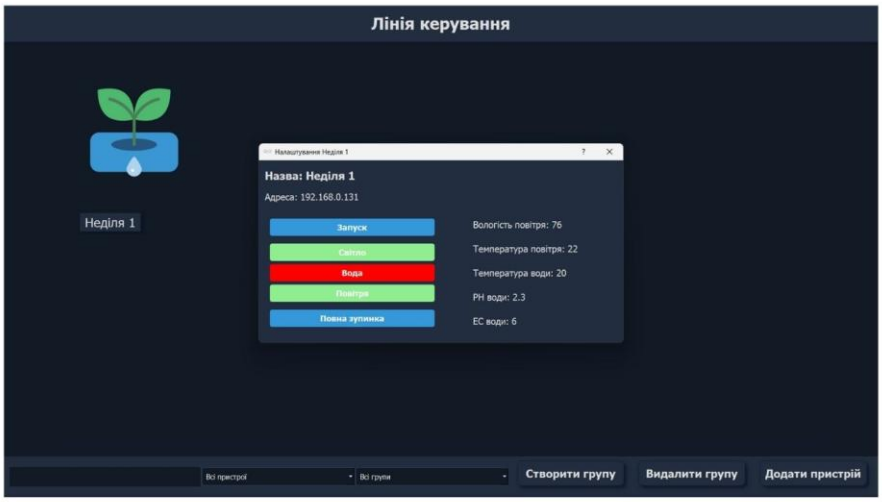


11

ГРАФІЧНИЙ ІНТЕРФЕЙС



ГРАФІЧНИЙ ІНТЕРФЕЙС



ВИСНОВКИ

У процесі розробки мультицентричної системи керування IoT-пристроями створено повнофункціональне рішення з інтерактивним інтерфейсом і апаратним модулем на базі ESP32, що забезпечує автономну роботу. Інтерфейс дозволяє створювати групи пристроїв, додавати й сортувати модулі, а також здійснювати гнучке управління їхніми функціями залежно від типу. Система підтримує динамічне відображення стану виконавчих елементів і сенсорні вимірювання, а керування здійснюється через релейні модулі. Рішення є масштабованим, надійним і придатним для інтелектуальних аграрних застосувань.

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. **Д. Пантелеєв, В. Завацький. Застосування PyQt5 для моніторингу та керування IoT-пристроям. 2-а Міжнародна науково-практична конференція «Інформаційні системи та технології: результати і перспективи» (IST 2025), 5 березня 2025 р. (Київ, Україна). К. : ФІТ КНУТШ, 2025 р. С. 121-122.**
2. **А. Казначеєва, Д. Пантелеєв. Підходи в інтеграції гібридних енергетичних систем в IoT. 2-а Всеукраїнська науково-технічна конференція «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу україни і світу» (IST 2025), 18 листопада 2024 р. (Київ, Україна). К. : ІСТ ДУІКТ, 2024 р. С. 292.**

ДЯКУЮ ЗА УВАГУ!