ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка програмно-апаратного комплексу автоматизованого моніторингу та підтримання життєдіяльності рослин»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

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

_____ *Данил СТЕЦЕНКО*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-42
Данил СТЕЦЕНКО
Ім'я, ПРІЗВИЩЕ

Керівник: доцент Волохін Віталій Васильович
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

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

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерної інженерії
_____ Наталія ЛАЩЕВСЬКА
_____
«_____» _____ 2026р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Стеценко Данил Романович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка програмно-апаратного комплексу автоматизованого моніторингу та підтримання життєдіяльності рослин
керівник кваліфікаційної роботи доцент Волохін Віталій Васильович,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «____» _____ 2026р. №_____
2. Строк подання кваліфікаційної роботи «___» _____ 2026р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література; Концепція інтернету речей; архітектура периферійних обчислень (Edge computing); мікрокомп'ютер Raspberry Pi; аналого-цифровий перетворювач; платформа .Net та мова C#; бази даних; Entity Framework Core; ASP.NET Core веб-розробка; протоколи та інтерфейси: I2C, GPIO, SSH; алгоритми керування;
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи та аналіз систем автоматичного контролю за рослинами
2. Проектування інфраструктури автоматизованої теплиці
3. Практична реалізація та оцінка ефективності
5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «___» _____ 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та існуючих систем автоматизації догляду за рослинами.	02.03-20.03.2026	Виконано
2	Вибір апаратних компонентів та проєктування інформаційної моделі.	25.03-10.04.2026	Виконано
3	Розробка алгоритмів керування мікрокліматом, збірка готової системи та налаштування робочого середовища.	13.04-18.04.2026	Виконано
4	Написання програмного коду керуючої служби та налаштування її фонової роботи в ОС Linux.	20.04-28.04.2026	Виконано
5	Розробка клієнт-серверного застосунку із веб-інтерфейсом користувача та проведення тестування комплексу в реальних умовах.	28.04-01.05.2026	Виконано
6	Оформлення розділів пояснювальної записки відповідно до вимог інституту.	10.04-02.05.2026	Виконано
7	Підготовка ілюстрацій, презентації та тексту доповіді.	04.05-08.05.2026	Виконано
8	Попередній захист кваліфікаційної роботи на кафедрі.	11.05-15.05.2026	Виконано

Здобувач(ка) вищої освіти _____ Данил СТЕЦЕНКО

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Віталій ВОЛОХІН

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: n стор., n рис., n табл., n джерел.

Мета роботи - розробка автоматизованої системи для керування розумною теплицею на базі мікроконтролера Raspberry Pi 3 Model B.

Об'єкт дослідження - процес автоматизованого догляду з мінімальним втручанням людини за рослинами у закритому ґрунті.

Предмет дослідження - апаратно-програмний комплекс для автоматизації розумної теплиці на базі Raspberry Pi та застосунку на мові програмування C#.

Короткий зміст роботи: Процес вирощування рослин вимагає постійного контролю. Різні культури потребують індивідуального підходу до освітлення, поливу, вологості повітря і т.д. Керування цими процесами вручну не завжди є ефективним та підвищує ризик втрати врожаю через людський фактор, а тому постає питання, як з боку користувача забезпечити автономну та стабільну підтримку ідеальних умов для рослин без постійного нагляду. Це досягається використанням у тому числі систем «розумних» теплиць, обладнаних датчиками та виконавчими механізмами, такими як водяні помпи та реле. Таку можливість надає функціонал Raspberry Pi у поєднанні з розробленим програмним забезпеченням. Система автоматично збиратиме дані з датчиків вологості грунту та інших сенсорів, порівнюватиме їх з індивідуальними для кожної рослини профілями догляду з бази даних та вмикатиме відповідні модулі.

На основі проведеного дослідження було досліджено актуальність розробки власної системи автоматичного догляду на базі Raspberry Pi та C#, особливості її побудови, апаратні та програмні компоненти.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗОВАНА ТЕПЛИЦЯ, RASPBERRY PI, C#, БАЗА ДАНИХ, МІКРОКЛІМАТ, ВОЛОГІСТЬ, АЛГОРИТМ, УФ-ДАТЧИК, РЕЛЕ.

ЗМІСТ

ВСТУП 3
1. ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ СИСТЕМ АВТОМАТИЧНОГО КОНТРОЛЮ ЗА РОСЛИНАМИ 4
1.1 Аналіз предметної галузі та актуальність автоматизації мікроклімату 4
1.2 Огляд існуючих рішень та систем «розумних» теплиць 6
1.3 Обґрунтування вибору апаратної (Raspberry Pi) та програмної (C#) платформ 10
1.4 Концепція Інтернету речей та периферійних обчислень у сучасній автоматизації 12
2. ПРОЕКТУВАННЯ ІНФРАСТРУКТУРИ АВТОМАТИЗОВАНОЇ ТЕПЛИЦІ 13
2.1 Аналіз технічних вимог та формування загальної апаратної архітектури 13
2.2 Підбір компонентів та розрахунок апаратного навантаження 15
2.3 Проєктування структури бази даних 25
2.4 Розробка алгоритмів керування системою 27
2.5 Візуальна модель системи з використанням UML 30
2.6 Організація процесу запуску 32
3. КОНСТРУКТИВНИЙ РОЗДІЛ: ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ 34
3.1 Програмна реалізація логіки керування системою мовою C# 34
3.2 Збірка, налаштування та тестування 36
3.3 Розробка веб-інтерфейсу користувача та організація клієнт-серверної взаємодії 39
3.4 Напрями вдосконалення та перспективи розвитку системи 42
ВИСНОВОК 44

ВСТУП

У 21-ому столітті автоматизація глибоко проникла майже в усі сфери людської діяльності, в тому числі і у сільське господарство та індивідуальне рослинництво. Процес вирощування квітів у закритому ґрунті завжди вимагає безперервного моніторингу. Більшість людей схиляються до повністю ручного догляд, контролюючи життедіяльність рослин спираючись на власний досвід.

Для невеликої кімнатної оранжереї такого підходу зазвичай достатньо, але коли виникає необхідність підтримувати ідеальні умови, він не є ефективним. Людина не здатна цілодобово аналізувати показники середовища, пропущений або надмірний полив, зміна сонячної активності, та інші фактори можуть нашкодити рослинам. Для того, щоб користувач мав стабільний результат без необхідності постійної фізичної присутності, управління процесами варто делегувати автоматиці. Це дозволяє усунути ризик людської помилки та створити певну відмовостійкість процесу вирощування.

Можливість побудувати таку архітектуру надає використання мікрокомп’ютерів у поєднанні з профільним програмним забезпеченням. Програмна логіка дозволяє створити гнучкий центр керування, апаратна частина збирає реальні фізичні показники (вологість грунту, рівень освітлення тощо), а програмний код обробляє отримані дані, а інші пристрої створюють необхідні умови, задані у базі даних, отримуючи керуючі сигнали від мікроконтролера. Таким чином, розробка подібного апаратно-програмного комплексу дозволяє майже повністю забезпечити автономне підтримання ідеальних умов.

1. ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ СИСТЕМ АВТОМАТИЧНОГО КОНТРОЛЮ ЗА РОСЛИНАМИ

1.1 Аналіз предметної галузі та актуальність автоматизації мікроклімату

На відміну від природного середовища, вирощування культур у закритому ґрунті є складним процесом, тому що тут практично відсутні механізми саморегуляції, і підтримка життєздатності рослин повністю лягає на систему управління або на людину. Хоча традиційний ручний підхід історично домінував у невеликих господарствах, його обмеженням завжди було те, що людина не може ідеально реагувати на зміну середовища.

Показовим прикладом неефективності ручного догляду є процес поливу. Полив за допомогою лійки, шланга чи іншого ручного інвентарю має низку недоліків. По-перше, одномоментне внесення великого об'єму води створює гідравлічний удар на поверхні субстрату. Через це ущільнюється верхній шар ґрунту, а це сильно погіршує аерацію кореневої системи та може викликати гіпоксію (кисневий голод) рослин. По-друге, значна частина води просто проходить крізь кореневий шар транзитом, вимиваючи цінні макро- та мікроелементи в нижні частини ґрунту, де вони стають недоступними для живлення.

Натомість автоматизований полив, що керується показниками ємнісних або резистивних датчиків вологості ґрунту, працює інакше. Датчики дозволяють орієнтуватися на об'ємну вологість у режимі реального часу. Коли показник падає нижче критичного рівня, мікроконтролер активує водяну помпу, яка подає воду дозовано, і завдяки цьому волога розподіляється у субстраті не за рахунок гравітаційного просочування, а завдяки капілярним силам ґрунту. Це зберігає пухку

структуру землі, забезпечує прийнятний баланс води та кисню біля коріння і виключає як пересихання, так і утворення боліт у горщику.

Крім вологості ґрунту, фундаментальне значення має мікроклімат повітря. Сучасна проблематика тепличного господарства вимагає відмовитися від розгляду окремих показників температури та відносної вологості на користь їх комплексного взаємозв'язку - дефіциту тиску водяної пари. Цей показник показує здатність повітря поглинати вологу, а це впливає на те, як якісно рослини можуть випаровувати вологу через листя. Якщо повітря у теплиці занадто сухе, рослина припиняє процес втрати вологи, і процес фотосинтезу зупиняється. Якщо ж повітря перезволожене, рослина втрачає здатність споживати воду з поживними речовинами своїм корінням. Розрахунок та постійна підтримка такого балансу вимагають безперервного аналізу телеметрії з кількох сенсорів одночасно, а це, очевидно, неможливо реалізувати без застосування обчислювальної техніки.

Не менш критичним фактором є управління освітленням. Фізіологічні процеси рослин залежать не від факту наявності світла, а від його спектрального складу (в основному, УФ-випромінювання та фотосинтетично активної радіації) та строгих циклів світлового дня. Механічне вмикання ламп користувачем не здатне забезпечити стабільні багатодобові алгоритми, які б імітували ідеальні умови для різних фаз вегетації, таких як проростання, цвітіння та плодоношення.

Огляд предметної області свідчить про те, що біологічні потреби рослин можна відслідковувати за допомогої інженерії та алгоритмізації, що покращить їх умови, якщо перейти від візуального контролю до інженерії. Використання мікрокомп'ютерних платформ для збору даних та керування наосами, лампами, зволожувачами тощо дозволяє автономно гарантувати дотримання правильного догляду та мінімізувати ризики помилки.

1.2 Огляд існуючих рішень та систем «розумних» теплиць

Сучасні системи автоматизації закритого ґрунту можна розділити на три основні категорії залежно від принципу їх роботи та цільового сегмента. Найпростішим і найдоступнішим варіантом є використання автономних таймерів-розеток та базових реле часу (рис. 1). Вони функціонують за розкладом, який задає користувач, замикаючи живлення помп або фітоламп у визначені години. Перевагою таких пристроїв є їхня простота налаштування та низька вартість. Недолік цього підходу полягає у тому, що система не знає реальних умов середовища, наприклад, полив відбуватиметься навіть тоді, коли ґрунт вже перезволожений через зниження температури в приміщенні. Це може призводити до загнивання кореневої системи, і не підпадає під визначення “автоматизованого контролю”.



Рисунок 1.1 - Цифровий таймер поливу на шланг

Також існують готові набори, побудовані переважно на базі мікроконтролерів сімейства Arduino або ESP (рис. 2). Їхня логіка роботи вже спирається на зворотний зв'язок від середовища, за допомогою аналогових датчиків вологості та фоторезисторів, що передають сигнали на плату, яка активує виконавчі механізми при досягненні певних порогових значень, прописаних у коді. З одного боку, такі рішення дозволяють створити справжню базову автоматизацію, вони відносно недорогі, а пошуку готових програмних бібліотек досить простий для просунутого користувача. З іншого боку, більшість таких наборів комплектуються дешевими резистивними датчиками ґрунту, які швидко виходять з ладу через постійну гальванічну корозію у вологій землі, до того ж, неможливо побудувати на базі примітивного мікроконтролера складну архітектуру з використанням реляційних баз даних для збереження профілів рослин чи реалізувати багатопотоковість.

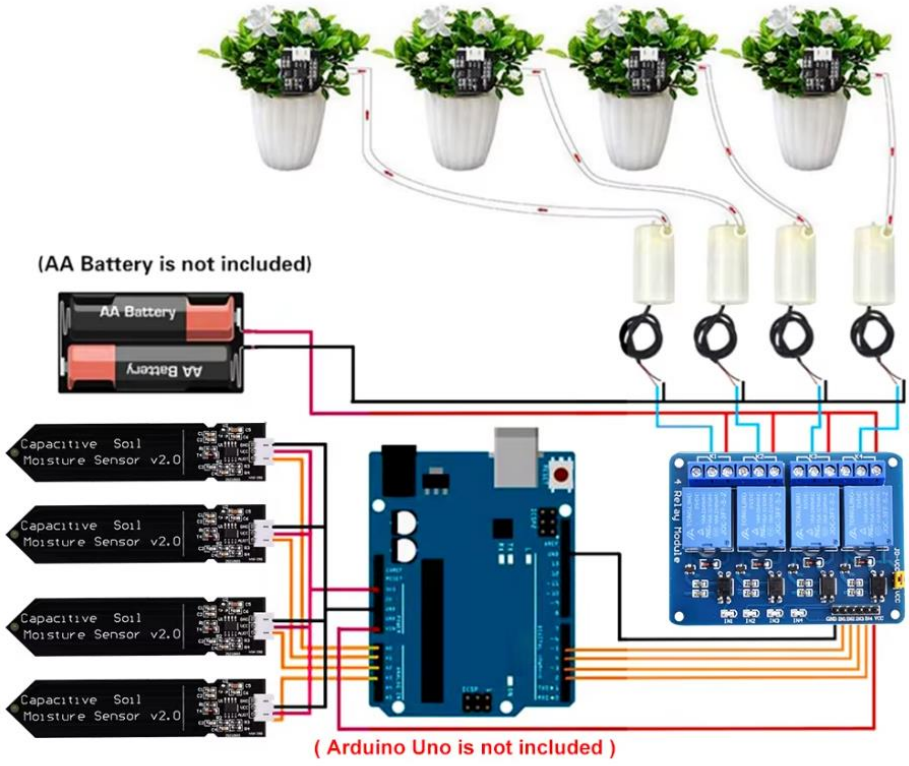


Рисунок 1.2 - один з наборів автоматизованого поливу, що продаються у сервісі AliExpress

Третій сегмент представлений великими та дорогими промисловими комплексами від спеціалізованих аграрних виробників. Вони інтегровані з хмарними сервісами можуть аналізують метеодані та здатні керувати мікрокліматом на великих площах у десятки гектарів. Їхній функціонал покриває всі вищеперераховані потреби і навіть більше, але для приватного використання, (як невеликі автономні теплиці чи дослідницькі проєкти) вони є надлишковими. Головною перешкодою стає не лише їхня вартість, але й закритість екосистеми. Користувач не може самостійно додати власний і нестандартизований УФ-датчик або змінити логіку опитування бази даних без придбання дорогих фірмових модулів розширення та залучення офіційних інтеграторів.

Окремої уваги заслуговують апаратні рішення для гідропоніки — методу вирощування без використання ґрунту, де коріння живиться безпосередньо з циркулюючого водного розчину. Це вузькоспеціалізований сегмент ринку, який вимагає найвищого рівня апаратного контролю. Відмова від класичного субстрату (тобто ґрунту) дозволяє досягти максимальної швидкості росту та виключає ризик ґрунтових хвороб, проте ціна помилки тут є критичною. Оскільки ґрунт не виступає буфером, будь-яка апаратна затримка помпи або збій у дозуванні поживних речовин здатні незворотно пошкодити кореневу систему за лічені години. Тому автоматизація гідропонних установок об'єктивно є більш складною, і вона потребує інтеграції специфічних і дорогих сенсорів рівня кислотності та електропровідності, а також застосування дублюючих систем безпеки, що робить такі рішення надмірно складними та фінансово обтяжливими для класичних ґрунтових теплиць, проте вони є найпростішими для догляду з точки зору користувача.



Рисунок 1.3 - гідропонна система компанії Yoocaa

Враховуючи все вищесказане, можна побачити пробіл в сучасних пропозиціях, а саме - система, яка б поєднувала серйозні обчислювальні можливості, достатні для розгортання локальної бази даних і складних алгоритмів, але яка залишалася б економічно доступною та відкритою для модифікацій. Саме тому проєктування власного комплексу на базі мікрокомп'ютера Raspberry Pi з написанням власної програмної логіки є найбільш раціональним шляхом, який дозволяє подолати обмеження існуючих на ринку систем та створити дещо нове.

1.3 Обґрунтування вибору апаратної (Raspberry Pi) та програмної (C#) платформ

При проєктуванні будь-якої складної системи необхідне створення прототипу. У рамках даної кваліфікаційної роботи не ставиться за мету проєктування фінального комерційного продукту, що готовий до масового виробництва, або створення вузькоспеціалізованої промислової плати. Натомість, мета полягає у демонстрації можливостей запропонованої архітектури, перевірці алгоритмів взаємодії між апаратною та програмною частинами, а також тестуванні концепції автоматизованого догляду за рослинами. Виходячи з цього, головними критеріями вибору платформ стали максимальна гнучкість налаштування, швидкість розгортання та зручність тестування.

Для реалізації апаратної частини прототипу було обрано одноплатний мікрокомп'ютер Raspberry Pi 3 model B. Потрібно зазначити, що для серійного виробництва контролерів зазвичай проєктуються власні друковані плати (PCB) на базі дешевих і енергоефективних мікропроцесорів. Проте для етапу первинного проєктування та перевірки Raspberry Pi виступає дуже потужною та гнучкою дослідницькою базою. На відміну від класичних мікроконтролерів, які виконують єдиний прошитий цикл інструкцій, ця платформа дозволяє запустити повноцінну операційну систему сімейства Linux. Це є важливою умовою для обраної архітектури, оскільки проєкт розумної теплиці вимагає одночасного виконання кількох ресурсоємних процесів: утримання локальної реляційної бази даних, маршрутизації мережевих запитів та безпосереднього опитування датчиків. Наявність на платі фізичної шини GPIO дозволяє напряму комутувати низькорівневу периферію, таку, як блоки реле, водяні помпи чи датчики освітлення, з розвинутою операційною системою, а також постійно змінювати чи вдосконалювати як код, так і апаратну частину.

Для написання логіки було обрано мову C# та платформу .NET. C# не єдиний можливий інструмент для подібних задач. В сегменті IoT зазвичай використовують C/C++/Python для швидких скриптів та економії пам'яті, але для цілей нашого проєктування C# пропонує баланс між продуктивністю виконання (тому що автор роботи найбільш знайомий саме з мовою C#) та зручністю побудови складної логіки. Ця мова має механізм асинхронного програмування (async/await), а це дозволяє реалізувати неблокуюче опитування сенсорів, наприклад, поки система чекає відповіді від датчика вологості через інтерфейс I2C, центральний процесор може паралельно обробляти запити до бази даних або створювати керуючі сигнали для інших модулів.

По-друге, екосистема .NET має вбудовані та надійно протестовані інструменти для роботи з реляційними даними. Це суттєво пришвидшує розробку модуля зберігання профілів рослин, дозволяючи використовувати об'єктно-реляційне відображення замість написання SQL-запитів, що значно знижує ризик помилок на етапі прототипування. До того ж, кросплатформеність сучасного стандарту .NET дозволяє писати та компілювати код на власному високопродуктивному ПК, а потім розгортати скомпільовану програму безпосередньо на ARM-архітектурі Raspberry Pi.

Таким чином, зв'язка Raspberry Pi та мови C# цілком підходить для проєктування. У нас відсутні обмеження, притаманні мікроконтролерам, що використовуються у масовому виробництві, можна сфокусуватися безпосередньо на розробці логіки мікроклімату і є можливість легко масштабувати чи модифікувати архітектуру проєкту в процесі розробки та у майбутньому.

1.4 Концепція Інтернету речей та периферійних обчислень у сучасній автоматизації

Наразі часто можна зустріти інтеграцію у сучасні пристрої концепцію Інтернету речей, яка передбачає їх об'єднання у єдину цифрову мережу для автоматизованого збору даних та обміну інформацією. У контексті аграрного сектору та індивідуального рослинництва ця концепція трансформувалася у напрямок «Smart Farming».

Проте, розгортання класичних IoT-систем сьогодні стикається з проблемою надмірної централізації. Традиційна модель передбачає, що всі зібрані дані з датчиків одразу передаються до віддалених хмарних серверів для обробки та прийняття рішень. Такий підхід має низку суттєвих недоліків, серед яких найголовніше - залежність від стабільності інтернет-з’єднання. У випадку автоматизованої теплиці навіть короткочасна втрата зв’язку з хмарним сервером може призвести до збою в алгоритмі поливу, а такого сценарію дуже бажано уникати. В основі даної кваліфікаційної роботи обрано альтернативний підхід - периферійні обчислення. Суть полягає у тому, що обчислювальні потужності та програмна логіка перенесені до джерела даних — на «периферію» мережі. У даній системі роль такого периферійного вузла виконує мікрокомп'ютер сімейства Raspberry Pi. Він не лише проміжний передавач, а й центр обчислень, який локально обробляє сигнали з сенсорів, звертається до вбудованої бази даних та самостійно формує керуючі сигнали.

Розроблений апаратно-програмний комплекс є прикладом впровадження архітектури периферійних обчислень у сегменті Інтернету речей, адже дозволяє створити не просто автоматизовану станцію, а самостійний IoT-вузол, здатний до автономного функціонування.

2. ПРОЕКТУВАННЯ ІНФРАСТРУКТУРИ АВТОМАТИЗОВАНОЇ ТЕПЛИЦІ

2.1 Аналіз технічних вимог та формування загальної апаратної архітектури

Почнемо з аналізу технічних вимог до системи. У контексті нашого дослідження автоматизована теплиця розглядається як багатокомпонентна система, що повинна працювати в умовах специфічного мікроклімату та забезпечувати надійну взаємодію між високорівневою обчислювальною логікою і низькорівневою периферією.

Загальну апаратну архітектуру розроблюваної системи можна концептуально розділити на три взаємопов'язані рівні: сенсорний (рівень збору даних), обчислювальний (рівень прийняття рішень) та виконавчий (рівень фізичного впливу на середовище), а також слід виділити окрему підсистему розподілу живлення.

Сенсорний рівень формує масив вхідних даних для системи. Відповідно до поставлених завдань щодо мікроклімату, архітектура передбачає інтеграцію кількох ключових типів датчиків. По-перше, це датчики вологості ґрунту. З інженерної точки зору перевага надається ємнісним датчикам, оскільки, на відміну від дешевих резистивних аналогів, їхні електроди не піддаються швидкій гальванічній корозії у вологому середовищі. По-друге, це модулі контролю освітлення, зокрема датчики ультрафіолетового (УФ) випромінювання. Вони є критично важливими, оскільки інтенсивність УФ-спектра безпосередньо впливає на протікання фізіологічних процесів у рослинах.

Обчислювальний рівень базується на мікрокомп'ютері Raspberry Pi 3 Model B. Його задачею є збір даних через інтерфейси введення-виведення GPIO, хостинг

локальної бази даних та виконання керуючих алгоритмів на C#. На цьому етапі проєктування виникає специфічна апаратна проблема: система, що використовується в Raspberry Pi, на відміну від мікроконтролерів класу Arduino, не має вбудованих апаратних аналого-цифрових перетворювачів (АЦП). Отже, мікрокомп'ютер не здатний напряму зчитувати аналогові сигнали з ємнісних датчиків вологості чи деяких типів УФ-сенсорів. Для вирішення цієї архітектурної проблеми інфраструктура обов'язково повинна включати зовнішні модулі АЦП або передбачати використання цифрових сенсорів, які спілкуються з центральним процесором через цифрові шини передачі даних, такі як I2C (Inter-Integrated Circuit) або SPI (Serial Peripheral Interface).

Також важливо було вирішити проблему узгодження логічних рівнів. Архітектура мікропроцесорів Raspberry Pi використовує логіку 3.3 В, тоді як значна частина стандартної периферії може бути розрахована на 5 В. Тому потрібно враховувати безпеку портів GPIO, використовуючи окремий блок живлення для запобігання пошкоджень контролера.

Виконавчий рівень складається з блоків реле, які виконують роль фізичних комутаторів для управління водяними помпами та фітолампами. Оскільки слабкострумові виходи Raspberry Pi не здатні безпосередньо комутувати потужне навантаження, сигнал керування подається на модуль реле. Значною перевагою цього вузла буде наявність опторозв'язки (оптронів). Оптоізоляція фізично розділяє керуючий ланцюг мікрокомп'ютера від силового ланцюга реле за допомогою світлового сигналу. Це важливо при роботі з індуктивними навантаженнями, якими є двигуни водяних помп. У момент вимкнення помпи виникає електрорушійна сила самоіндукції - різкий стрибок зворотної напруги, який без належної ізоляції та наявності захисних зворотних діодів може буквально миттєво вивести мікрокомп'ютер з ладу.

Окрема увага при проєктуванні архітектури приділяється підсистемі розподілу живлення. Поширеною помилкою при конструюванні подібних систем є

спроба живити електромеханічні компоненти, як вищезгадані насоси, безпосередньо від контактів 5V на платі Raspberry Pi. При одночасному запуску кількох таких пристроїв виникають пускові струми, які сильно перевищують номінальні, що призводить до різкого падіння напруги на загальній шині живлення. Мікрокомп'ютер фіксує недостатню напругу і може перезавантажитися або пошкодити файлову систему на накопичувачі. З огляду на це, спроєктована апаратна архітектура вимагає розділення ліній живлення: логічна частина (Raspberry Pi та цифрові сенсори) живиться від власного стабілізованого джерела, а виконавча частина - від окремого блоку живлення правильно підібраної потужності. При цьому для коректної передачі керуючих сигналів ці два незалежні контури обов'язково потрібно об’єднати спільною лінією заземлення - цей принцип також називається «Common ground».

Отже, визначена апаратна архітектура являє собою продуману багаторівневу топологію, яка і забезпечує збір необхідних даних для керування мікрокліматом, і гарантує електричну безпеку та стабільність обчислювального процесора в умовах роботи з індуктивними навантаженнями.

2.2 Підбір компонентів та розрахунок апаратного навантаження

Для реалізації спроєктованої архітектури автоматизованої теплиці критично важливим є вибір конкретних апаратних модулів, які забезпечать стабільну роботу системи в умовах тривалої експлуатації. Підбір компонентів здійснювався на основі аналізу їхніх технічних характеристик, енергоспоживання та сумісності з логічними рівнями Raspberry Pi.

Центральним обчислювальним вузлом системи обрано мікрокомп’ютер Raspberry Pi 4 Model B. Вибір даної моделі обумовлений наявністю достатнього об’єму оперативної пам’яті (1 Гб) для роботи повноцінної ОС на базі Unix, що,

своєю чергою, спрощує процес розробки. Крім того, наявність вбудованих інтерфейсів I2C та SPI дозволяє легко масштабувати сенсорну підсистему в майбутньому. Чому так – далі у роботі.

Ледь не найважливішою складовою апаратної частини є вимірювання рівня вологості ґрунту. Зараз існують два основні типи датчиків - резистивні та ємнісні. Резистивні датчики мають два відкриті металеві електроди, між ними пропускається електричний струм, і система вимірює опір середовища. Що більше води в ґрунті, то менший опір через домішки солей у воді. Проте цей метод має недолік для тривалої експлуатації: проходження струму через вологе середовище викликає електроліз та гальванічну корозію, що призводять до хибних показників. З огляду на це, для даної було обрано ємнісні датчики вологості (модель Capacitive Soil Moisture Sensor v1.2, рис. 2.1). У них інший принцип роботи, датчик виступає в ролі конденсатора, а сам ґрунт є діелектриком. Зміна кількості води у змінює його діелектричну проникність, що, у свою чергу, впливає на ємність датчика. Вимірювальні доріжки повністю покриті водонепроникною паяльною маскою, вони не мають прямого контакту з ґрунтом, а це повністю виключає ризик корозії. Такі датчики генерують аналоговий сигнал від 0 до 3,0 В, який відображає об'ємний вміст води незалежно від мінералізації ґрунту.



Рисунок 2.1 - Capacitive Soil Moisture Sensor V1.2

Проблемою інтеграції датчиків вологості ґрунту саме цієї моделі (та сенсорів ультрафіолетового випромінювання) є те, що ці датчики - аналогові. Такі сенсори реагують на зміни фізичного середовища плавною та поступовою зміною напруги на виході. Але центральний процесор моделі Raspberry Pi 3 (Broadcom BCM2711) є цифровим пристроєм, він може оперувати лише дискретними логічними рівнями: наявність напруги близько 3.3 В розпізнається як логічна одиниця (HIGH), а відсутність напруги (0 В) - логічний нуль (LOW). Процесор не має апаратних можливостей для розпізнавання проміжних значень напруги. Якщо подати аналоговий сигнал 2.1 В безпосередньо на цифровий пін GPIO (General-Purpose Input/Output), мікрокомп'ютер може зчитати його некоректно через явище блукаючого потенціалу, а в гіршому випадку пін взагалі вийде з ладу.

Для вирішення цієї несумісності було використано аналогово-цифровий перетворювач на базі мікросхеми ADS1115 (рис. 2.2). Його головна функція полягає у процесі квантування: перетворенні неперервного аналогового сигналу у вигляді напруги на дискретне ціле число (цифровий код).

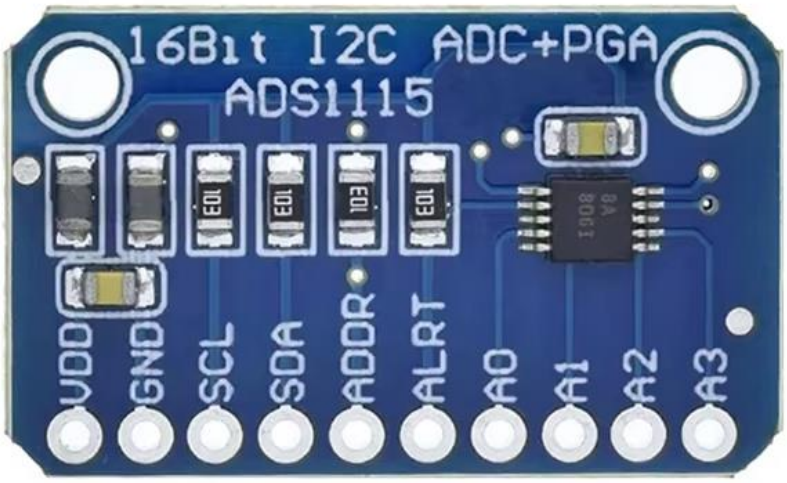


Рисунок 2.2 - Аналого-цифровий перетворювач на базі чіпу ADS1115

Після того, як АЦП виміряв напругу і перетворив її на числове значення, це число зберігається у внутрішньому регістрі мікросхеми, а далі передається до Raspberry Pi. АЦП ADS1115 підтримує до чотирьох аналогових каналів одночасно, а тому передавати дані по окремих дротах для кожного датчика було б нераціонально і призвело б до швидкого вичерпання вільних пінів мікрокомп'ютера. Натомість передача масиву даних здійснюється через цифрову шину I2C (Inter-Integrated Circuit). Цей протокол є галузевим стандартом для зв'язку між мікросхемами на невеликих відстанях і вимагає лише двох фізичних ліній зв'язку – SDA (Serial data line) та SCL (serial clock line). SDA - лінія, по якій передаються біти даних в обидва боки (від АЦП до Raspberry Pi і навпаки), SCL - лінія, по якій Raspberry Pi генерує прямокутні імпульси для синхронізації передачі кожного біта. Взаємодія відбувається за принципом «Master-Slave», де мікрокомп'ютер Raspberry Pi виступає в ролі ведучого (Master). Логіка програмного забезпечення формує запит та відправляє його на шину I2C, вказуючи унікальну

шістнадцяткову апаратну адресу АЦП (зазвичай 0x48). АЦП (Slave), розпізнавши свою адресу на шині, відповідає двома байтами даних, які містять оцифроване значення напруги з потрібного датчика. Це рішення не лише вирішує проблему відсутності вбудованих аналогових входів у Raspberry Pi, а й створює масштабовану архітектуру, тому що завдяки адресній системі шини I2C, до двох пінів SDA та SCL можна за потреби паралельно підключити інші цифрові пристрої (до 128 од.) без необхідності змінювати фізичну топологію проводки.

Окрім контролю стану ґрунту, важливою складовою теплиці також є моніторинг атмосферних показників. До складу системи з моніторингу входять сенсори температури та відносної вологості повітря, а також датчики ультрафіолетового випромінювання.

Для збору даних про стан повітря в архітектуру інтегровано комбінований цифровий датчик температури та вологості, модель GY-BME280/BMP280 (рисунок 2.3). Він об’єднує в собі два різні фізичні сенсори та власний мікроконтролер для обробки сигналів.



Рисунок 2.3 - багатофункціональний датчик GY-BME280/BMP280

Вимірювання температури здійснюється за допомогою термістора з негативним температурним коефіцієнтом. Він показує дані завдяки тому, що його електричний опір нелінійно зменшується при нагріванні середовища. Вимірювання вологості повітря базується на використанні вологопоглинаючого полімерного діелектрика, розміщеного між двома мікроелектродами, і коли змінюється кількість водяної пари в повітрі, полімер поглинає або віддає молекули води, що призводить до зміни його діелектричної проникності та, відповідно, електричної ємності конденсатора.

Перевагою такого модуля є його цифровий інтерфейс - вбудований у датчик мікроконтролер вимірює аналогові зміни опору та ємності, проводить їх калібрування і формує цифрові дані. Raspberry Pi ініціює зв'язок, надсилаючи стартовий імпульс LOW, після чого датчик відповідає серією мікросекундних імпульсів, які кодують 40-бітний пакет інформації, у якому містяться цілі та дробові

значення вологості, температури, а також контрольна сума для перевірки цілісності даних. Використання такого цифрового протоколу дозволяє підключати сенсор безпосередньо до портів GPIO I2C без використання додаткових АЦП.

Система також здійснює моніторинг рівня освітлення за допомогою аналогового датчика УФ-випромінювання GUVA-S12SD (рис. 3.4), тому що процеси фотоморфогенезу (формоутворення) у рослинах прямо залежать від наявності світла певної довжини хвилі. У ньому використовується напівпровідниковий фотодіод, чутливий до хвиль, довжина яких лежить у діапазоні від 240 до 370 нм. Принцип його дії базується на внутрішньому фотоелектричному ефекті: фотони ультрафіолетового світла, потрапляючи на p-n перехід напівпровідника, передають свою енергію електронам, переводячи їх у зону провідності. Це призводить до генерації мікрострумів, сила яких є прямо пропорційною інтенсивності падаючого випромінювання.



Рисунок 3.4 - датчик ультрафіолетового випромінювання GUVA-S12SD

Оскільки струми, згенеровані фотодіодом, надзвичайно малі, а саме порядки наноампер, модуль оснащується підсилювачем. Він перетворює мікроструми на

стабільну аналогову напругу (зазвичай у діапазоні 0–3 В), яку система здатна зафіксувати та обробити, але, на відміну від цифрового датчика вологості повітря, сигнал з УФ-сенсора є неперервним, тому його необхідно підключати до АЦП.

Для керування поливом та освітленням використовуються модулі електромагнітних реле (рис. 3.5). У них є вбудовані електромагнітні котушки, які при подачі струму замикають контакти силового ланцюга (для забезпечення відмовостійкості керуючої плати було обрано релейні модулі з додатковою оптоізоляцією). Сигнал 3.3 В від Raspberry Pi запалює світлодіод всередині оптрона, а фототранзистор відкриває транзисторний ключ, що живить котушку від окремої лінії 5 В. У релейних модулях є три контакти: нормально замкнений (NC — Normally Closed), загальний (COM — Common) та нормально розімкнений (NO — Normally Open). Для даної системи було обрано схему підключення через контакти COM та NO. Провідник, що йде від блоку живлення (для безпосереднього живлення роботи реле), фіксувався у клемі COM, а провідник, що веде до зволожувача, насоса або лампи — у клемі NO.

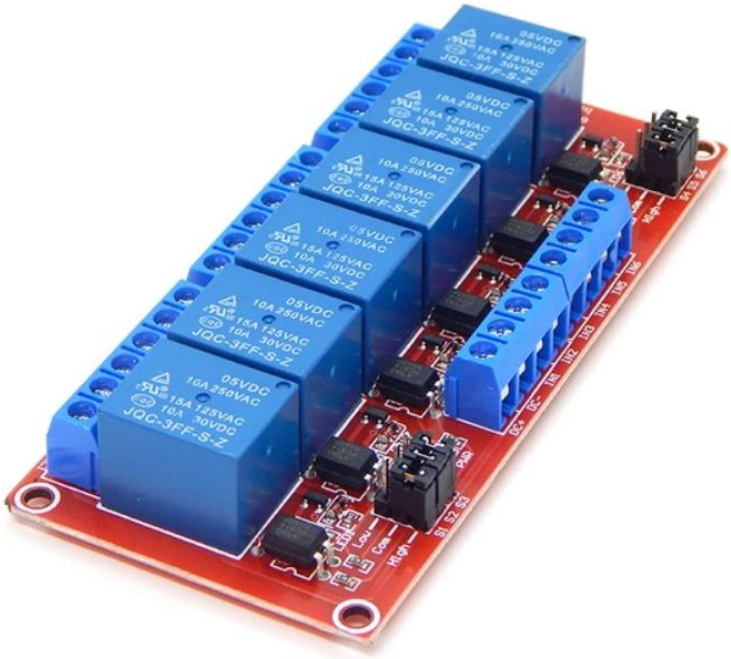


Рисунок 3.5 - шестиканальний модуль реле з робочою напругою 5В

Для поливу обрано заглибні мініатюрні помпи постійного струму (DC Water Pumps). Вони компактні та живляться від напруги 5 В, тому підходять для інтеграції в ізольовані системи резервуарів з водою. Окремою задачею у процесі реалізації стала інтеграція специфічних периферійних пристроїв - фітолампи та мініатюрного зволожувача повітря. Особливість тут полягала в тому, що вони спроєктовані для живлення від стандартних зовнішніх USB-адаптерів з напругою 5 В. Довелося провести фізичну «модифікацію» ліній живлення пристроїв. Стандартний кабель USB містить кілька жил, серед яких для безпосередньо живлення використовуються дві основні: позитивний потенціал VBUS (5 В) та загальний провідник GND, «земля». Для створення потрібного ланцюга і підключення їх до реле ізоляція кабелю знімалася, дві жили розрізалися, утворюючи вільні кінці, а вже їх можна підключати до реле.

Також при проєктуванні апаратної частини потрібно підібрати підсистему живлення. Процесор Raspberry Pi побудований на транзисторній логіці, яка оперує напругою 3.3 В, виходи GPIO видають рівно 3.3 В у стані HIGH і, що навіть важливіше, вони здатні безпечно приймати на вхід сигнал не більше 3.3 В. Але значна частина використаної периферії (наприклад, котушки електромагнітних реле та насоси) вимагає для своєї роботи напруги живлення 5 В. У нас виникає проблема: якщо пристрій, що живиться від 5 В, отримає від GPIO порту сигнал напругою 3.3 В, він просто не працюватиме. Тому в цій архітектурі живлення модулів і передача даних розділені і живляться від двох різних джерел. Датчики вологості та АЦП живляться від безпечної лінії 3.3 В, а для управління 5-вольтовими реле, насосами, зволожувачем, та УФ-лампою ми використовуємо окремий блок живлення. Окрім вихідної напруги блоку живлення, важливо також враховувати піковий струм системи, а він розраховується як сума максимальних струмів усіх споживачів. Струм при паралельному підключенні рахується сумарно, а тому:

- сумарний струм утримання котушок реле при одночасному ввімкненні кількох каналів (приблизно 70 мА на один канал; для 6 каналів — 420 мА;
- пусковий струм водяних помп постійного струму (в момент старту ротора струм може сягати 400-500 мА на одну помпу, для трьох помп — близько 1,5А;
- зволожувач та УФ-лампа потребують кожен по 2 А, тобто сумарно 4.

“Силова” частина системи споживає сумарно ~6 А на піку, тому було обрано блок живлення (перетворювач з 220В) на 5В та 10А. Як виглядає схема повністю зібраної системи показано на рис. 2.6

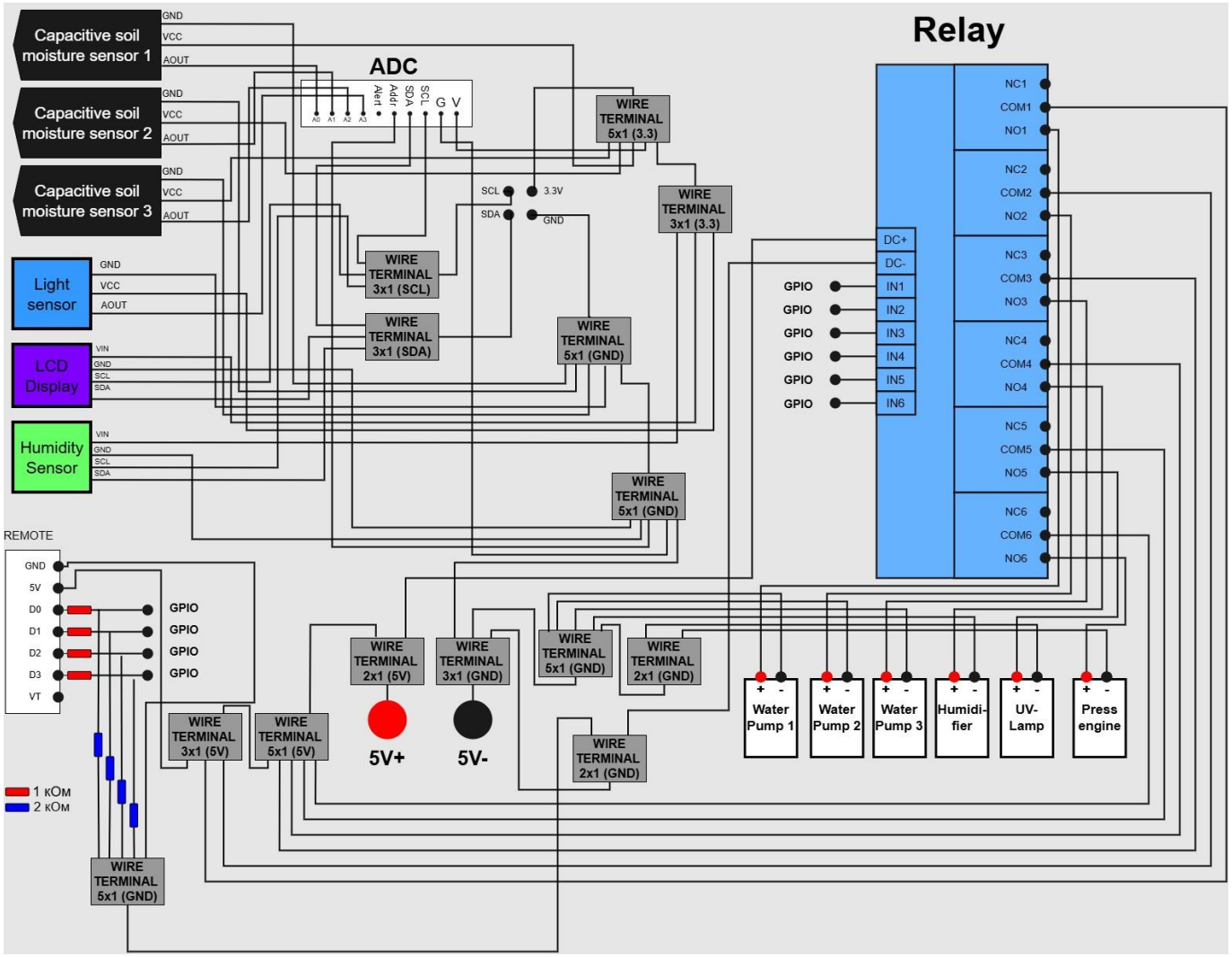


Рисунок 2.6 - принципова схема зібраної системи

2.3 Проєктування структури бази даних

Забезпечення автономної роботи подібних систем вимагає створення системи зберігання даних. Запис параметрів мікроклімату безпосередньо у коді програми є неправильним підходом і вважається «дурним тоном» у програмуванні, оскільки він позбавляє систему гнучкості та унеможливлює зміну налаштувань без перекомпіляції проєкту. Замість цього ми використовуємо базу даних, а саме SQLite. На відміну від класичних клієнт-серверних баз даних (як PostgreSQL чи MySQL), SQLite є embedded-базою даних. Уся інформація зберігається у єдиному

локальному файлі на накопичувачі, і це повністю усуває мережеві затримки при зверненні до неї, мінімізує споживання оперативної пам'яті та спрощує процес резервного копіювання і розгортання системи.

Проєктування інформаційної моделі здійснювалося за парадигмою Code-First з використанням фреймворку Entity Framework Core (EF Core). Такий підхід дозволяє описати структуру бази даних у вигляді класів мовою C# із використанням базових принципів об’єктно-орієнтованого програмування, після чого фреймворк автоматично генерує відповідні таблиці та зв'язки у файлі SQLite. В основі інформаційної моделі лежать дві головні таблиці, які пов'язані між собою відношенням «один-до-багатьох» (1:N) - PlantProfile та ActivePot.

PlantProfile виконує роль довідника. Вона зберігає еталонні параметри мікроклімату, які є ідеальними для конкретного біологічного виду рослини. Структура таблиці включає наступні поля:

- Id - унікальний ідентифікатор профілю;
- Name - назва виду рослини;
- MinSoilMoisture та MaxSoilMoisture - межі вологості ґрунту, при виході за які система ініціює або зупиняє полив;
- MinAirHum та MaxAirHum - допустимий діапазон відносної вологості повітря для керування зволожувачем;
- MinLightLevel - мінімальний поріг інтенсивності природного освітлення, нижче якого вмикається штучна фітолампа;
- WakeUpTime та SleepTime — поля типу TimeSpan, що задають добові біоритми рослини. Вони визначають часові межі «світлового дня», протягом яких системі дозволено вмикати фітолампи, імітуючи природний цикл сонця.

ActivePot (Активний горщик) є описом реального об'єкта — конкретного горщика, підключеного до системи. Ця сутність пов'язує біологічні вимоги з апаратною конфігурацією мікрокомп'ютера. Її структура складається з:

- Id - унікальний ідентифікатор горщика (або слота, який зчитує дані з горщика, що одне і те саме);
- PlantProfileId - зовнішній ключ, що вказує, який саме біологічний профіль з таблиці PlantProfile застосовується до цього горщика;
- PlantName - користувацька назва конкретної висадки;
- RelayPin - конфігураційний параметр, що зберігає номер фізичного піна GPIO на платі Raspberry Pi, до якого підключено реле водяної помпи саме цього горщика;
- SensorChannel - номер аналогового каналу на модулі АЦП, до якого підключено ємнісний датчик вологості цього ж горщика;
- PlantedDate — дата висадки рослини, що дозволяє системі (або користувачу) аналізувати вік культури та прогнозувати етапи її вегетації.

2.4 Розробка алгоритмів керування системою

Враховуючи специфічну апаратну інфраструктуру та спроєктовану базу даних, у ході роботи було створено алгоритм, який забезпечить їхню взаємодію. Програмна логіка мікрокомп'ютера повинна працювати як фоновий сервіс, що циклічно збирає дані, звіряє їх з еталонними значеннями та приймає рішення про комутацію навантажень.

Враховуючи високу інерційність біологічних та термодинамічних процесів (ґрунт, наприклад, не висихає за секунду, а температура не падає миттєво),

використання Event-Driven-архітектури (на основі подій) через апаратні переривання є недоцільним. Замість цього, алгоритм базується на архітектурі циклічного опитування із заданим інтервалом (під час тестувань, наприклад, 1 раз на хвилину). Це знижує навантаження на процесор Raspberry Pi та мінімізує кількість звернень до шини I2C і бази даних SQLite.

Головний алгоритм системи складається з наступних послідовних етапів:

1. Ініціалізація - підключення до бази даних SQLite через Entity Framework Core, конфігурація пінів GPIO на виведення (OUTPUT) та встановлення їх у безпечний стан (LOW).
2. Зчитування з таблиці ActivePot списку всіх підключених горщиків разом із їхніми прив'язаними біологічними профілями (PlantProfile).
3. Для кожного ActivePot система звертається до модуля АЦП (ADS1115) за вказаним SensorChannel та зчитує поточні аналогові значення вологості ґрунту і рівня УФ-випромінювання.
4. Порівняння отриманих фізичних даних з ідеальними параметрами з PlantProfile.
5. Надсилання сигналів HIGH/LOW на відповідні піни для увімкнення чи вимкнення помп, ламп та зволожувачів.
6. Асинхронне призупинення виконання потоку (Task.Delay) до наступного циклу.

Найважливішою частиною алгоритму є логіка керування поливом через насоси. На перший погляд, можна увімкнути помпи, якщо поточна вологість менша за задану, і вимкнути, якщо більша. Проте на практиці такий підхід призводить до явища «брязкоту» виконавчих механізмів. Коли вологість ґрунту коливається рівно на межі спрацювання, система генеруватиме десятки команд увімкнення/вимкнення реле за хвилину, що швидко виведе з ладу і реле, і

електродвигун помпи. Для вирішення такої проблеми в алгоритм введено механізм гістерезису або зони нечутливості алгоритму. Для цього в базі даних передбачено два окремі поля: MinSoilMoisture (поріг увімкнення) та MaxSoilMoisture (поріг вимкнення). Логіка працює наступним чином:

- Якщо поточна вологість падає нижче MinSoilMoisture, алгоритм подає сигнал HIGH на відповідний пін — помпа вмикається.
- Далі система ігнорує дрібні коливання вологості. Помпа продовжує працювати доти, доки вологість не досягне верхньої межі MaxSoilMoisture.
- Тільки після досягнення MaxSoilMoisture алгоритм подає сигнал LOW, і полив припиняється. Рослина знову починає споживати воду, і цикл повторюється лише тоді, коли вологість знову впаде до мінімуму.

Логіка керування фітолампою є двофакторною і залежить як від показників сенсора, так і від системного часу Raspberry Pi. Вона спирається на поля WakeUpTime, SleepTime та MinLightLevel. Спочатку алгоритм перевіряє, чи входить поточний час у дозволене вікно «світлового дня» (між WakeUpTime та SleepTime). Якщо час поза межами цього вікна, система примусово вимикає реле лампи, імітуючи ніч і дозволяючи рослині перейти у фазу темнового дихання. Якщо ж час входить у дозволене вікно, система перевіряє показники УФ-датчика. Якщо природного світла в кімнаті достатньо (значення вище за MinLightLevel), лампа не вмикається задля економії електроенергії. Якщо ж зафіксовано похмуру погоду або сутінки, алгоритм активує фітолампу, компенсуючи нестачу інсоляції.

Розроблена алгоритмічна модель також враховує можливість відмови обладнання. Під час зчитування даних по шині I2C або 1-Wire можуть виникати втрати пакетів через електромагнітні перешкоди. Для вирішення цієї проблеми код містить блоки обробки виключень (try-catch). У разі втрати зв'язку з датчиком алгоритм не зупиняє всю програму (щоб не постраждали інші підключені горщики), а логує помилку і примусово переводить реле проблемного горщика у

стан LOW (вимкнено). Логіка гарантує, що при поломці датчика помпа не затопить приміщення водою через нескінченний цикл поливу.

Детальні графічні блок-схеми розроблених алгоритмів, що візуалізують послідовність прийняття рішень та розгалуження процесів, винесені у Додаток Б даної кваліфікаційної роботи.

2.5 Візуальна модель системи з використанням UML

Для наочної демонстрації складних взаємодій ми використовуємо уніфіковану мову моделювання (UML), щоб створити абстрактну модель системи, яка відображає як вимоги користувача, так і внутрішню логіку роботи модулів. Першим етапом моделювання є створення діаграми прецедентів, тобто опис можливостей системи з точки зору áкторів. У даному проєкті головним актором є користувач (Адміністратор системи), а другорядним - час (системний таймер), який ініціює автоматичні події.

Основними функціональними завданнями (use cases), які розв'язує система, є:

- Створення та модифікація біологічних профілів рослин у базі даних.
- Прив’язка фізичних горщиків до конкретних профілів та апаратних ресурсів (пінів GPIO).
- Моніторинг показників вологості та освітлення у реальному часі.
- Автоматичне керування виконавчими механізмами (полив, освітлення) на основі алгоритму гістерезису.
- Логування помилок у разі виходу датчиків з ладу.

Діаграма на рис. 2.7 візуалізує межі системи та визначає обсяг робіт, які має виконувати програмне забезпечення.

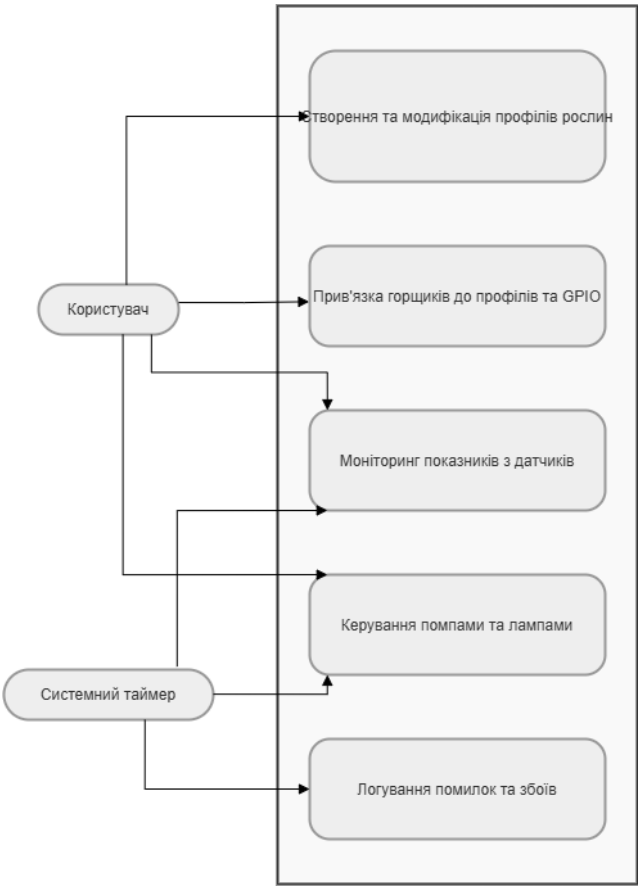


Рисунок 2.7 – діаграма прецедентів

Опис динамічної поведінки системи та логіку передачі сигналів можна побачити на діаграмі послідовності (рис. 2.8). Вона демонструє, як об’єкти системи взаємодіють між собою протягом одного циклу опитування, а сам процес можна розділити на наступні кроки:

1. Об'єкт-контролер ініціює запит до Бази даних для отримання списку активних горщиків.
2. Для кожного знайденого об'єкта контролер надсилає сигнал до модуля АЦП через шину I2C.
3. АЦП повертає оцифроване значення напруги.

4. Контролер виконує порівняння отриманих даних із пороговими значеннями з об'єкта PlantProfile.
5. Якщо умови алгоритму вимагають дії, Контролер надсилає сигнал до Об'єкта GPIO, який фізично замикає Реле.
6. Після завершення ітерації система переходить у стан очікування (Task.Delay).

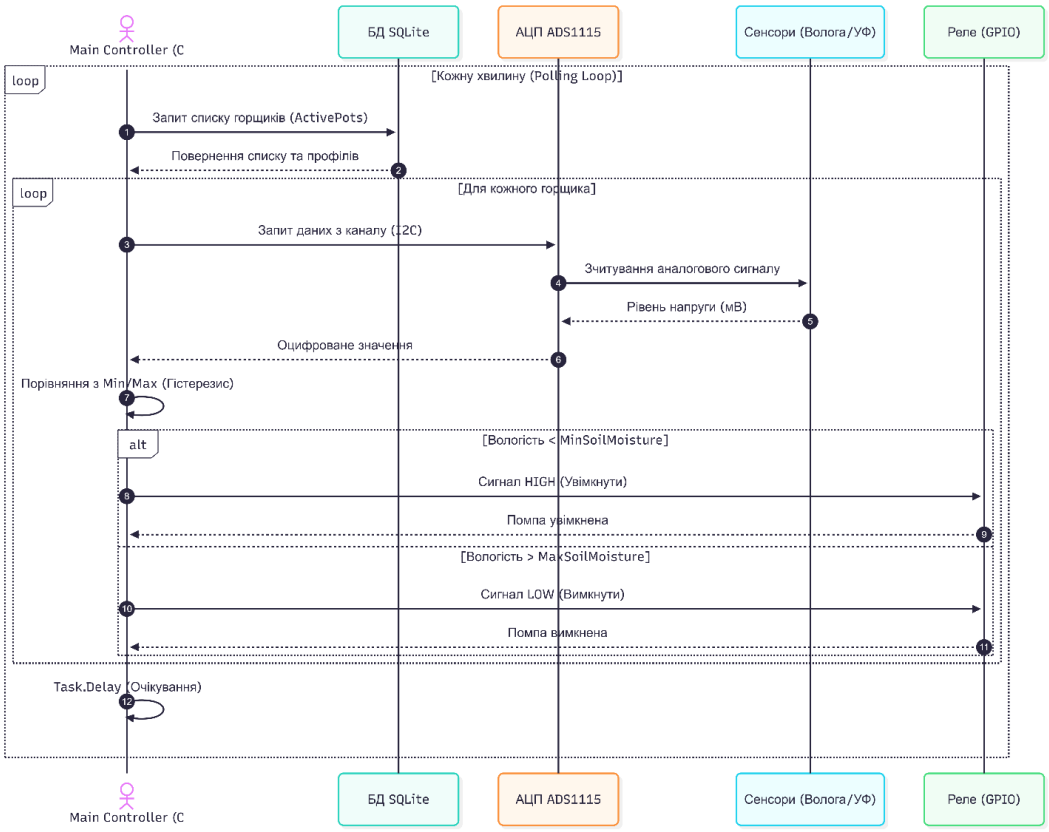


Рис 2.8 - діаграма послідовностей

2.6 Організація процесу запуску

Оскільки теплиця є автономною системою, будь-який програмний збій під час оновлення коду може призвести до незворотної втрати дослідницьких зразків. Організація безпечної та стабільної роботи системи на базі мікрокомп’ютера Raspberry Pi вимагає комплексного підходу, що охоплює як захист від

несанкціонованого доступу, так і впровадження надійної методології розгортання програмного забезпечення.

Віддалене розгортання було обрано в першу чергу через зручність розробки (редагування та компіляція коду безпосередньо на мікрокомп'ютері зайняла б надто багато часу). Написання коду, його налагодження та компіляція здійснюються на домашній робочій станції в середовищі Visual Studio, що доступно завдяки кросплатформеності .NET. Процес перенесення програмного забезпечення на Raspberry Pi робиться наступним чином:

1. Компіляція відбувається командою dotnet publish із зазначенням цільової архітектури linux-arm64, вона створює набір файлів, які не вимагають наявності повного SDK на самому мікрокомп'ютері, що зменшує навантаження на мікроконтролер і підвищує швидкість роботи.
2. Передача збірки здійснюється через протокол SFTP (Secure File Transfer Protocol). На відміну від застарілого FTP, цей протокол шифрує як облікові дані, так і сам вміст файлів, запобігаючи перехопленню коду в локальній мережі, що також впливає на безпеку.
3. Остаточне налаштування та запуск служби виконуються через термінал по протоколу Secure Shell, також відомий як SSH.

Окремим аспектом безпеки є захист від збоїв живлення. Оскільки раптове вимкнення може призвести до пошкодження SQLite-файлу, у програмній реалізації активовано режим запису Write-Ahead Logging. Це гарантує, що база даних зможе відновити свою цілісність навіть у разі критичного переривання процесу запису.

3. КОНСТРУКТИВНИЙ РОЗДІЛ: ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ

3.1 Програмна реалізація логіки керування системою мовою C#

Створення керуючого програмного забезпечення розпочалася зі створення програмного середовища. В якості цільової платформи було використано кросплатформений фреймворк .NET, що дозволило вести розробку та налагодження коду у зручному середовищі Visual Studio на базі ОС Windows, з подальшою компіляцією та розгортанням готових бінарних файлів на ARM-архітектурі мікрокомп'ютера Raspberry Pi під управлінням ОС Linux.

Архітектурно-програмний комплекс побудований за принципом модульності і розділений на три основні логічні рівні: рівень доступу до даних (Data Access Layer), рівень апаратної абстракції (Hardware Abstraction Layer) та ядро бізнес-логіки (Main Controller).

Для роботи з локальною базою даних SQLite був використаний фреймворк для об'єктно-реляційного відображення Entity Framework Core. У коді реалізовано клас контексту бази даних AppDbContext, який успадковується від базового класу DbContext. У ньому зареєстровані колекції DbSet<PlantProfile> та DbSet<ActivePot>, що відповідають за «мапінг» C#-класів на фізичні таблиці SQLite. Такий підхід дозволив повністю відмовитися від написання SQL-запитів. Наприклад, для отримання списку всіх активних горщиків разом із їхніми біологічними профілями використовується LINQ-запит з жадібним завантаженням (Eager Loading): var pots = await _context.ActivePots.Include(p => p.PlantProfile).ToListAsync();.

Оскільки стандартна бібліотека класів .NET не містить інструментів для взаємодії з фізичними пінами процесора, для керування GPIO було інтегровано офіційну бібліотеку від Microsoft - System.Device.Gpio. Завдяки їй у програмі створюється екземпляр класу GpioController, який бере на себе функцію низькорівневої маршрутизації сигналів. Під час ініціалізації системи програма зчитує з бази даних значення RelayPin для кожного горщика, відкриває відповідні фізичні контакти на Raspberry Pi (OpenPin) і задає їм режим роботи на виведення сигналу (PinMode.Output).

Для взаємодії з аналого-цифровим перетворювачем ADS1115 (до якого підключені ємнісні датчики вологості ґрунту та УФ-сенсори) була використана екосистема Iot.Device.Bindings. Цей NuGet-пакет дозволяє ініціалізувати обмін даними по шині I2C. У коді створюється об'єкт конфігурації I2cConnectionSettings із зазначенням апаратної адреси модуля на шині, після чого програма отримує можливість зчитувати значення напруги у мілівольтах з конкретного каналу перетворювача, викликаючи відповідний асинхронний метод.

Центральним елементом програми є клас-контролер, що реалізує описану в пункті 2.4 алгоритмічну модель циклічного опитування. Враховуючи, що система повинна працювати в режимі 24/7, критично важливим було недопущення блокування основного потоку виконання (Thread Blocking), що призвело б до стовідсоткового завантаження ядра процесора і перегріву мікрокомп'ютера.

Для цього головний цикл реалізований як асинхронна фонова задача з використанням ключових слів async та await. Алгоритм послідовно проходить через колекцію ActivePots, опитує датчики по шині I2C, порівнює отримані мілівольти з пороговими значеннями з об'єкта PlantProfile та подає команди PinValue.High або PinValue.Low на відповідні реле. Після завершення ітерації для всіх горщиків потік викликає метод await Task.Delay(), переходячи у режим очікування на заданий інтервал часу (наприклад, 60 секунд). Під час цього "сну" програма фактично

звільняє ресурси процесора Raspberry Pi, дозволяючи операційній системі виконувати інші фонові процеси.

Окрім цього, на програмному рівні було реалізовано механізм перехоплення виключень (try-catch блоки). Якщо під час опитування датчика по шині I2C виникає апаратна помилка (наприклад, IOException через поганий контакт провода), програма не "падає", а локалізує помилку, виводить повідомлення в лог консолі та примусово встановлює реле відповідного горщика у безпечний вимкнений стан, переходячи до обробки наступної рослини.

Фрагменти вихідного коду програми, що демонструють ініціалізацію контексту бази даних та реалізацію асинхронного циклу опитування, наведені у Додатку В.

3.2 Збірка, налаштування та тестування

Перехід від проєктування до експлуатації вимагає проведення комплексного тестування, починаючи від збірки електронних компонентів, конфігурації операційної системи мікрокомп'ютера до розгортання скомпільованого коду та калібрування сенсорів.

В якості програмної основи для Raspberry Pi було обрано офіційну операційну систему Raspberry Pi OS Lite на базі дистрибутиву Debian. Вибір версії "Lite" без графічного інтерфейсу користувача є прийнятним рішенням і навіть містить низку переваг. Відсутність віконного менеджера вивільняє значний обсяг оперативної пам'яті та ресурсів процесора, що важливо для забезпечення безперебійної роботи фонових систем автоматизації. Взаємодія з мікрокомп'ютером здійснювалася без підключення монітора та клавіатури через протокол безпечного віддаленого доступу SSH. Для забезпечення апаратної взаємодії з модулем АЦП

ADS1115 на рівні ядра операційної системи було примусово активовано апаратну підтримку шини I2C.

Оскільки розроблений програмний комплекс написаний мовою C#, для його виконання на Linux-архітектурі ARM64 було завантажено та інстальовано середовище виконання .NET Runtime. Це дозволило перенести скомпільовану збірку з робочої станції безпосередньо на мікрокомп'ютер без необхідності компіляції вихідного коду на самому Raspberry Pi.

Для того, щоб система керування теплицею була повноцінно автономною, розроблена програма не повинна залежати від ручного запуску користувачем. Наприклад у разі відключення електроенергії та подальшого перезавантаження Raspberry Pi, логіка мікроклімату має відновлюватися автоматично. Для вирішення цього інженерного завдання програму було інтегровано в систему ініціалізації Linux — systemd. Було створено спеціальний конфігураційний файл служби (daemon), у якому прописано шлях до виконуваного файлу .NET-додатка, параметри перезапуску у разі програмного збою (Restart=always) та затримку старту (RestartSec=10). Такий підхід перетворив додаток на системну фонову службу.

Перед початком тестування в реальному ґрунті було проведено обов'язкове апаратне калібрування ємнісного датчика вологості. АЦП видає значення напруги в мілівольтах, і тому їх необхідно було спочатку виміряти, а потім лінійно апроксимувати показники від 0% до 100% для подальшого використання. Процес калібрування відбувався у два етапи:

1. Фіксація «сухого» еталона: датчик розміщувався у повністю сухому старому ґрунті, і система фіксувала максимальне значення зворотної напруги.
2. Фіксація «вологого» еталона: датчик занурювався у склянку з водою (еквівалент 100% вологості), після чого фіксувалося мінімальне значення напруги. Отримані граничні значення були внесені в конфігураційний блок

коду, де за допомогою математичної функції лінійної інтерполяції мілівольти перетворюються на зручні для сприйняття відсотки вологості.

Фінальний етап апробації передбачав тестування системи з підключеним реальним навантаженням (водяними помпами та фітолампами) на тестовому горщику з субстратом. Під час тестування перевірялася коректність роботи механізму гістерезису, описаного в алгоритмічній моделі. Штучно створювалися умови прискореного висихання ґрунту. Було емпірично доведено, що при досягненні нижнього порогу (MinSoilMoisture), наприклад, 30%, Raspberry Pi успішно ініціює зміну логічного стану відповідного GPIO-піна на HIGH. Оптопара релейного модуля фіксує сигнал, замикає силовий ланцюг, і помпа починає подачу води. Завдяки високій частоті опитування (через асинхронний метод Task.Delay), система вчасно фіксує досягнення верхнього порогу (MaxSoilMoisture), наприклад 60%, і переводить реле у стан LOW, не допускаючи перезволоження землі.

Тестування підсистеми освітлення продемонструвало коректну реакцію логіки на системний час (синхронізація Raspberry Pi з серверами точного часу) та показники УФ-датчика: фітолампа успішно ігнорувала затінення в нічний період та миттєво активувалася при штучному перекритті природного світла у дозволений період.

Таким чином, практична апробація підтвердила високу надійність спроєктованої апаратно-програмної архітектури. Система здатна стабільно функціонувати в автономному режимі, взаємодіяти з базою даних, зчитувати фізичні показники середовища та комутувати силові навантаження без втручання оператора.

3.3 Розробка веб-інтерфейсу користувача та організація клієнт-серверної взаємодії

Для забезпечення зручної взаємодії користувача з автоматизованою системою було спроєктовано та реалізовано графічний інтерфейс користувача. З огляду на вимоги до кросплатформеності та незалежності від операційної системи, від розробки класичних десктопних або мобільних додатків було вирішено відмовитися на користь архітектури веб-додатка (Web Application).

У такій клієнт-серверній архітектурі мікрокомп'ютер Raspberry Pi виконує подвійну функцію: окрім низькорівневого опитування датчиків через шини GPIO/I2C, він одночасно виступає в ролі локального веб-сервера. Програмний код безперервно прослуховує вказаний мережевий порт. Користувачу не потрібно завантажувати жодного додаткового програмного забезпечення на свій смартфон чи комп'ютер. Для доступу до панелі керування теплицею достатньо підключитися до тієї ж локальної Wi-Fi мережі, що й Raspberry Pi, відкрити будь-який сучасний веб-браузер та ввести локальну IP-адресу мікрокомп'ютера. Браузер (клієнт) надсилає HTTP-запит до Raspberry Pi (сервера), який, у свою чергу, генерує HTML-сторінку на основі актуальних даних з бази SQLite та показників датчиків, і повертає її користувачу.

Головна сторінка веб-інтерфейсу розділена на кілька інформаційних блоків, що дозволяє користувачеві оцінити стан усієї системи завдяки інтуітивно-зрозумілому інтерфейсу (рис. 3.1 та 3.2). У верхній частині розміщено панель загальної телеметрії мікроклімату. Вона в режимі реального часу відображає агреговані показники середовища: температуру повітря, відносну вологість та рівень ультрафіолетового освітлення. Ці дані формуються на основі постійного опитування цифрового датчика температури/вологості та аналогового УФ-сенсора, оцифрованого через модуль ADS1115.

Основна робоча область інтерфейсу відведена під динамічне відображення фізичних слотів (горщиків). Кожен запис з таблиці ActivePot локальної бази даних SQLite автоматично рендериться у вигляді окремої інтерактивної картки. Картка містить вичерпну конфігураційну та поточну інформацію про конкретну рослину:

- Назву застосованого біологічного профілю (наприклад, "Томат балконний").
- Системний номер горщика.
- Апаратний слот — ідентифікатор фізичного піна (наприклад, GPIO 17), до якого підключено реле відповідної помпи.
- Поточний стан вологості ґрунту (у сирих значеннях АЦП або відсотках), що зчитується відповідним ємнісним датчиком.

Незважаючи на те, що система є повністю автономною, веб-інтерфейс надає можливість ручного перехоплення керування. Кожна картка обладнана кнопкою «Полити (Авто-доза)», натискання якої ініціює примусове замикання відповідного реле через GPIO-контролер в обхід автоматичного розкладу, що є зручним для проведення технічного обслуговування або внесення рідких добрив.

Окремим важливим компонентом веб-додатка є механізм конфігурації нових пристроїв, що візуалізує закладену на етапі проєктування (пункт 2.3) логіку бази даних. Для додавання нового об'єкта догляду передбачено кнопку «Додати горщик».

При взаємодії з ним сервер викликає модальне вікно (форму), яке напряму взаємодіє з довідковою таблицею PlantProfile. Користувачу пропонується розкривний список (Dropdown), дані для якого динамічно підтягуються з БД за допомогою Entity Framework. Користувач обирає необхідний біологічний шаблон (наприклад, той самий "Томат балконний"), після чого система створює новий запис у таблиці ActivePot, автоматично призначаючи йому вільні піни GPIO та канали

АЦП, і нова картка миттєво з'являється на головному дашборді. Така реалізація веб-інтерфейсу робить систему інтуїтивно зрозумілою для кінцевого споживача.

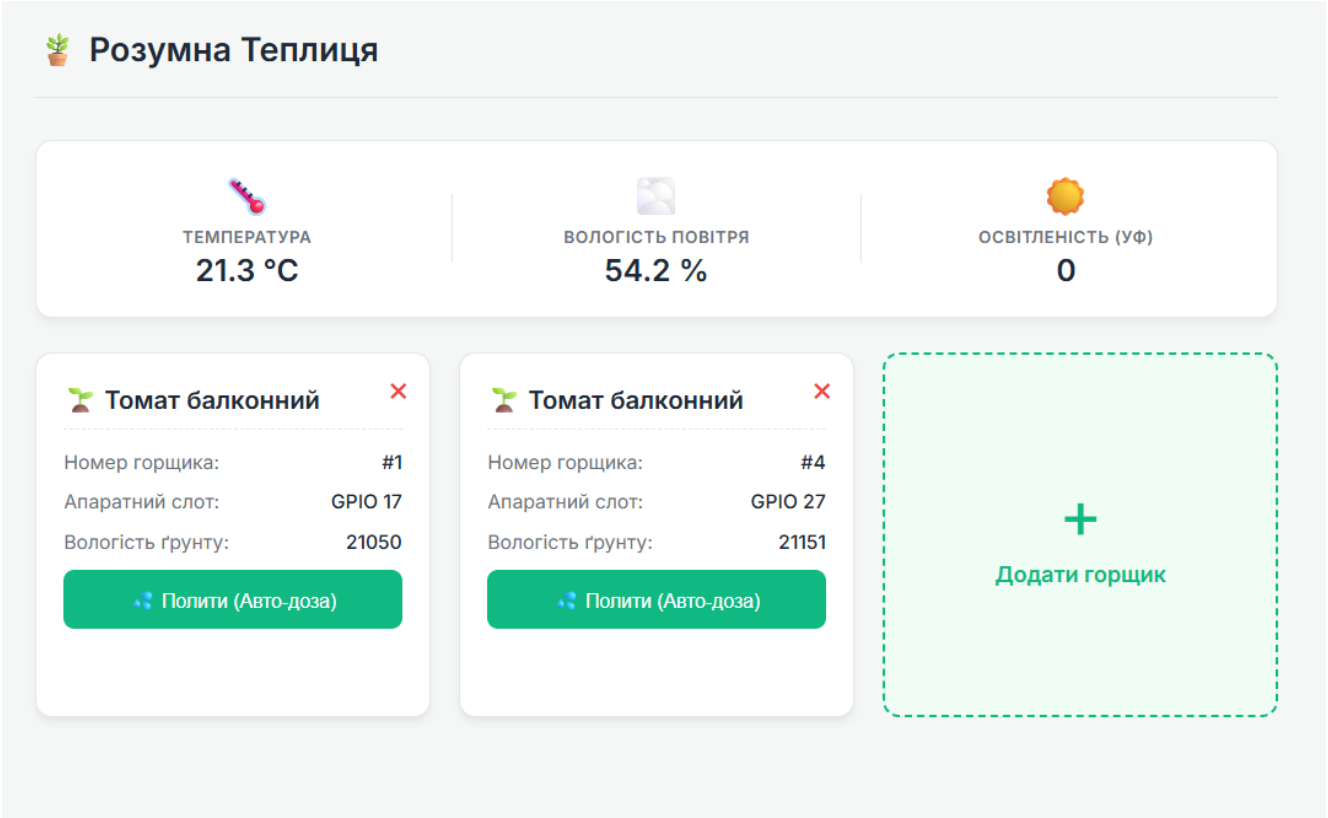


Рисунок 3.1 - Інтерфейс додатку

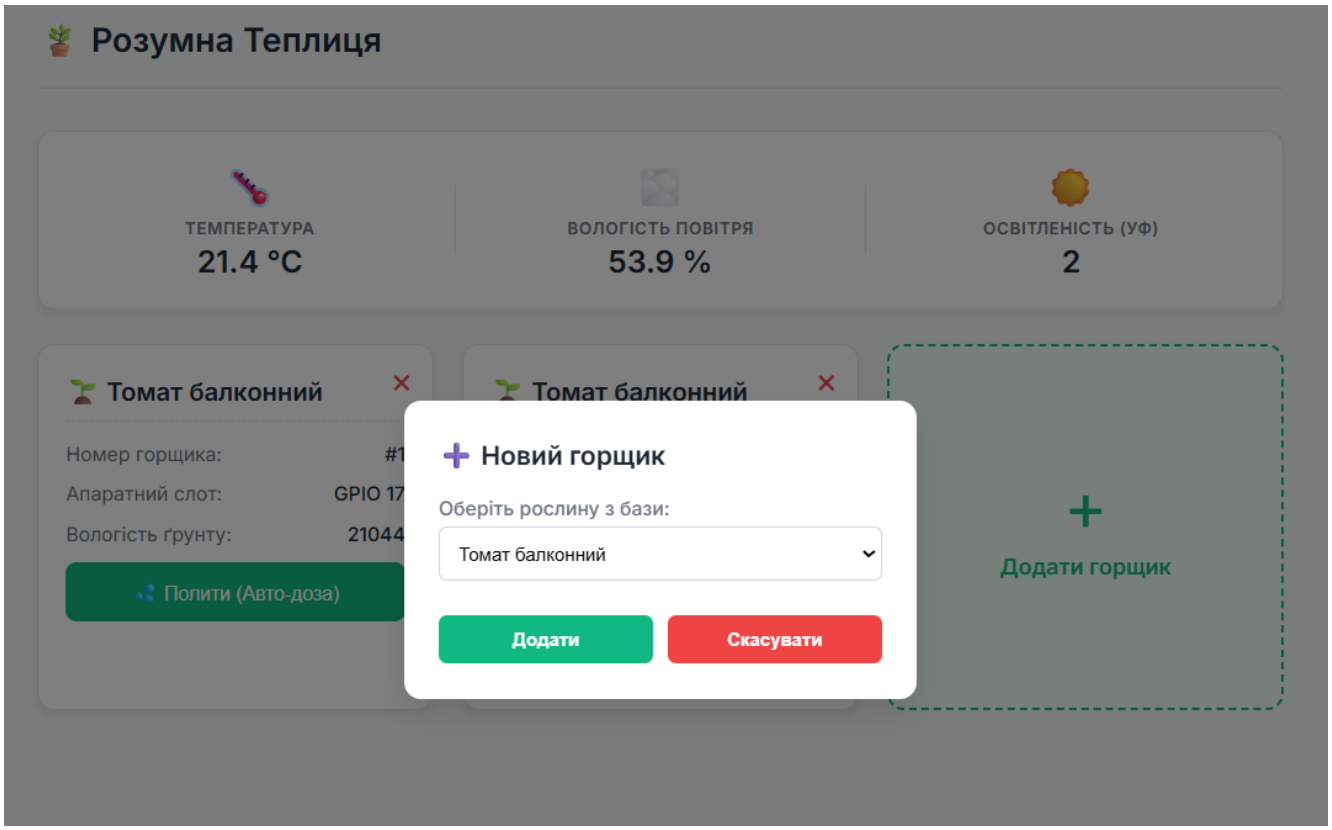


Рисунок 3.2 - Інтерфейс додатку

3.4 Напрями вдосконалення та перспективи розвитку системи

Розроблений комплекс має значний потенціал для подальшої модернізації. На основі попереднього тестування та користування було визначено кілька потенційних напрямів розвитку, що дозволять перетворити прототип на інтелектуальну екосистему промислового рівня.

Найбільш цікавим вектором розвитку бачиться інтеграція модулів машинного зору. Завдяки наявності інтерфейсу CSI на платі Raspberry Pi, систему можна доповнити камерою високої роздільної здатності. Використання бібліотек аналізу зображень, як-от OpenCV, або інтеграція з хмарними AI-сервісами дозволить автоматично розпізнавати ранні стадії захворювань рослин за зміною пігментації листя або виявляти появу комах-шкідників. Програма зможе надсилати сповіщення про необхідність біологічного втручання ще до того, як проблема стане видимою для ока людини.

Враховуючи необхідність забезпечення безперебійної роботи системи в умовах нестабільного електропостачання, важливим кроком є інтеграція системи енергетичної автономії. Архітектура живлення може бути доповнена контролером заряду та акумуляторною збіркою на базі літій-залізо-фосфатних елементів, а інтеграція сонячних панелей дозволить створити повністю енергонезалежний комплекс, здатний функціонувати у віддалених локаціях без підключення до централізованих мереж, чи взагалі, доповнити систему до рівня автономних дорогих аграрних систем.

Подальша модернізація програмної частини не виключає перехід від локального веб-інтерфейсу до повноцінної мобільної екосистеми. Використання фреймворку .NET MAUI дозволить створити нативний мобільний додаток для iOS та Android, використовуючи вже написану логіку на C#. Окрім візуалізації даних, у такий додаток доцільно інтегрувати систему Push-сповіщень через, наприклад, Telegram-ботів, що інформуватимуть користувача про критичне зниження рівня води у резервуарах або аномальні температурні стрибки.

Для розширення споживчих можливостей систему планується доповнити підтримкою протоколу MQTT. Це дозволить інтегрувати автоматизовану теплицю в єдиний контур керування розумним будинком (наприклад, через платформу Home Assistant). Таким чином, теплиця зможе взаємодіяти з іншими пристроями: наприклад, закривати автоматичні штори в кімнаті при фіксації надмірного сонячного випромінювання або вмикати витяжну вентиляцію приміщення при підвищенні вологості повітря в зоні вирощування.

Таку систему можна модернізувати і для масового виробництва. Використання замість дорогого мікроконтролера Raspberry Pi більш дешевого ESP32 S3, mosfet-транзисторів замість блоків реле, проектування пропрієтарної друкованої плати із усіма потрібними чіпами та сенсорами, що масово виготовлятиметься на профільних фабриках – усе це дозволяє значно, у рази (якщо не десятки разів) зменшити вартість.

ВИСНОВОК

У даній кваліфікаційній роботі було розв'язано поставлене науково-технічне завдання, що полягало у розробці автоматизованої системи керування мікрокліматом у закритому приміщенні. Огляд предметної області зазначив, що існує ймовірність того, що традиційні методи ручного догляду за рослинами можуть бути малоефективними, ресурсоємкими та нести ризики, зумовлені людським фактором. Порівняльний огляд виявив, що рішень посередині між аматорськими мікроконтролерами та промисловими системами не існує. Цим була викликана ідея розробки системи, яка б поєднувала обчислювальні можливості, гнучке налаштування та відкриту архітектуру.

Рішенням під час проєктування стало використання необхідної кількості периферійних пристроїв, а на відміну від класичних хмарних рішень, розроблений комплекс обробляє дані локально. Таким чином, була досягнута повна автономність системи. Було спроєктовано комплексну апаратну архітектуру, яка успішно нівелює обмеження обраного мікрокомп’ютера, як, наприклад, відсутність вбудованого аналого-цифрового перетворювача. Крім того, реалізована система ізольованого живлення та комутації навантажень через модулі реле забезпечила електричну безпеку процесора мікроконтролера.

Програмну основу комплексу склала розроблена інформаційна модель бази даних на основі SQLite, яка у зв’язці із фреймворком Entity Framework дозволила створити структуру зберігання порівняльних профілів рослин. Сама ж керуюча логіка, написана мовою C# та розгорнута як системна служба Linux, продемонструвала ефективність, тоді як використання асинхронної моделі програмування знизило навантаження на процесор до можливого мінімуму.

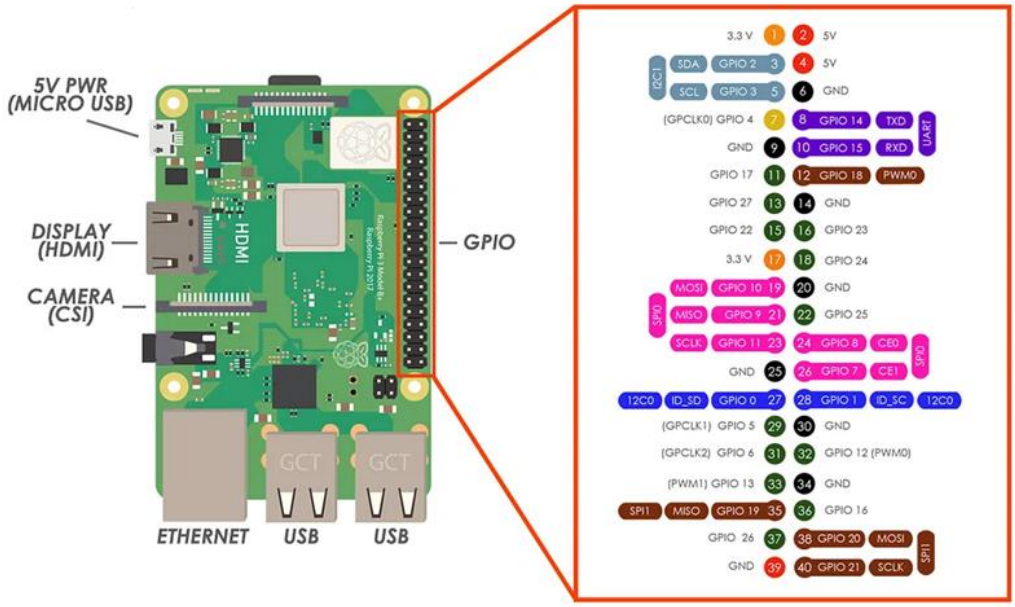
Також варто відзначити створення повноцінного веб-інтерфейсу користувача. Завдяки реалізації клієнт-серверної архітектури на базі ASP.NET Core,

система дозволяє здійснювати моніторинг показників оточуючого середовища та керування теплицею з будь-якого пристрою без встановлення додаткового програмного забезпечення. Тестування в реальних умовах підтвердило коректність роботи всіх вузлів. Своєчасний автоматичний полив відбувається згідно з встановленими налаштуваннями, системи освітлення та зволоження повітря функціонують також коректно.

На основі отриманих результатів, можна вважати, що створений апаратно-програмний комплекс має потенціал для подальшого масштабування. Пріоритетними напрямами його вдосконалення є впровадження технологій комп'ютерного зору для ранньої діагностики хвороб рослин, глибока інтеграція в екосистеми розумного дому через протокол MQTT та здешевлення виробництва. Можна сміливо стверджувати, що мета роботи досягнута, а розроблений проєкт демонструє ефективність поєднання сучасних IT-технологій з інженерними рішеннями для автоматизації агротехнічних процесів.

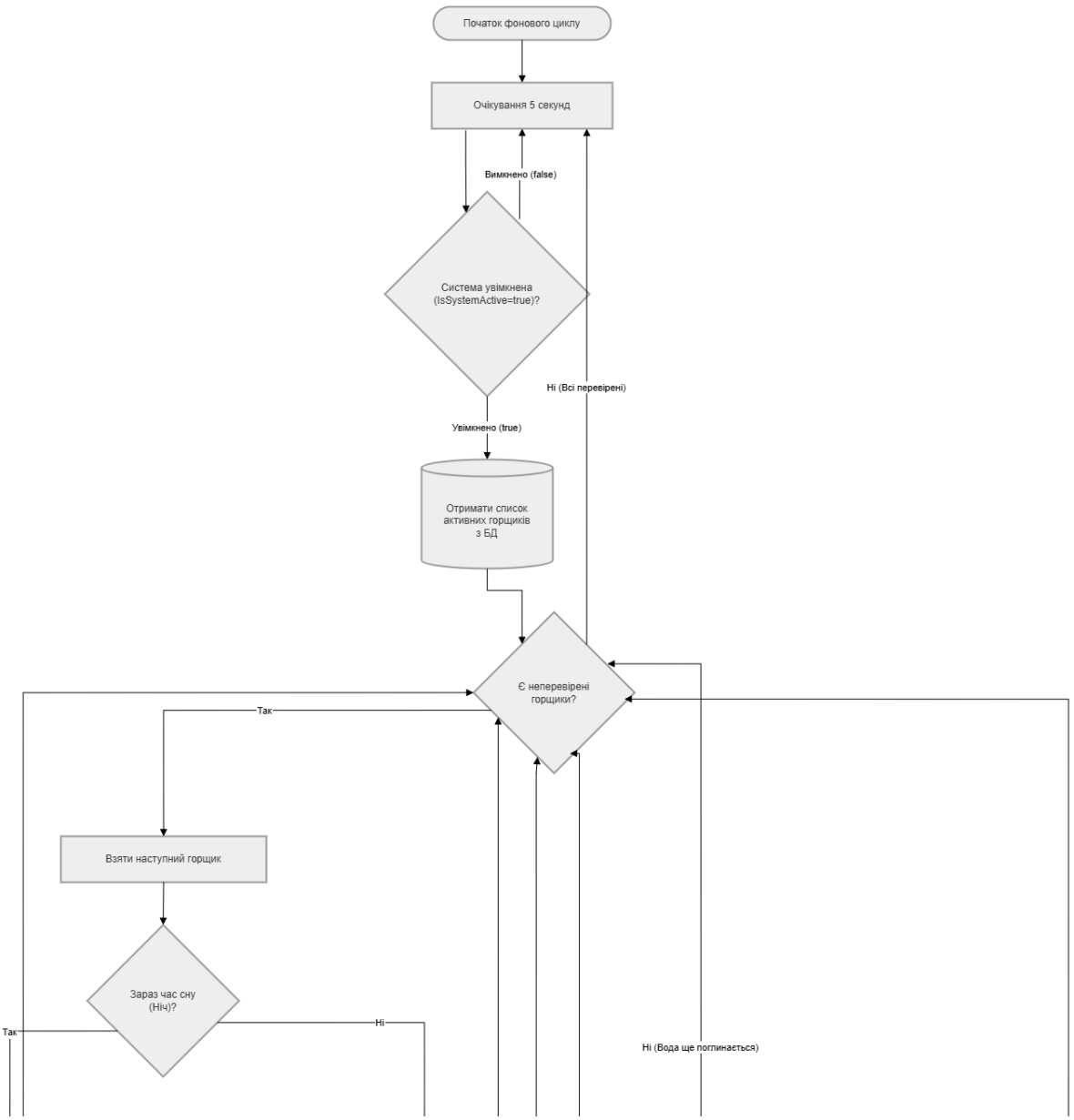
ДОДАТКИ

Додаток А – схема Raspberry PI і таблиця з призначенням портів у описаній системі



Фізичний номер порта	Назва порта	Призначення/підключений пристрій
1	3.3 V	Живлення датчиків та інших пристроїв, що потребують низької потужності
6	GND	Заземлення
3	I2C SDA	Лінія шини I2C, зв’язок із сенсорами
5	I2C SCL	Лінія шини I2C, синхронізація даних
11	GPIO 17	Піни для керування насосами
13	GPIO 27	
15	GPIO 22	
16	GPIO 23	Керування УФ-лампи
18	GPIO 24	Керування зволожувачем
22	GPIO 25	Керування кнопкою на зволожувачі

Додаток Б


Початок фонового циклу
Очікування 5 секунд
Вимкнено (false)
Система увімкнена (IsSystemActive=true)?
Ні (Всі перевірені)
Увімкнено (true)
Отримати список активних горщиків з БД
Є неперевірені горщики?
Так
Взяти наступний горщик
Зараз час сну (Ніч)?
Так
Ні
Ні (Вода ще поглинається)

Список використаних джерел

1. Microsoft Learn. Документація з мови C# (версії 9.0–12.0). URL: https://learn.microsoft.com/en-us/dotnet/csharp/

2. Microsoft Learn. Огляд платформи .NET та середовища виконання. URL: https://learn.microsoft.com/en-us/dotnet/core/introduction

3. Microsoft Learn. Документація з Entity Framework Core. URL:https://learn.microsoft.com/en-us/ef/core/

4. Microsoft Learn. Бібліотека System.Device.Gpio для роботи з інтерфейсами введення-виведення. URL: https://learn.microsoft.com/en-us/dotnet/iot/

5. Raspberry Pi Foundation. Raspberry Pi 4 Model B documentation and hardware specifications. URL: https://www.raspberrypi.com/documentation/

6. SQLite. SQL As Understood By SQLite: Official Documentation. URL: https://www.sqlite.org/docs.html

7. Texas Instruments. ADS1115: Ultra-Small, Low-Power, 16-Bit Analog-to-Digital Converter with Internal Reference. Datasheet. URL: https://www.ti.com/product/ADS1115

8. Bosch Sensortec. BME280 Combined humidity and pressure sensor. Technical Datasheet. URL: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/

9. Iot.Device.Bindings. Documentation for .NET IoT Device Bindings library. URL: https://github.com/dotnet/iot/tree/main/src/devices

10. Systemd.service. Systemd unit configuration documentation for Linux service management. URL: https://www.freedesktop.org/software/systemd/man/systemd.service.html

11. Троєлсен Е., Джепікс Ф. C# 9 та платформа .NET 5: основні принципи та технології програмування. 10-те вид. 2021. 1218 с.

12. Албахарі Дж. C# 9.0. Довідник. Повний опис мови. 2021. 1040 с.

13. Ріхтер Дж. CLR via C#. Програмування на платформі Microsoft .NET Framework 4.5 мовою C#. 4-те вид. 2019. 896 с.

14. Мартін Р. Чиста архітектура. Мистецтво розробки програмного забезпечення. 2018. 352 с.

15. Smart Greenhouse Automation using IoT and C#. International Journal of Scientific & Engineering Research. 2019.

16. Stack Overflow. Developer Survey 2024: Trends in IoT and C# development. URL: https://survey.stackoverflow.co/

17. GitHub IoT Repository. Community patterns for .NET on Raspberry Pi. URL: https://github.com/dotnet/iot

18. Монк С. Raspberry Pi. Збірник рецептів. Рішення та приклади програмування. 2016. 520 с.

19. Habr. Програмування мікрокомп’ютерів Raspberry Pi мовою C# та .NET. URL: https://habr.com/

20. Sommerville I. Software Engineering. 10th Edition. Pearson, 2015. 811 p.

21. Фаулер М. Шаблони корпоративних програмних додатків. 2003. 544 с.

22. Ларман К. Застосування UML та шаблонів проєктування. 3-тє вид. 2004. 736 с.

23. Ептон Е., Халфакрі Г. Raspberry Pi: посібник користувача. 2014. 280 с.

24. Buyya R., Dastjerdi A. V. Internet of Things: Principles and Paradigms. Morgan Kaufmann, 2016. 378 p.

25. Фрімен А. ASP.NET Core: розробка професійних веб-застосунків. 2022. 1040 с.

26. Клірі С. Конкурентне програмування на C#: асинхронні, паралельні та багатопотокові рішення. 2-ге вид. 2020. 256 с.

27. Дейт К. Дж. Вступ до систем баз даних. 8-ме вид. 2018. 1328 с.

28. Немет Е., Снайдер Г., Хейн Т. UNIX та Linux: керівництво системного адміністратора. 5-те вид. 2020. 1168 с.

39. Шерц П., Монк С. Практична електроніка для винахідників. 4-те вид. 2017. 1056 с.

30. Fraden J. Handbook of Modern Sensors: Physics, Designs, and Applications. Springer, 2016. 758 p.