Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Мікроконтролерна система моніторингу росту рослин з використанням AWS
Назва теми

КвРКІ 200236.20.02.11 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-20-2 _____ Х. Р. Лозова
Підпис, Ініціали, прізвище

Керівник _____ В. В. Яцків
Підпис, дата, Ініціали, прізвище

Нормоконтролер _____ І. О. Засорнова
Підпис, дата, Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис, Ініціали, прізвище

«10» _червня_ 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 10 " 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Лозовій Христині Романівній
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Мікроконтролерна система моніторингу росту рослин з використанням AWS

Керівник проекту (роботи) Яцків В. В., д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Теоритичні основи досліджуваної системи

Проєктування програмно-технічного засобу для системи моніторингу росту рослин з використанням AWS

Програмно-апаратна реалізація мікроконтролерної системи моніторингу росту рослин з використанням AWS

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

Архітектура ПЗ проєкту

Архітектура ПЗ для мікроконтролерної системи

Апаратне забезпечення проєкту

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Засорнова І. О., доцент кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання «_10_»_____01_____2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2024	виконано
3	Робота над розділом 1 – теоритичні основи досліджуваної системи	01.03.2024	виконано
4	Робота над розділом 2 – проектування програмно-технічного засобу	01.04.2024	виконано
5	Робота над розділом 3 – програмно-апаратна реалізація мікроконтролерної системи моніторингу росту рослин з використанням AWS	29.04.2024	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	виконано
7	Попередній захист ВКР	30.05.2024	виконано
8	Захист ВКР на засіданні ЕК	Червень 2024 року	

Студент _____ Х. Р. Лозова
Підпис Ініціали, прізвище

Керівник роботи _____ В. В. Яцків
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 200236.20.02.11 ПЗ	Пояснювальна записка	57		
			Графічні матеріали			
2		КвРКІ 200236.20.02.11 Е8	Архітектура ПЗ проєкту	1		
3		КвРКІ 200236.20.02.11 Е8	Архітектура ПЗ для Мікроконтролерної системи	1		
4		КвРКІ 200236.20.02.11 Е8	Апаратне забезпечення проєкту	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 200236.20.02.11 ВП		
Розробив		Лозова			Відомість проекту	Літера	Аркуш / Аркушів
Перевір.		Яцків				У	1 / 1
Н. контр.		Засорнова				ХНУ, КІ2-20-2	
Затв.		Говорущенко		10.06			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Мікроконтролерна система моніторингу росту рослин з використанням AWS».

Автор роботи: Лозова Христина Романівна.

Керівник роботи: Яцків Василь Васильович.

Пояснювальна записка: 57 с., 13 рис., 3 дод., 51 джерел.

Графічна частина: 3 креслення.

AWS, МІКРОКОНТРОЛЕРНА СИСТЕМА, МОНІТОРИНГ.

Метою дипломної роботи є розробка мікроконтролерної системи моніторингу росту рослин з використанням AWS, а також оцінка ефективності її застосування для забезпечення оптимальних умов вирощування рослин.

Об'єктом дослідження є функціювання мікроконтролерної системи моніторингу росту рослин.

Предметом дослідження є оцінка методів та технологій збору та обробки даних про ріст рослин з використанням AWS.

Під час проведення даного дослідження був використаний метод систематичного огляду літератури для вивчення і аналізу предметної області даного дослідження з текстових джерел інформації.

Підпис студента

04.06.2024

Дата

ЗМІСТ

ВСТУП

Сучасне сільське господарство стає все більш залежним від технологій, які дозволяють підвищити ефективність та продуктивність вирощування рослин. Однією з ключових задач у цьому контексті є моніторинг росту рослин та умов навколишнього середовища, що впливають на їх розвиток. Застосування мікроконтролерних систем у поєднанні з хмарними технологіями відкриває нові можливості для точного та оперативного збору, аналізу і використання даних з метою оптимізації процесу вирощування.

Актуальність теми зумовлена необхідністю підвищення ефективності сільськогосподарського виробництва в умовах змін клімату та зростаючого попиту на продовольство. Використання мікроконтролерних систем для моніторингу параметрів навколишнього середовища дозволяє в режимі реального часу отримувати точні дані, необхідні для прийняття оперативних рішень щодо управління умовами вирощування рослин. Інтеграція цих систем з хмарними платформами, такими як Amazon Web Services (AWS), забезпечує масштабованість та надійність зберігання і обробки даних, що є критичним для сучасних агротехнологій.

Дипломна робота присвячена розробці мікроконтролерної системи моніторингу росту рослин з використанням платформи AWS. AWS пропонує широкий спектр інструментів та сервісів для зберігання, обробки та аналізу даних, що дозволяє створити надійну та масштабовану систему для моніторингу та управління умовами вирощування рослин.

Мета дипломної роботи полягає у розробці та впровадженні мікроконтролерної системи, яка забезпечить автоматизований моніторинг параметрів навколишнього середовища (температура, вологість, освітленість, вологість ґрунту) з подальшою обробкою даних на платформі AWS для прийняття обґрунтованих рішень щодо управління умовами вирощування рослин.

Об'єктом дослідження є процеси моніторингу росту рослин та управління умовами навколишнього середовища в сільському господарстві.

Предметом дослідження є мікроконтролерні системи та хмарні технології, що використовуються для збору, передачі, зберігання та обробки даних про умови вирощування рослин.

У роботі буде розглянуто вибір апаратних засобів (мікроконтролер, сенсори), середовища програмування та інструментів розробки, а також проаналізовано можливі ризики та розроблено стратегії їх управління. Особлива увага буде приділена інтеграції системи з хмарною платформою AWS, що дозволить забезпечити надійність, масштабованість та оперативність обробки даних.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ

1.1 Аналіз предметної області і виявлення наявних проблем і завдань

На сьогоднішній час одним з основних завдань є підвищення якісного процесу вирощування рослин для отримання найкращого врожаю. Для цього використовується моніторинг росту рослин. Спостереження можуть бути зроблені вручну або автоматизовані, механізовані для спрощення роботи агрономів.

Хоча міське сільське господарство, за словами Джорджа Монбіо, можливо не в змозі повністю задовольнити наші щоденні потреби у овочах, воно стає все більш поширеним. Трав'яний сад не лише є красивим, але також має освітню та розслабляючу цінність. Розваги є тим, чого ми найбільше прагнемо, а отримання овочів стає приємним бонусом.

Основою ефективного вирощування є регулярний моніторинг росту рослин. Тому будемо використовувати мікроконтролерну систему моніторингу росту рослин з використанням AWS.

Мікроконтролерна система передбачає моніторинг різних показників і керування параметрами мікроклімату з метою забезпечення росту рослин.

До основних параметрів, які може контролювати система моніторингу, можна віднести [1]:

– освітлення – регулювання рівня підсвічування або затемнення рослин;

– водопостачання – необхідно керувати процесом поливу рослин;

– температуру – для унеможливлення замерзання чи перегріву рослин;

– циркуляцію повітря та вологість – закрите приміщення теплиці спричиняє підвищення рівня вологості та не достатню кількість вуглекислого газу та кисню для рослин залежно від часу доби.

Потрібно здійснювати одночасне регулювання цих показників для забезпечення кращого росту рослин.

Для цього використовують такі підсистеми:

– вентиляція – відкривання та закривання вікон, ввімкнення та вимкнення вентиляторів;
– зрошення – регулярна подача води відповідно до графіку; дозування поживних;
– речовин – аналізуючи ґрунт можна здійснювати розподіл поживних речовин по зрошувальній системі.

Для ефективного регулювання параметрів мікроклімату теплиці усіма цими підсистемами потрібно керувати одночасно об’єднавши їх у одну велику систему, котра дозволить оптимізувати їх роботу.

Ми можемо експериментувати з різними умовами освітлення, режимами живлення та комбінаціями рослин. Незалежно від того, які спроби ми підніматимемо, важливо здійснювати вимірювання росту рослин і порівнювати результати. Один з важливих показників цього є розмір зеленої зони. За допомогою Raspberry Pi та кількох рядків коду Python ми можемо повністю автоматизувати цей процес. Крім того, ми бажаємо зберегти отримані дані в хмарному сховищі для подальшого аналізу. AWS слугує такою хмарою для нас. Сервіс AWS IoT (Інтернет речей) відповідатиме за передачу всіх даних від Pi до AWS.

Рисунок 1.1 – Конфігурація AWS

AWS – інструмент конфігурації(див. рисунок 1.1), використовується для оцінювання, перевірки та аналізу конфігурації своїх ресурсів в AWS, локальному середовищі та інших хмарах [2].

Amazon Web Services (AWS) було створено компанією Amazon.com. Основним засновником Amazon.com є Джефф Безос [3].

Amazon Web Services (AWS) представляє собою найрозгалуженішу та найпоширенішу хмарну платформу у світі, яка надає понад 200 повнофункціональних сервісів з центрами обробки даних по всьому світу. Мільйони клієнтів, серед яких є швидкозростаючі стартапи, великі корпорації та провідні державні установи, обирають AWS для зниження витрат, збільшення гнучкості та прискорення впровадження інновацій [4].

AWS має більше можливостей, ніж будь-який інший хмарний провайдер. Від обчислень та сховища даних до машинного навчання та Інтернету речей. Це означає, що переміщення програм в хмару та створення нових стає швидшим і простішим.

Мікроконтролерна система (МКС) – це зібрана в єдине ціле сукупність взаємодіючих інтегральних схем цифрової логіки та аналогових ланцюгів, організована у обчислювальну або в керуючу систему з мікропроцесором (мікроконтролером) як вузлом обробки інформації [5]. Мікроконтролер - це невеликий електронний пристрій, який програмується для використання у керуючих пристроях, системах передачі даних та управління технологічними процесами.

1.2 Порівняльний аналіз переваг та недоліків існуючих рішень

Автори у [6] розглядають організацію системи керування процесами вирощування рослин у тепличному комплексі з використанням платформи Arduino, заснованої на мікроконтролері ATmega2560 (див. рисунок 1.2). Для передачі даних щодо результатів вимірювання параметрів мікроклімату теплиць

автори рекомендують використовувати Ethernet модуль, що потребує кабельного підключення до локальної мережі.

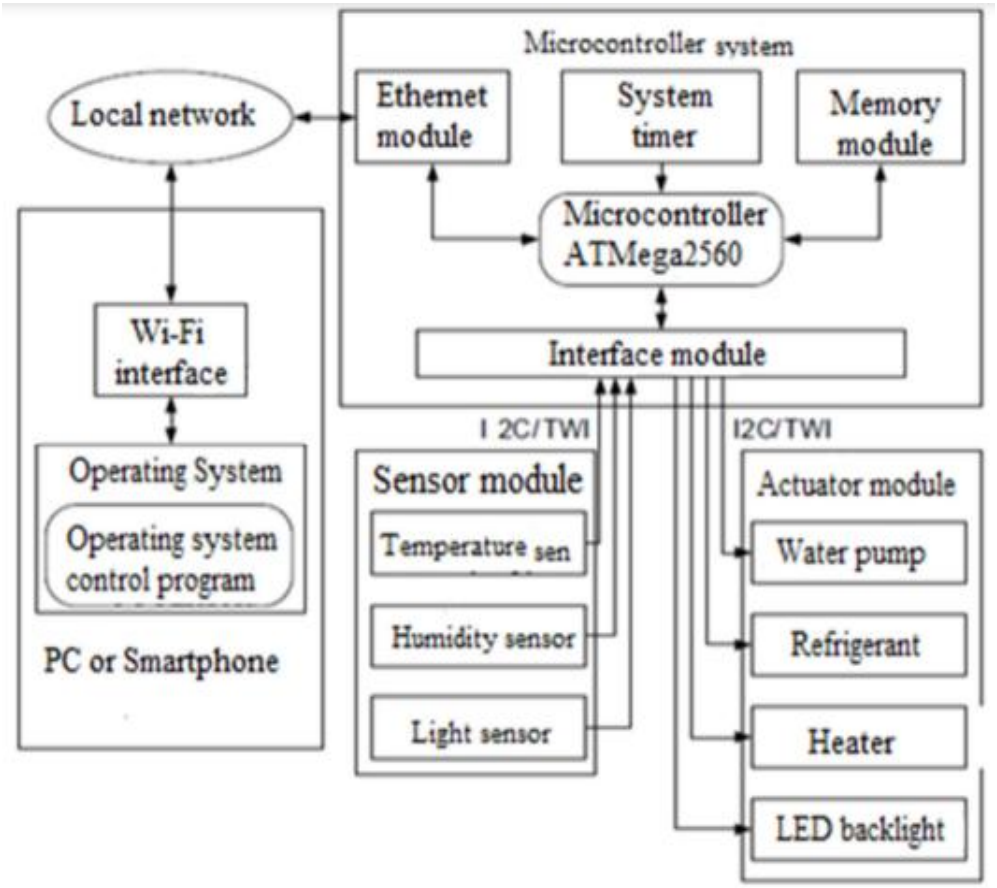


Рисунок 1.2 – Структурна схема системи управління мікрокліматом теплиці на основі Arduino Mega [6]

У статті [7] описується метод прийняття рішень щодо крапельного зрошення рослин у теплиці на основі інформації про їхній стан та параметри мікроклімату. Запропонований алгоритм передбачає початок процесу зрошення лише у випадку, коли товщина листка рослин зменшується принаймні на 20% від максимального значення. У статті також представлені результати впровадження цього методу у системі зрошення, побудованій на базі платформи Arduino, які демонструють зменшення витрат розчину для живлення рослин на 30%.

У роботі [8] пропонується метод підвищення ресурсоємності у вирощуванні рослин у тепличному комплексі. Контролери у цій системі керують потоками інформації, отримуючи дані від датчиків та передаючи їх до комп'ютера (див. рисунок 1.3). Керуючі сигнали надходять з комп'ютера до виконавчих механізмів (насосів, нагрівачів, вентиляторів тощо).

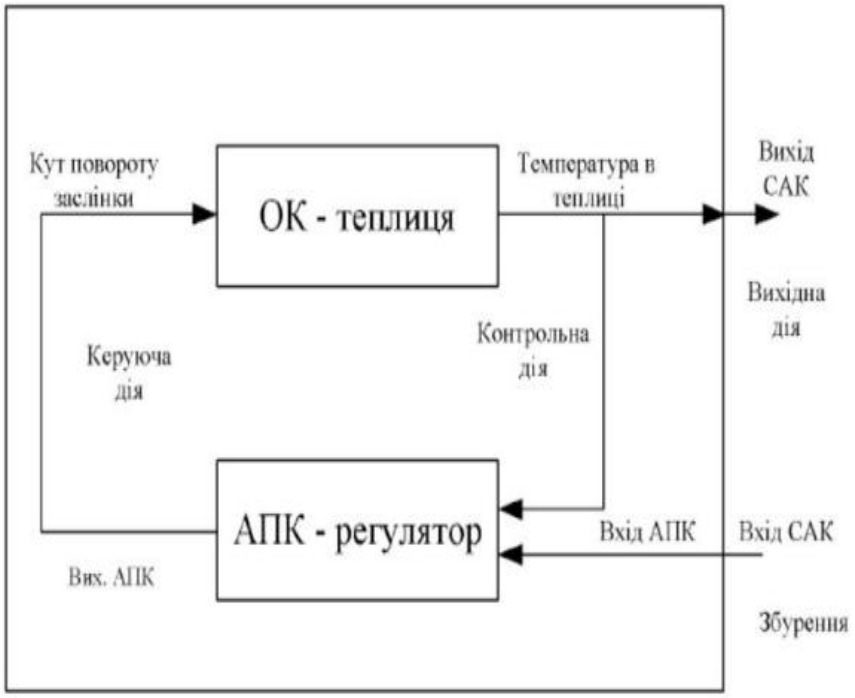


Рисунок 1.3 – Блок-схема алгоритму щодо реалізації методу прийняття рішень для крапельного зрошення рослин в теплиці [8]

Розроблений метод ґрунтується на використанні нейромережевих алгоритмів та програмного забезпечення для передбачення зовнішніх перешкод. Атмосферні метеорологічні умови є основними факторами, що можуть впливати на цільовий об'єкт. Керування здійснюється шляхом регулювання температури за допомогою повороту заслонки на певний кут. Недоліком такого підходу є необхідність використання комп'ютера для впровадження нейромережевих алгоритмів, що може значно збільшити вартість обладнання.

SoilSense - це система зрошення, яка ґрунтується на інтелектуальних датчиках, що дозволяє ефективно використовувати воду та підвищувати врожайність рослин. Для використання цієї системи достатньо просто встановити датчики в ґрунті теплиці та завантажити додаток на смартфон. SoilSense зменшує витрати води і збільшує врожайність [9]. Зовнішній вигляд апаратного забезпечення та веб-інтерфейс користувача SoilSense показані на рисунку 1.4.

Рисунок 1.4 – Зовнішній вигляд компонентів системи SoilSense

SoilSense - це сучасний комплекс, який об'єднує апаратне та програмне забезпечення. Полив рослин відбувається автоматично завдяки унікальному патентованому алгоритму. Будучи повністю цифровою, система SoilSense не потребує технічного обслуговування механізмів, що дозволяє виробникам сконцентруватися на оптимізації виробничих процесів. Однак недоліком цієї системи є обмежений набір датчиків, що не дає можливості контролювати всі параметри мікроклімату в тепличному комплексі.

Система Intelligrow допомагає виробникам сільськогосподарської продукції, які вирощують рослини у теплицях, досягти вищої врожайності протягом усього року. Ця система надає можливість віддалено контролювати стан врожаю через веб-браузер з будь-якого комп'ютера, планшета або телефону [10]. На рисунку 1.5 показаний додаток Intelligrow від компанії Autogrow.

Рисунок 1.5 – Інтерфейс користувача системи Intelligrow [10]

Один із недоліків системи Intelligrow полягає у тому, що апаратне забезпечення має відносно високу вартість. Крім того, для користування програмним забезпеченням потрібно сплачувати щомісячну плату.

1.3 Підходи до вирішення задачі за темою дослідження

Одним із підходів до створення мікроконтролерної системи для моніторингу росту рослин може бути доцільним використання мікроконтролерів Raspberry Pi або Arduino Uno. Вони можуть бути використані для збору даних про середовище та рівень росту рослин. Для програмування та управління такими мікроконтролерами можна використовувати мови програмування Python та C++, а

також фреймворки, які підтримують інтеграцію з хмарними сервісами, зокрема з AWS IoT.

1.4 Постановка задачі

Перелік задач:

– дослідити процеси та процедури функціонування системи моніторингу росту рослин з використанням AWS";

– провести теоретичний аналіз області моніторингу росту рослин;

– охарактеризувати структуру предметної області та базову модель;

– описати існуючі механізми реалізації, ідентифікувати проблеми та шляхи їх вирішення;

– визначити основні функції системи на основі проведених досліджень, сформулювати функціональні та нефункціональні вимоги, розробити модель функцій;

– підвести підсумки про необхідність розробки системи;

– сформувати об'єкт та мету для подальших досліджень.

Оцінити виконання поставлених завдань та розробити працездатну підсистему керування мікрокліматом у кіберфізичній системі "Розумний будинок", а також зробити висновки на основі результатів виконаної роботи.

1.5 Висновки

На основі аналізу предметної області та порівняльного аналізу існуючих рішень можна зробити висновок, що існують різні підходи до створення мікроконтролерних систем для моніторингу росту рослин. Зокрема, розглянуті системи, що базуються на платформах Arduino та Raspberry Pi, а також комерційні рішення, такі як SoilSense та Intelligrow.

Переваги і недоліки кожної з цих систем можуть вплинути на їх вибір для конкретного застосування. Наприклад, система SoilSense пропонує простий та ефективний спосіб зрошення рослин, але має обмежений функціонал датчиків. З іншого боку, Intelligrow надає можливість віддаленого контролю та моніторингу росту рослин, але може бути дорожчим у використанні через плату за програмне забезпечення.

Однак, незважаючи на існуючі рішення, є необхідність у подальшому дослідженні та розробці системи моніторингу росту рослин з використанням AWS. Поставлені завдання для цього включають аналіз процесів та процедур функціонування системи, вивчення теоретичних аспектів області моніторингу росту рослин, опис існуючих механізмів реалізації, визначення функцій системи та її вимог, а також розробка працездатної підсистеми керування мікрокліматом.

Отже, подальша робота має на меті розробку ефективної та працездатної системи моніторингу росту рослин з використанням AWS, яка відповідатиме вимогам та потребам користувачів у сільському господарстві.

2 ПРОЄКТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ ДЛЯ СИСТЕМИ МОНІТОРИНГУ РОСТУ РОСЛИН З ВИКОРИСТАННЯМ AWS

2.1 Розробка користувальницьких вимог

Розробка користувальницьких вимог до мікроконтролерної системи моніторингу росту рослин з використанням AWS базується на аналізі потреб кінцевих користувачів, таких як аграрії, садівники та науковці. Вимоги спрямовані на забезпечення зручності використання, ефективності та надійності системи. Основні користувальницькі вимоги включають [30]:

1) простота встановлення та налаштування:

a) система повинна бути легкою у встановленні та налаштуванні, не потребуючи глибоких технічних знань від користувачів. Для системи моніторингу росту рослин важливо забезпечити легкість встановлення та налаштування. Інтуїтивний інтерфейс користувача, автоматизація процесів, інтеграція з AWS IoT, мінімальні апаратні вимоги та наявність підтримки користувачів є ключовими аспектами, що роблять систему доступною та зручною для використання навіть для користувачів без глибоких технічних знань. Це дозволяє фермерам та агрономам зосередитись на вирощуванні рослин, залишаючи технічні аспекти надійній та легко керованій системі;

b) інструкції з налаштування повинні бути доступні у вигляді зрозумілих покрокових керівництв або відеоінструкцій;

2) зручний інтерфейс користувача:

a) інтерфейс повинен бути інтуїтивно зрозумілим і доступним через веб-браузер або мобільний додаток. Основою будь-якої діалогової системи є зручний інтерфейс "користувач-комп’ютер" [11]. Він забезпечує взаємодію між користувачем і процесом, що виконує певне завдання. З точки зору програмного забезпечення, інтерфейс включає дві компоненти (рисунок 2.1):

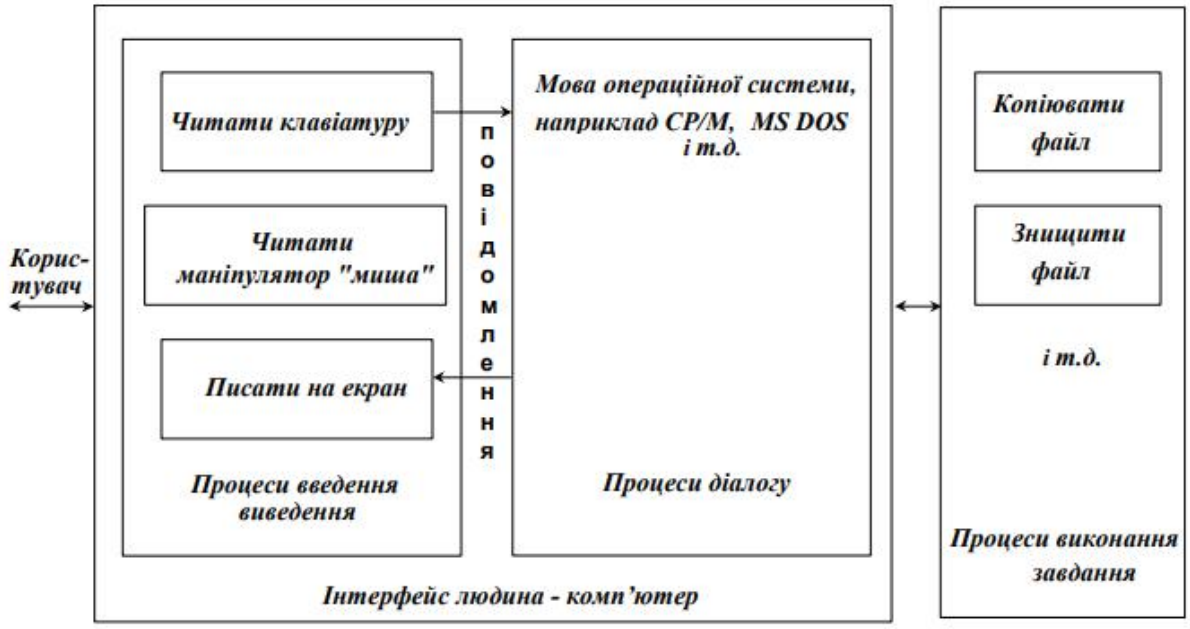


Рисунок 2.1 – Складові частини інтерфейсу людина-комп'ютер на прикладі операційної системи

b) користувачі повинні мати можливість легко переглядати дані про стан рослин, включаючи показники вологості ґрунту, освітленості, температури та інші важливі параметри [11];

3) моніторинг у реальному часі та сповіщення:

a) система повинна забезпечувати збір даних у реальному часі з регулярними оновленнями;

b) користувачі повинні отримувати сповіщення про критичні зміни параметрів, такі як надмірна сухість ґрунту або різке зниження температури;

4) доступ до історичних даних та аналітики:

a) система повинна зберігати історичні дані для подальшого аналізу та порівняння;

b) інструменти аналітики повинні дозволяти користувачам виявляти тенденції та робити прогнози щодо росту рослин;

c) інтеграція з AWS. Інтеграція додатків – це процес взаємодії різних програмних систем без втручання користувача. Сучасний дизайн додатків

забезпечує гнучкий обмін даними між додатками для підвищення ефективності, модульності та можливості повторного використання. Інтеграція додатків дає розробникам можливість створювати додатки, які повторно використовують існуючі послуги та системи. Таким чином вони можуть робити більше з меншими витратами на програмування. Крім того, взаємодія додатків у складних корпоративних робочих процесах значно полегшує операції з автоматизації [14];

5) дані повинні зберігатися у хмарному сховищі AWS, забезпечуючи високу доступність та безпеку:

а) система повинна використовувати сервіси AWS для обробки та аналізу даних, такі як AWS IoT, AWS Lambda та Amazon S3. AWS IoT сервіси, які допомагають підключатися до мільярдів пристроїв та керувати ними [15]. AWS Lambda - це сервіс безсерверних обчислень, який запускає програмний код у відповідь на певні події і відповідає за автоматичне виділення необхідних обчислювальних ресурсів. Перелік подій включає зміни в стані або оновлення, наприклад, коли користувач поміщає товар у кошик на веб-сайті інтернет-комерції. AWS Lambda можна використовувати для розширення можливостей інших сервісів AWS за допомогою спеціальної логіки або для створення власних серверних сервісів із застосуванням можливостей масштабування, продуктивності та безпеки AWS. AWS Lambda автоматично запускає програмний код у відповідь на різні події, які як HTTP-запити через Amazon API Gateway, зміна об'єктів у кошиках Amazon Simple Storage Service, оновлення таблиць в Amazon DynamoDB або зміна станів в AWS Step Functions. Lambda запускає код у високопродуктивному обчислювальному середовищі та повністю виконує адміністрування обчислювальних ресурсів [16]. Amazon S3 – це сервіс зберігання об'єктів, що пропонує найкращі в галузі показники продуктивності, масштабованості, доступності та безпеки даних. Клієнти будь-якої величини та з будь-якої промислової галузі можуть зберігати та захищати необхідний обсяг даних для практично будь-якого прикладу використання. Наприклад, для озер даних, хмарних додатків та мобільних додатків. Вигідні класи сховища та прості у

використанні інструменти адміністрування дозволяють оптимізувати витрати, організувати дані та точно налаштувати обмеження доступу відповідно до потреб бізнесу чи законодавчих вимог [17];

b) у тому числі обслуговування сервера та операційної системи, виділення ресурсів та автоматичне масштабування, розгортання виправлень коду та системи безпеки, а також моніторинг коду та ведення журналів. Від вас потрібно лише надати програмний код;

Інтеграція мікроконтролерної системи моніторингу росту рослин з AWS. Інтеграція додатків в AWS забезпечує ефективну взаємодію між ізольованими компонентами мікросервісів та безсерверних додатків з мінімальним обсягом коду. Це дозволяє зменшити вплив змін, спрощуючи оновлення та прискорюючи випуск нових можливостей. Використання AWS звільняє від необхідності написання зайвого коду та управління серверами, дозволяючи зосередитися на інноваціях [12].

Сервіси AWS забезпечують незалежне масштабування і стійкість до збоїв, що критично для безперервної роботи системи моніторингу росту рослин. Також, вони надають можливість конфіденційного обміну повідомленнями, адаптуючись до будь-якої пропускної здатності і забезпечуючи високу надійність та доступність [31].

Використання AWS IoT [24] для інтеграції мікроконтролерів Raspberry Pi або Arduino Uno дозволяє ефективно збирати та аналізувати дані про мікроклімат теплиці, оптимізуючи умови для росту рослин [22].

6) безпека та конфіденційність даних:

a) система повинна гарантувати безпеку передавання та зберігання даних;

b) доступ до даних повинен бути захищений механізмами автентифікації користувачів [13].

Автентифікація для мікроконтролерної системи моніторингу росту рослин з використанням AWS Автентифікація в комп'ютерних системах перевіряє

особистість користувача через унікальні характеристики (відбитки пальців), спеціальні пристрої (смарт-карти), або секретні паролі. Паролі є простим та дешевим методом, але потребують додаткових заходів безпеки. Смарт-карти забезпечують вищий рівень захисту завдяки одноразовим паролям і відповіді на виклики [13].

Для мікроконтролерної системи моніторингу росту рослин з використанням AWS важливо забезпечити надійну автентифікацію. AWS IoT надає захищене з'єднання та автентифікацію пристроїв, що гарантує безпеку та цілісність переданих даних [21].

7) можливість масштабування:

a) система повинна легко масштабуватися для підтримки різної кількості сенсорів та користувачів;

b) інфраструктура повинна бути гнучкою, щоб адаптуватися до зростання обсягів даних та збільшення кількості користувачів.

Вимоги, викладені вище, розроблені з урахуванням потреб кінцевих користувачів і спрямовані на забезпечення ефективної та зручної роботи мікроконтролерної системи моніторингу росту рослин з використанням AWS.

2.2 Функційні вимоги

2.2.1 Збір даних про ріст рослин

Система повинна здійснювати безперервний збір даних з використанням різних сенсорів [23]:

– сенсори повинні постійно вимірювати рівень вологості ґрунту та передавати ці дані до центрального мікроконтролера;

– температура повітря та ґрунту, сенсори повинні забезпечувати вимірювання температури повітря навколо рослин і температури ґрунту;

– освітленість, сенсори освітленості повинні реєструвати рівень освітленості, який отримують рослини протягом дня;

– рівень CO2, сенсори повинні вимірювати концентрацію вуглекислого газу в повітрі.

За фенологічних спостережень фіксують:

– появу і повні сходи;

– повну бутонізацію;

– повне цвітіння;

– початок і повне достигання.

У фазі повних сходів виділяють по три пробні майданчики на кожній ділянці у двох несуміжних повтореннях загальною площею 1 м². Їх помічають кілочками, а схему розміщення заносять до польового журналу [31].

Густоту стояння рослин визначають після останнього міжрядного обробітку та під час збирання врожаю вегетативної маси [32].

Оцінка стану посівів сортів. Стан посівів сортів оцінюють візуально у фази розвитку:

– повних сходів;

– повного цвітіння;

– повного достигання насіння;

– після несприятливих метеорологічних явищ.

Оцінюють у всіх повтореннях, враховуючи:

– висоту рослин;

– густоту стояння;

– вирівняність;

– дружність сходів;

– цвітіння рослин;

– ступінь пошкодження (ураження) шкідливими організмами;

– стійкість до вилягання;

– осипання насіння;

– придатність до механізованого збирання врожаю;

– вимірювання висоти рослин.

Висоту рослин вимірюють у фазі повного цвітіння мірною лінійкою (змієголовник молдавський у фазі повного цвітіння припиняє свій ріст) на виділених майданчиках[33].

Оцінка стану посівів за 9-баловою шкалою. Стан посівів сортів оцінюють за 9-баловою шкалою візуально:

– 9 балів – рослини у відмінному стані;
– 7 балів – добрий стан;
– 5 балів – середній стан;
– 3 бали – незадовільний стан;
– 1 бал – дуже поганий стан або повна загибель рослин.

Стійкість сортів до несприятливих метеорологічних явищ. Стійкість сортів змієголовника до несприятливих метеорологічних явищ визначають візуально з оцінкою: стійкий, нестійкий. За настання посухи або суховіїв спостерігають за станом рослин у повтореннях, а після припинення цих явищ оцінюють: стійкий, нестійкий сорт[34].

2.2.2 Обробка та зберігання даних на AWS

Система повинна забезпечувати надійну передачу та обробку зібраних даних на платформі AWS:

– передача даних: дані з сенсорів повинні передаватися на сервери AWS за допомогою бездротових технологій (Wi-Fi, LTE тощо);

– зберігання даних: дані повинні зберігатися у хмарному сховищі AWS, такому як Amazon S3 або Amazon DynamoDB, для забезпечення високої доступності та безпеки;

– обробка даних: використання AWS Lambda для обробки даних у реальному часі, включаючи перевірку на наявність аномалій та підготовку даних для візуалізації.

Щоб обробити великі дані, використовуються три основні методи аналізу: пакетний, інтерактивний та потоковий [20]. Великі дані можуть бути оброблені в пакетному режимі, коли вони вже зберігаються у сховищі. Це найчастіше використовується для агрегування даних та створення аналітичних звітів на їх основі [35].

Розглянемо приклад системи моніторингу певної кількості соціальних мереж. У такій системі дані моніторингу передаються з невеликою періодичністю у вигляді повідомлень від кожного підключеного модуля (тобто соціальної мережі). Для визначення обсягу даних за день необхідно підсумувати всі дані з кожного модуля соціальних мереж. Якщо підсумковий результат потрібен у вигляді щоденного (щотижневого, щомісячного тощо) звіту, найпростіше реалізувати таку систему, як показано на рисунку 2.2.

Рисунок 2.2 – Архітектура системи обробки даних з аналогічних джерел, побудована на основі розділення нереляційного сховища даних та агрегованих даних

Всі повідомлення можна зберігати в нереляційних сховищах табличного типу, таких як HBase, Cassandra або MongoDB. Кожен рядок таблиці міститиме часову мітку (час надходження або відправлення повідомлення), ідентифікатор відправника і саме повідомлення [35].

Для створення періодичних звітів за допомогою систем бізнес-аналітики (Business Intelligence, BI), таких як PowerBI або Tableau, або для відображення цієї інформації в браузері, дані в агрегованому вигляді повинні бути збережені в SQL базі даних. Розподіл на нереляційне і реляційне сховище необхідний через те, що система моніторингу збирає велику кількість даних від численних джерел. Це означає, що таблиця в реляційній БД може містити мільйони рядків: наприклад, 100 джерел із середньою кількістю повідомлень (20 на день) дадуть 200 мільйонів рядків на рік. Запити на вибірку даних з такої таблиці можуть займати багато часу. Додатково, часті запити на оновлення таблиці і запити від веб-порталу або користувачів для отримання звітів можуть створити серйозні проблеми для архітектури з однією базою даних [36].

Пакетна обробка з угрупованням і підсумовуванням результатів може бути виконана за допомогою інструментів, таких як Hadoop MapReduce або Apache Spark. Алгоритми угруповання можуть бути реалізовані за допомогою програм на Java (для MapReduce і Spark) або Python (для Spark) і виконуватись у кластері. Таким чином, розміри таблиці з агрегованими даними в SQL базі даних будуть значно меншими, ніж у таблиці з сирими даними. Наприклад, якщо зберігати годинне агрегування, то в SQL-таблиці буде в 12 разів менше рядків, ніж у NoSQL, а якщо добове – то в 288 разів. Це дозволить значно прискорити запити для клієнтів: вибірка з таблиці агрегатів буде простою і високопродуктивною, оскільки не вимагатиме угруповань у самому запиті [37].

Архітектура, що включає сховища як сирих, так і агрегованих даних, є дуже гнучкою і дозволяє створити інтелектуальну систему зберігання та агрегації даних. Вона може прогнозувати потік даних на основі історичної вибірки і виявляти вузькі місця, що є значним кроком вперед у порівнянні з простою агрегацією. Такий підхід забезпечує збереження даних у формах, які є найбільш зручними для аналізу різними сервісами: для побудови звітів дані зберігаються в агрегованому вигляді, а для Data Mining – у сирому [38].

2.2.3 Відображення та аналіз даних користувачем

Система повинна надавати користувачам зручний інтерфейс для доступу до зібраних даних, забезпечуючи ефективне використання інформації для аналізу та прийняття рішень [39].

Веб-інтерфейс та мобільний додаток:

– веб-інтерфейс, користувачі повинні мати можливість отримувати доступ до зібраних даних через веб-браузер. Веб-інтерфейс повинен бути інтуїтивно зрозумілим і підтримувати всі основні браузери; Основні функціональні можливості повинні включати.

– дашборди, індивідуальні та налаштовувані дашборди, які відображають ключові показники росту рослин.

– навігація, простий і зрозумілий інтерфейс для навігації між різними розділами системи, включаючи розділи з візуалізацією даних, аналітикою та звітами.

– фільтрація та пошук, можливість фільтрації даних за різними параметрами (наприклад, датою, типом рослин, місцем вирощування) та швидкий пошук потрібної інформації.

– мобільний додаток, система повинна мати мобільний додаток для доступу до даних з будь-якого місця. Основні функції мобільного додатку повинні включати:

– реальний час, доступ до даних в реальному часі з можливістю отримання миттєвих сповіщень про зміни в стані рослин;

– інтуїтивно зрозумілий інтерфейс, спрощений, але функціональний інтерфейс для швидкого доступу до основних функцій системи;

– офлайн режим, можливість роботи в офлайн режимі з подальшим синхронізуванням даних при відновленні інтернет-з'єднання.

Візуалізація даних:

– графіки та діаграми, дані повинні відображатися у вигляді різних типів графіків (лінійні, стовпчикові, кругові, гістограми тощо) для наочного аналізу трендів та змін;

– інтерактивність, графіки повинні бути інтерактивними, з можливістю масштабування, фільтрації та деталізації інформації;

– анімовані візуалізації, використання анімації для покращення розуміння змін у даних за певний період;

– таблиці, дані повинні бути доступні у вигляді таблиць для детального аналізу та порівняння;

– сортування та фільтрація, таблиці повинні мати функції сортування та фільтрації даних за різними критеріями;

– експорт даних, можливість експорту даних з таблиць у різні формати (CSV, Excel, PDF) для подальшого аналізу;

– інші візуальні елементи, використання різних візуальних елементів, таких як індикатори, теплові карти, діаграми Венна, для представлення комплексної інформації в зручному форматі.

Аналітика та звіти:

– інструменти для аналізу даних, система повинна надавати користувачам потужні інструменти для аналізу даних, включаючи:

– фільтрація даних, можливість фільтрування даних за різними параметрами для отримання специфічної інформації;

– порівняльний аналіз, інструменти для порівняння різних наборів даних для виявлення тенденцій та закономірностей;

– трендовий аналіз, засоби для аналізу трендів у даних з метою прогнозування майбутніх змін та прийняття обґрунтованих рішень;

– генерація звітів, система повинна підтримувати автоматичну генерацію звітів за заданими параметрами:

– щоденні, щотижневі, щомісячні звіти, можливість налаштування періодичних звітів для моніторингу росту рослин і стану умов вирощування;

– користувацькі звіти, можливість створення користувацьких звітів за специфічними критеріями, встановленими користувачем;

– автоматичне розсилання, функція автоматичної розсилки звітів на електронну пошту користувачів або їх завантаження у хмарне сховище;

– рекомендації щодо оптимізації росту рослин, система повинна надавати користувачам рекомендації на основі аналізу зібраних даних:

– оптимізація умов вирощування, рекомендації щодо коригування рівня вологості, температури, освітленості тощо для оптимізації росту рослин;

– прогнозування ризиків, прогнози щодо можливих ризиків, таких як посуха, шкідники, захворювання рослин, з відповідними рекомендаціями для їх запобігання;

– планування робіт, рекомендації щодо оптимального часу для проведення різних агротехнічних заходів (полив, підживлення, збір урожаю тощо) на основі аналізу фенологічних даних.

Реалізація цих функцій дозволить користувачам ефективно використовувати зібрані дані для прийняття обґрунтованих рішень, спрямованих на підвищення врожайності та якості рослин [40].

2.2.4 Налаштування параметрів та управління

Користувачі повинні мати можливість налаштовувати параметри системи та керувати нею для забезпечення оптимальної роботи та відповідності специфічним потребам вирощування рослин. Цей розділ охоплює функції налаштування порогових значень, управління сповіщеннями та доступ до налаштувань через різні інтерфейси [41].

Налаштування порогових значень:

– становлення порогових значень: користувачі повинні мати можливість встановлювати порогові значення для різних параметрів;

– вологість ґрунту: встановлення мінімальних та максимальних значень вологості ґрунту для запобігання пересиханню або перенасиченню водою;

– температура: налаштування допустимого діапазону температур для повітря та ґрунту, щоб забезпечити оптимальні умови для росту рослин;

– освітленість: визначення необхідного рівня освітленості для різних фаз росту рослин, щоб гарантувати достатнє освітлення;

– рівень CO2: встановлення порогових значень для концентрації вуглекислого газу в повітрі, щоб забезпечити ефективний фотосинтез [32];

– гнучкість налаштувань: користувачі повинні мати можливість задавати різні порогові значення для різних видів рослин або ділянок вирощування, враховуючи специфічні потреби кожного виду;

– автоматичне коригування: система повинна пропонувати функцію автоматичного коригування порогових значень на основі історичних даних та аналізу умов вирощування.

Управління сповіщеннями:

– налаштування сповіщень: користувачі повинні мати можливість налаштовувати сповіщення для різних параметрів;

– типи сповіщень: вибір типу сповіщень (SMS, електронна пошта, повідомлення у мобільному додатку) залежно від переваг користувача;

– частота сповіщень: налаштування частоти сповіщень для уникнення перевантаження інформацією (наприклад, негайні сповіщення для критичних значень або зведення раз на день для менш важливих параметрів);

– переважні умови: визначення умов, за яких будуть відправлятися сповіщення (наприклад, перевищення температури на 5°C вище заданого порогу або зниження рівня вологості на 10%);

– підтвердження отримання: система повинна мати функцію підтвердження отримання сповіщень користувачами для забезпечення своєчасної реакції на критичні ситуації;

– історія сповіщень: зберігання історії сповіщень з можливістю їх перегляду та аналізу для виявлення частих проблем та оптимізації налаштувань.

Доступ до налаштувань:

– веб-інтерфейс: користувачі повинні мати можливість дистанційно налаштовувати систему через зручний веб-інтерфейс;

– легкість використання: інтуїтивно зрозумілий інтерфейс для налаштування параметрів системи;

– безпека: захист налаштувань за допомогою паролів та двофакторної аутентифікації для запобігання несанкціонованому доступу;

– доступність: підтримка всіх основних браузерів для забезпечення доступу з будь-якого пристрою;

– мобільний додаток: користувачі повинні мати можливість налаштовувати систему через мобільний додаток з такими функціями;

– реальний час: можливість негайного внесення змін до налаштувань в реальному часі.

– офлайн режим: здатність робити налаштування в офлайн режимі з подальшим синхронізуванням при відновленні інтернет-з'єднання;

– пуш-сповіщення: отримання миттєвих повідомлень про критичні зміни параметрів та можливість їх оперативного коригування;

– синхронізація налаштувань: автоматична синхронізація налаштувань між веб-інтерфейсом та мобільним додатком для забезпечення узгодженості і актуальності даних;

– рольовий доступ: надання можливості управління налаштуваннями різним категоріям користувачів з різними рівнями доступу;

– адміністратори: повний доступ до всіх налаштувань системи;

– оператори: обмежений доступ до налаштувань, необхідних для виконання щоденних завдань;

– спостерігачі: доступ тільки для перегляду налаштувань та отримання сповіщень без можливості внесення змін.

Впровадження цих функцій забезпечить користувачам можливість ефективного налаштування та управління системою, дозволяючи максимально адаптувати її до індивідуальних потреб і умов вирощування рослин [41].

2.2.5 Сповіщення та алерти

Система повинна забезпечувати надсилання сповіщень та алертів користувачам у разі критичних змін параметрів [29]:

– Email та SMS сповіщення: надсилання сповіщень через електронну пошту та SMS;

– Push-сповіщення: надсилання push-сповіщень через мобільний додаток.

Система сповіщень та алертів повинна забезпечувати надійне і оперативне сповіщення користувачів про критичні зміни в параметрах. Це охоплює надсилання сповіщень через різноманітні канали зв'язку, такі як електронна пошта, SMS та push-сповіщення через мобільні додатки. Важливо, щоб ці сповіщення були точними і інформативними, містивши деталі про те, який параметр перевищив порогове значення, час цього відбуття та рекомендації щодо подальших дій [30].

Користувачам слід мати можливість персоналізувати свої налаштування сповіщень, встановлюючи пріоритети та частоту отримання. Це дозволяє кожному користувачеві налаштувати сповіщення відповідно до їхніх потреб і вимог. Крім того, система повинна підтримувати архів сповіщень для подальшого аналізу та збереження історичних даних. Це допомагає виявляти тренди та шаблони змін, що можуть мати важливе значення для користувачів [42].

Динамічні порогові значення можуть коригуватися автоматично на основі історичних даних та аналізу трендів. Це дозволяє системі адаптуватися до змінних умов та забезпечувати більш точне та ефективне управління процесами. Крім того, система повинна надавати інформацію про стан самої системи, включаючи збої сенсорів або інші проблеми, які можуть впливати на її функціонування. Ця

інформація дозволяє оперативно реагувати на проблеми та забезпечує надійну роботу системи в цілому [43].

2.3 Не функційні вимоги

Не функційні вимоги описують характеристики, що визначають продуктивність, надійність, масштабованість, безпеку та інші аспекти системи, які не стосуються конкретних функцій [44].

2.3.1 Вимоги до продукту

1) продуктивність:

– система повинна забезпечувати обробку та передачу даних від сенсорів у реальному часі з мінімальною затримкою;

– час відгуку на запити користувача не повинен перевищувати 2 секунд для основних операцій, таких як перегляд даних і генерація звітів;

2) надійність:

– система повинна бути високонадійною та стійкою до збоїв. Коефіцієнт безвідмовної роботи повинен становити не менше 99,9%;

– дані повинні зберігатися з використанням механізмів резервного копіювання для запобігання втратам у разі апаратних або програмних збоїв.

Перша ключова вимога стосується продуктивності. Це означає, що система повинна бути здатною ефективно обробляти та передавати дані в реальному часі, мінімізуючи будь-які затримки. Крім того, користувачі очікують, щоб система реагувала на їхні запити швидко та ефективно, не перевищуючи час відповіді 2 секунди для основних операцій, таких як перегляд даних та генерація звітів [45].

Друга важлива вимога - надійність. Користувачі мають право очікувати, що система буде працювати беззбойно та надійно. Коефіцієнт безвідмовної роботи повинен бути на рівні не менше 99,9%. Це означає, що система повинна бути стійкою до будь-яких збоїв та відновлюватися після них без втрати даних або

перерви в роботі. Для забезпечення цієї надійності, необхідно зберігати дані за допомогою механізмів резервного копіювання, щоб уникнути можливих втрат у разі апаратних або програмних збоїв.

2.3.2 Організаційні вимоги

1) вимоги до документації:

– система повинна супроводжуватися повною технічною документацією, включаючи керівництво користувача, керівництво з встановлення та налаштування, а також документацію з розробки;

– документація повинна бути доступною українською та англійською мовами;

2) підтримка та обслуговування:

– повинна бути організована служба технічної підтримки користувачів, що надає консультації через телефон, електронну пошту та чат;

– система повинна отримувати регулярні оновлення програмного забезпечення для підвищення безпеки та продуктивності.

Це не лише стосується практичних аспектів використання продукту, але й має важливе наукове значення в контексті організації робочих процесів і підтримки користувачів.

По-перше, наявність повної технічної документації є ключовим елементом для ефективного використання продукту. Дослідження показують, що доступність і зрозумілість документації значно полегшує процес впровадження системи та забезпечує ефективність роботи користувачів.

По-друге, належна підтримка користувачів є важливим аспектом для забезпечення стабільності та продуктивності системи. Дослідження в галузі обслуговування користувачів підкреслюють важливість швидкого та ефективного реагування на запити користувачів через різноманітні канали зв'язку, такі як телефон, електронна пошта та чат.

Крім того, регулярне оновлення програмного забезпечення відіграє важливу роль у забезпеченні безпеки та функціональності системи. Нові версії програмного забезпечення часто включають в себе виправлення помилок, поліпшення функціональності та заходи забезпечення безпеки, що допомагають забезпечити стабільну та ефективну роботу продукту.

2.3.3 Вимоги щодо взаємодії з зовнішнім середовищем.

1) інтероперабельність:

– система повинна бути сумісною з різними типами сенсорів і мікроконтролерів, що відповідають стандартам промислової автоматизації;

– повинна бути забезпечена можливість інтеграції з іншими системами обробки та аналізу даних через API;

2) екологічні умови:

– система повинна функціювати в різних кліматичних умовах, включаючи екстремальні температури, високу вологість та вплив пилу [31];

– обладнання повинно відповідати стандартам захисту IP65 або вищим.

Це визначає ключові аспекти, необхідні для забезпечення оптимальної роботи системи в різних умовах та для її взаємодії з зовнішнім середовищем. Особлива увага приділяється інтероперабельності та стійкості до екстремальних умов експлуатації [46].

Інтероперабельність системи є важливим аспектом, оскільки вона дозволяє системі ефективно співпрацювати з іншими пристроями та програмами. Дослідження в області інтероперабельності підкреслюють значення стандартизації інтерфейсів для забезпечення сумісності різних компонентів системи. Зокрема, важливо, щоб система була сумісною з різними типами сенсорів та мікроконтролерів, що використовуються в промисловій автоматизації, та мала можливість інтеграції з іншими системами обробки та аналізу даних через API.

Щодо стійкості до екстремальних умов експлуатації, важливо врахувати різноманітні фактори, такі як температурні коливання, вологість, пил та інші. Дослідження показують, що екстремальні умови можуть негативно впливати на роботу обладнання та системи в цілому. Тому система повинна бути розроблена з урахуванням цих факторів, а обладнання має відповідати високим стандартам захисту, наприклад, стандарту IP65 або вищим, що гарантує його стійкість до вологості та пилу.

Отже, врахування інтероперабельності та стійкості до екстремальних умов є критичними аспектами в розробці системи, що сприяють її ефективності та надійності у різних умовах експлуатації.

2.3.4 Вимоги до безпеки та конфіденційності даних.

1) безпека даних:

– дані повинні передаватися з використанням шифрування (SSL/TLS) для захисту від несанкційованого доступу. Сертифікат SSL/TLS — це цифровий об'єкт, який дозволяє системам перевіряти особистість та згодом встановлювати зашифроване мережеве з'єднання з іншою системою за допомогою протоколу Secure Sockets Layer/Transport Layer Security (SSL/TLS). Сертифікати використовуються в рамках криптографічної системи відомою як інфраструктура відкритого ключа (PKI). PKI дає одній стороні можливість встановлювати справжність іншої сторони за допомогою сертифікатів (за умови, що обидві сторони довіряють третій стороні, відомій як центр сертифікації). Таким чином, сертифікати SSL/TLS діють як цифрові посвідчення особи для захисту мережевих підключень та встановлення автентичності веб-сайтів в Інтернеті, а також ресурсів у приватних мережах[18];

– доступ до системи повинен бути захищений багатофакторною автентифікацією (MFA).Багатофакторна автентифікація (MFA) – це процес входу в систему, який складається з декількох кроків і вимагає від користувача вказати

більше інформації, а не тільки пароль. Наприклад, крім пароля, система може попросити користувача вказати код, надісланий на електронну пошту, відповісти на секретне запитання або сканувати відбитки пальців. Друга форма автентифікації може допомогти запобігти несанкціонованому доступу до облікового запису, якщо системний пароль було зламано. Цифрова безпека має вирішальне значення в світі, оскільки і підприємства, і користувачі зберігають конфіденційну інформацію в Інтернеті. Кожна людина взаємодіє з програмами, сервісами та даними, які зберігаються в Інтернеті, використовуючи облікові записи онлайн. Порушення або неправомірне використання цієї інформації в Інтернеті може мати серйозні реальні наслідки, такі як крадіжка фінансових коштів, порушення ділової активності та втрата конфіденційності. Хоча паролі захищають цифрові активи, їх просто недостатньо. Експерти у сфері кіберзлочинності намагаються активно підбирати паролі. Дізнавшись один пароль, можна отримати доступ до кількох облікових записів, для яких ви можете використовувати пароль повторно. Багатофакторна автентифікація діє як додатковий рівень безпеки, запобігаючи доступу неавторизованих користувачів до цих облікових записів, навіть якщо пароль був викрадений[19];

2) конфіденційність:

– повинні бути дотримані стандарти конфіденційності даних, такі як GDPR, для захисту особистої інформації користувачів;

– дані повинні бути анонімізовані, якщо вони використовуються для наукових досліджень або аналітики без згоди користувача.

2.3.5 Вимоги до масштабованості

1) горизонтальне та вертикальне масштабування:

– система повинна підтримувати горизонтальне масштабування (додавання нових сенсорів і мікроконтролерів) та вертикальне масштабування (підвищення продуктивності існуючих компонентів);

– інфраструктура AWS повинна забезпечувати автоматичне масштабування ресурсів залежно від навантаження.

Вимоги до масштабованості є одними з ключових аспектів, які визначають гнучкість та ефективність системи в умовах змінного обсягу даних та навантаження.

Почнемо з горизонтального та вертикального масштабування. Горизонтальне масштабування полягає в додаванні нових сенсорів та мікроконтролерів до системи, щоб розширити її функціональні можливості та забезпечити обробку додаткового обсягу даних. Це дозволяє системі легко адаптуватися до зростаючого навантаження та забезпечити продовження її роботи без значного переривання. З іншого боку, вертикальне масштабування передбачає підвищення продуктивності існуючих компонентів системи, таким чином, забезпечуючи оптимізацію роботи та зменшення затримок у відповіді. Обидва підходи до масштабування є важливими для забезпечення ефективної роботи системи в умовах змін [47].

Ще одним важливим аспектом є автоматичне масштабування ресурсів, яке забезпечується інфраструктурою AWS (Amazon Web Services). Це дозволяє системі динамічно адаптуватися до змін у навантаженні, автоматично масштабуючи ресурси, такі як обчислювальні потужності та сховища даних, для забезпечення ефективного використання ресурсів та оптимізації продуктивності. Автоматичне масштабування дозволяє системі зберігати стабільну продуктивність навіть при зміні обсягу даних та навантаження, що робить її більш гнучкою та масштабованою.

2.4 Вимоги предметної галузі

Вимоги предметної галузі визначають специфічні характеристики та обмеження, які пов'язані з аграрною сферою та інтеграцією з AWS для системи моніторингу росту рослин.

Вимоги предметної галузі є важливою частиною проекту, оскільки вони визначають унікальні характеристики та специфічні обмеження, що виникають в контексті аграрної сфери та інтеграції з платформою Amazon Web Services (AWS) для системи моніторингу росту рослин.

Однією з головних вимог є забезпечення сумісності інтегрованої системи зі специфічними потребами сільськогосподарського сектору. Це означає, що система повинна бути готовою до використання у різних галузях сільського господарства, таких як рослинництво, тваринництво, а також виноградарство та садівництво. Кожна з цих галузей має свої унікальні особливості та вимоги до моніторингу, які потрібно врахувати при розробці системи[48].

До інших вимог предметної галузі відносяться обмеження, пов'язані з використанням сільськогосподарської техніки та обладнання. Система повинна бути сумісною з різними типами сільськогосподарських датчиків, мікроконтролерів та інших пристроїв, які використовуються у виробництві сільськогосподарської продукції. Крім того, важливо враховувати особливості роботи в умовах зовнішнього середовища, таких як високі температури, вологість та вплив шкідливих факторів.

Інтеграція з платформою AWS також вносить свої особливості у вимоги до предметної галузі. Система повинна бути готовою до використання у хмарному середовищі, забезпечуючи безпеку, масштабованість та надійність в ході взаємодії з AWS. Крім того, важливо враховувати специфічні функції та можливості, які пропонує AWS для розвитку та оптимізації системи моніторингу росту рослин.

Загальна мета вимог предметної галузі полягає в забезпеченні ефективної та надійної роботи системи моніторингу росту рослин в умовах сільськогосподарського виробництва, з урахуванням всіх специфічних потреб та обмежень даного сектору.

2.4.1 Вимоги до сенсорів та мікроконтролерів

Система повинна використовувати сучасні сенсори та мікроконтролери з можливістю бездротового зв'язку, які забезпечать точний та надійний збір даних про ріст рослин. Вимоги до сенсорів включають високу точність вимірювань вологості ґрунту, освітленості та температури навколишнього середовища. Мікроконтролери повинні мати достатні ресурси для обробки даних та передачі їх на платформу AWS.

Вимоги до сенсорів та мікроконтролерів є критичними для успішної реалізації системи моніторингу росту рослин. Враховуючи постійний розвиток технологій у галузі агротехнологій, необхідно використовувати сучасні сенсори та мікроконтролери з високою точністю та надійністю.

Сучасні сенсори для моніторингу росту рослин мають ряд переваг, серед яких важливість бездротового зв'язку. Це дозволяє зручно розміщувати сенсори на великих площах поля або у складних умовах теплиць. Також важливо, щоб сенсори були точними та надійними вимірювачами, здатними відслідковувати вологість ґрунту, освітленість та температуру навколишнього середовища з високою точністю.

Мікроконтролери, що використовуються в системі, також мають бути високоякісними та потужними, здатними забезпечувати обробку даних та передачу їх на платформу AWS без затримок та втрат інформації. Вони повинні мати достатній обсяг оперативної пам'яті та обчислювальні ресурси для ефективної роботи в умовах постійного збільшення обсягу зібраних даних.

При виборі сенсорів та мікроконтролерів важливо враховувати сумісність з іншими компонентами системи та їх готовність до інтеграції з платформою AWS. Також слід враховувати вартість та доступність запасних частин, щоб забезпечити неперервну роботу системи у разі потреби заміни деяких компонентів.

2.4.2 Вимоги до інтеграції з AWS

Інтеграція з платформою AWS має бути ефективною та безперервною. Система повинна забезпечувати надійну передачу даних на AWS IoT Core для подальшої обробки та зберігання. Вимоги до інтеграції включають використання протоколів безпеки та шифрування для захисту конфіденційності даних під час передачі. Крім того, система повинна мати можливість автоматичного масштабування ресурсів на AWS для забезпечення оптимальної продуктивності та надійності.

2.5 Верифікація вимог

Верифікація вимог є ключовим етапом у процесі розробки програмно-технічного засобу, оскільки вона спрямована на перевірку відповідності розробленої системи встановленим вимогам. Під час верифікації вимог застосовуються різноманітні методи, включаючи тестування, моделювання та аналіз. Тестування може включати симуляцію реальних умов експлуатації системи або виконання тестових сценаріїв для перевірки правильності її реакції на різні вхідні дані. Моделювання дозволяє передбачити поведінку системи в різних сценаріях та перевірити її ефективність. Аналіз вимог дозволяє зрозуміти, чи відповідають вони реальним потребам та чи задовольняють вони очікування замовника. Після завершення верифікації вимоги повинні бути перевірені та підтверджені замовником перед продовженням процесу розробки.

Верифікація вимог є невід'ємною частиною процесу розробки програмно-технічного засобу, спрямованого на забезпечення відповідності розробленої системи встановленим вимогам. Цей етап включає в себе ретельний аналіз, перевірку та тестування кожного аспекту системи з метою визначення його відповідності вимогам та специфікаціям.

Одним з методів верифікації вимог є тестування. Це включає проведення різноманітних тестів, таких як модульне тестування, функціональне тестування,

інтеграційне тестування та системне тестування, для перевірки правильності реалізації функціональності системи та її відповідності вимогам.

Моделювання є іншим ефективним методом верифікації. Воно дозволяє імітувати різні умови роботи системи та перевірити її поведінку в різних сценаріях. Це дозволяє виявити можливі проблеми та неочікувані наслідки в ранніх етапах розробки.

Крім того, аналіз вимог грає ключову роль у верифікації. Це включає в себе ретельне перевіряння вимог на предмет їхньої відповідності реальним потребам та очікуванням замовника. Важливо переконатися, що вимоги чіткі, зрозумілі та повні, а також що вони взаємно сумісні та не суперечать одна одній.

Після завершення верифікації, вимоги повинні бути перевірені та підтверджені замовником. Це важливий етап, що гарантує, що розроблена система відповідає потребам та очікуванням замовника перед тим, як продовжувати процес розробки.

2.6 Висновки

У межах розділу 2 дипломної роботи зосереджено на проектуванні програмно-технічного засобу для системи моніторингу росту рослин з використанням AWS. Зокрема, були розглянуті та деталізовані користувацькі та функційні вимоги, визначені не функційні вимоги, а також розглянуті вимоги предметної галузі. Після цього було надано загальний огляд процесу верифікації вимог, який є необхідним кроком у забезпеченні якості та надійності розробленого програмного продукту. Завершує цей розділ висновок, у якому підкреслюється важливість дотримання встановлених вимог для успішного виконання проекту та подальшого розвитку програмно-технічного засобу.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ МІКРОКОНТРОЛЕРНОЇ СИСТЕМИ МОНІТОРИНГУ РОСТУ РОСЛИН З ВИКОРИСТАННЯМ AWS

3.1 Опис реалізації модулів апаратного та програмного забезпечення мікроконтролерної системи

Апаратне забезпечення включає в себе мікроконтролери, що відповідають за збір даних від різних сенсорів, таких як сенсори вологості ґрунту, температури, освітленості тощо [26]. Ці дані збираються і передаються на сервер AWS для подальшої обробки та аналізу [27].

Програмне забезпечення розроблене для ефективної обробки даних та забезпечення взаємодії з апаратним забезпеченням. Воно включає в себе алгоритми обробки даних, модулі зв'язку з сервером AWS та інтерфейс для взаємодії з користувачем. Це програмне забезпечення дозволяє здійснювати віддалений моніторинг росту рослин, формування польотних завдань та коригування їх у реальному часі за допомогою інтерфейсу користувача [28].

Для реалізації архітектури системи потрібні лише Raspberry Pi 4, веб-камера, обліковий запис AWS та основний комп'ютер для програмування. Установка цих елементів на рисунку 3.1:

1) Raspberry Pi 4: Це одноплатний комп'ютер, який використовується як основа для нашої системи. Завдяки своєму потужному процесору він здатний обробляти дані від підключених пристроїв і взаємодіяти з обліковим записом AWS;

2) вебкамера: вона використовується для отримання зображень рослин, які подальше аналізує Raspberry Pi. Пристрій підключається до Raspberry Pi за допомогою USB і передає зображення для подальшої обробки;

3) обліковий запис AWS: Це хмарна платформа, де ми зберігаємо дані та виконуємо їхню обробку. Наша система взаємодіє з AWS для передачі зібраних даних, їх збереження та аналізу;

4) основний комп'ютер для програмування: Це комп'ютер, на якому ми розробляємо програмне забезпечення для нашої системи. Цей комп'ютер використовується для написання коду, тестування та вдосконалення системи перед її впровадженням на Raspberry Pi.

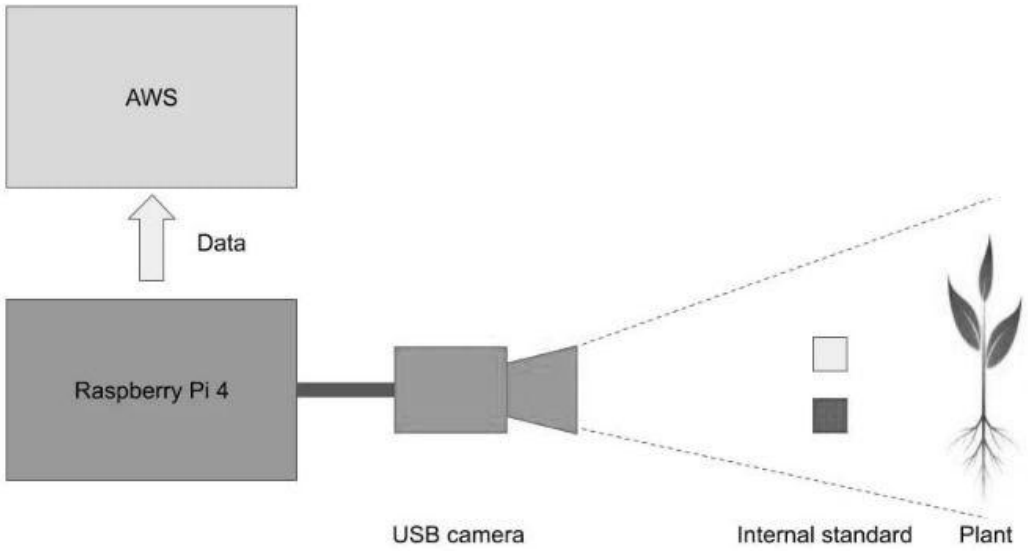


Рисунок 3.1 – Конфігурація та підключення елементів для створення мікроконтролерної системи моніторингу росту рослин з використанням AWS.

Для вимірювання площі у квадратних сантиметрах потрібно мати два квадратні шматочки паперу. Один із них білий, а інший – червоний. Кожен з них має сторону одного квадратного сантиметра. Ці шматочки розташовуються у нижній частині зображення, з метою визначення наближеного відношення кількості пікселів до одного квадратного сантиметра. У наукових дослідженнях такі об'єкти відомі як внутрішні еталони [49].

3.2 Опис процесу створення та налаштування конфігурації AWS

Перш ніж перейти до подальших етапів, необхідно належним чином налаштувати сервіс Amazon Web Services (AWS) для нашого проекту. У цьому контексті я в основному дотримувався інструкцій, наведених у офіційному посібнику.

Спочатку потрібно створити таблицю DynamoDB з ім'ям "pi_green" у консолі AWS. Для цього визначаємо поле "sample_time" як основний ключ розподілу, яке відповідає за розподіл даних між різними розділами таблиці, та поле "device_id" як додатковий ключ сортування, яке дозволить впорядкувати дані всередині кожного розділу. Важливо зауважити, що обидва ці ключі мають числовий формат, що визначає їхню типізовану природу та дозволяє ефективно працювати з даними.

Далі переходимо до керування Інтернетом речей (IoT) у консолі AWS. Натискаємо кнопку "Connect one device" і зберігаємо отриману точку доступу (endpoint). На наступному етапі необхідно надати пристрою певну назву, що відповідає його призначенню та ідентифікує його у системі. Підключення Raspberry Pi до AWS IoT наведено нижче на рисунку 3.2.

Під час налаштування Amazon Web Services (AWS) важливо вибрати мову програмування Python в розділі SDK для пристроїв AWS IoT (рисунок 3.3.). Це забезпечить можливість використання Python для розробки програмного забезпечення, що взаємодіє з платформою IoT на AWS.

Для вибору Python необхідно здійснити відповідні налаштування у консолі AWS. Після цього ми матимемо можливість розробляти програми для взаємодії з AWS IoT, використовуючи потужні можливості мови програмування Python[50].

У наступному кроці необхідно завантажити набір з'єднання на Raspberry Pi (рисунок 3.4.). Це набір містить програмне забезпечення, яке дозволить пристрою підключитися до AWS IoT і взаємодіяти з ним. Після завантаження програмного забезпечення на Raspberry Pi слід виконати вказані інструкції для його запуску.

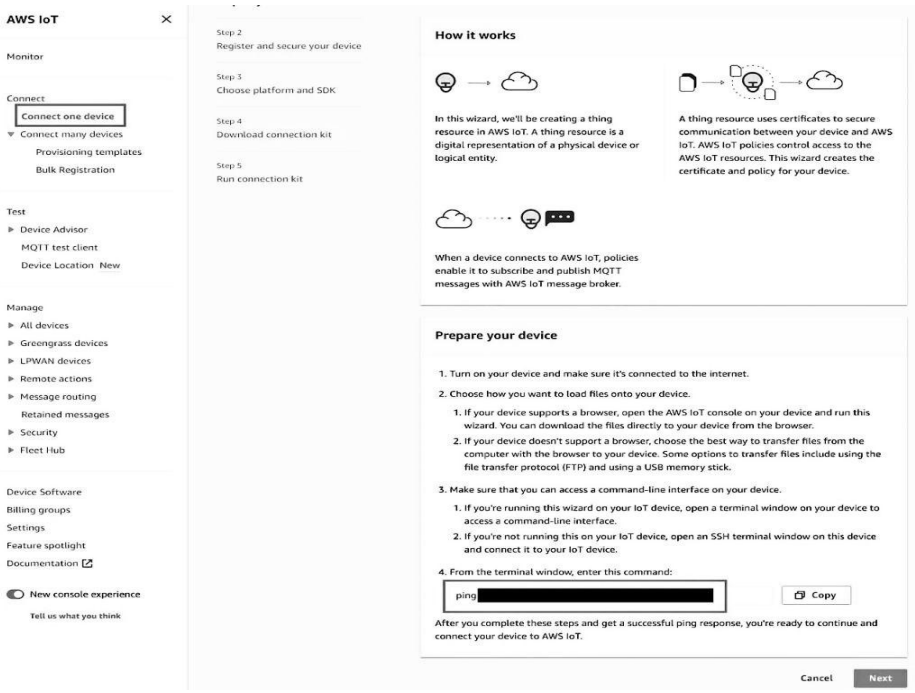


Рисунок. 3.2 – Підключення Raspberry Pi до AWS IoT

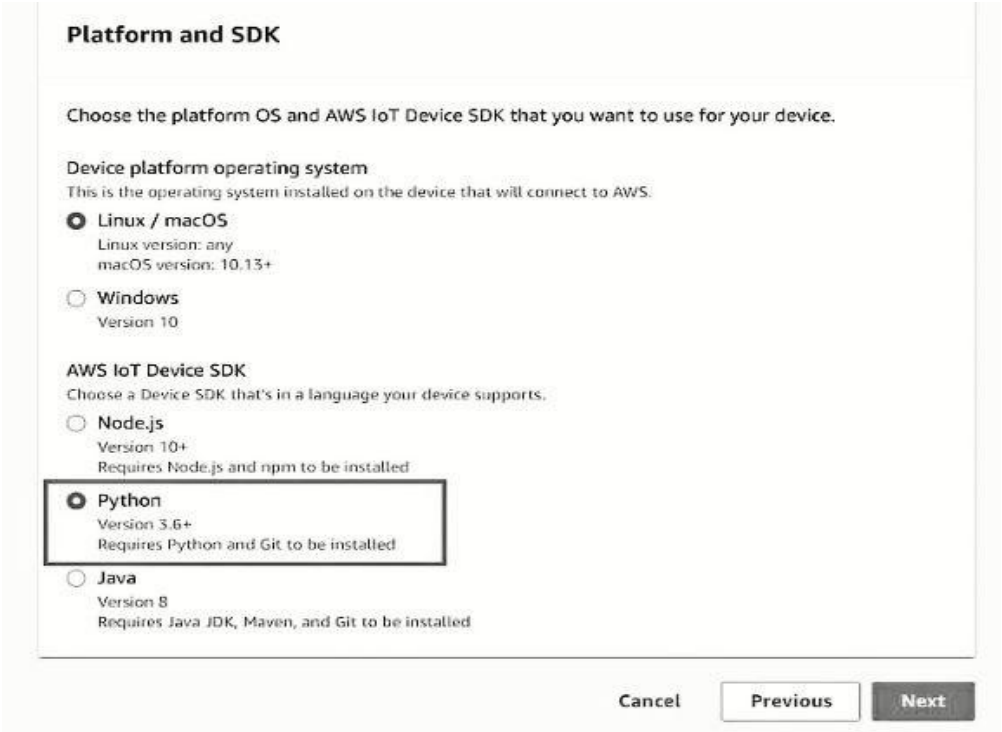


Рисунок. 3.3 – Вибір правильного SDK

Після успішного запуску ви побачите потік повідомлень на екрані підписки. Це означатиме, що Raspberry Pi успішно підключений до AWS IoT і готовий взаємодіяти з платформою.

Далі, щоб забезпечити можливість надсилання повідомлень на інші теми, необхідно змінити політику для пристрою. Політика визначає, які дії може виконувати пристрій в межах AWS IoT. Розслаблення політики дозволить Raspberry Pi відправляти повідомлення на інші теми, що є необхідним для взаємодії з іншими компонентами системи. Для цього слід перейти до розділу налаштувань політики в консолі AWS та внести необхідні зміни, дозволяючи Raspberry Pi відправляти повідомлення на потрібні теми.

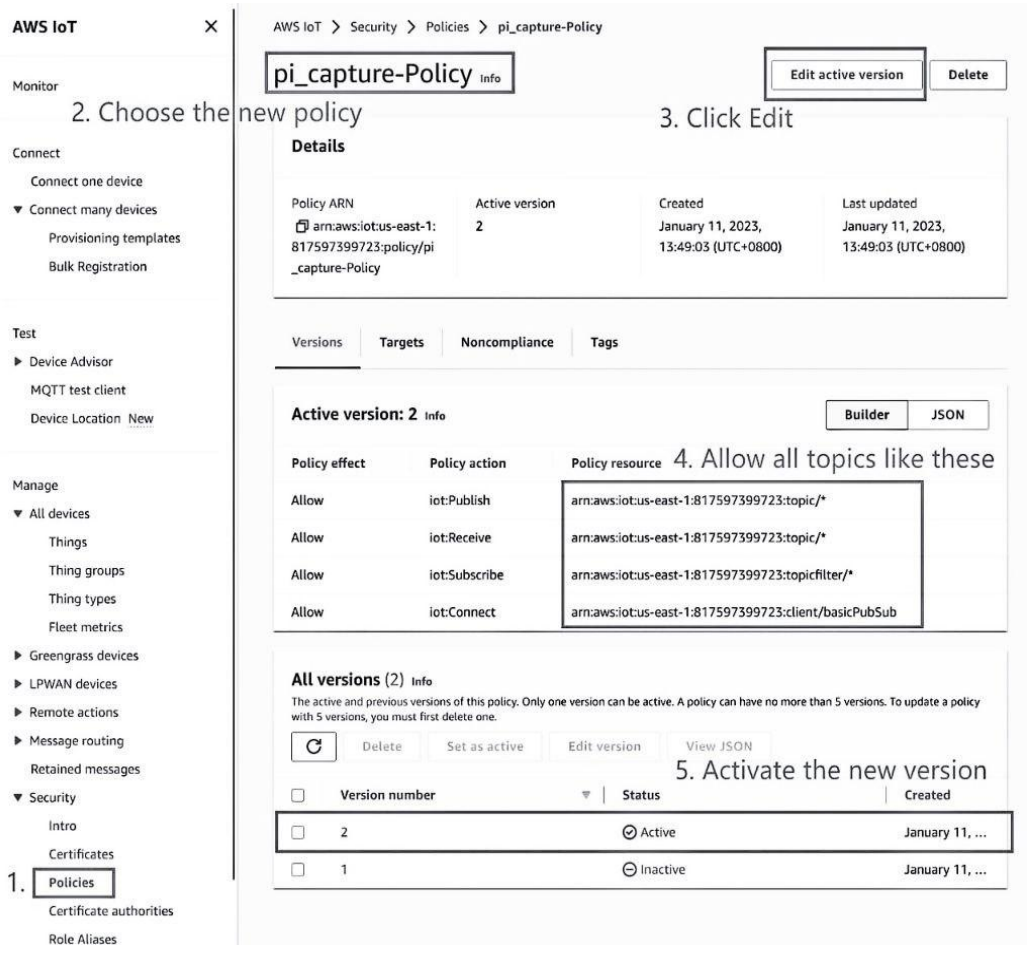


Рисунок. 3.4 – Зміна політики, щоб дозволити всі теми

3.3 Скрипт головного модуля Python

Однією з найбільш складних частин є головний скрипт, написаний мовою програмування Python. Першим етапом роботи цього скрипта є виконання фотозйомки за допомогою камери. Для цього скрипт ініціює підключення до камери, активує її та виконує зйомку. Цей крок є критичним, оскільки якість та чіткість отриманого зображення визначатимуть подальшу ефективність системи моніторингу росту рослин.

```
camera = cv2.VideoCapture(0)
ret, img = camera.read()
camera.release()
```

Після отримання зображення з камери скрипт переходить до аналізу його пікселів. Особливу увагу приділяється визначенню зелених, червоних та білих областей на зображенні. Цей аналіз виконується у просторі кольорів HSV (відтінок, насиченість, значення), що дозволяє більш точно визначити кольорові відтінки.

Для виявлення кожного з цих кольорів встановлюються певні пороги у просторі HSV. Ці пороги налаштовуються на основі попередньо визначених значень, які враховують досвід та експертні знання. Наприклад, для зеленого кольору можуть встановлюватися межі відтінків та насиченості, що відповідають зеленому відтінку рослинного листя.

Для визначення цих порогів кольору використовується Python-скрипт, розроблений користувачем nathancy на платформі stackoverflow.com. Цей скрипт надає зручний і ефективний спосіб налаштування порогів кольорів, що дозволяє точно визначити кольорові області на зображенні(рисунок 3.4).

```
def get_bitwise (img, lower, higher, lower_bound, upper_bound, left_bound,
right_bound):
## convert to hsv
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

mask = cv2.inRange(hsv, lower, higher)
```

```
result = cv2.bitwise_and(img, img, mask = mask)

## limit the whereabouts of the standards
for h in range(result.shape[0]):
if h > lower_bound and h < upper_bound:
for w in range(result.shape[1]):
if w > left_bound and w < right_bound:
pass
else:
result[h, w, :] = 0
else:
result[h, :, :] = 0

return result

## slice the green
green = get_bitwise(img, (int(colorinfo["GREEN_HMin"]),
int(colorinfo["GREEN_SMin"]), int(colorinfo["GREEN_VMin"])),
(int(colorinfo["GREEN_HMax"]), int(colorinfo["GREEN_SMax"]),
int(colorinfo["GREEN_VMax"])), 0, img.shape[0], 0, img.shape[1])
cv2.imwrite(f'processed_img/{current_time}_green.png", green)
green_mask = green > 0
green_pixels = np.count_nonzero(green_mask)


## do the same for white and red standards,
## but we limit their positions to avoid noisy pixels
white = get_bitwise(img, (int(colorinfo["WHITE_HMin"]),
int(colorinfo["WHITE_SMin"]), int(colorinfo["WHITE_VMin"])),
(int(colorinfo["WHITE_HMax"]), int(colorinfo["WHITE_SMax"]),
int(colorinfo["WHITE_VMax"])), int(colorinfo["WHITE_LOWER_BOUND"]),
int(colorinfo["WHITE_UPPER_BOUND"]), int(colorinfo["WHITE_LEFT_BOUND"]),
int(colorinfo["WHITE_RIGHT_BOUND"]))
cv2.imwrite(f'processed_img/{current_time}_white.png", white)
white_mask = white > 0
white_pixels = np.count_nonzero(white_mask)


red = get_bitwise(img, (int(colorinfo["RED_HMin"]), int(colorinfo["RED_SMin"]),
int(colorinfo["RED_VMin"])), (int(colorinfo["RED_HMax"]),
int(colorinfo["RED_SMax"]), int(colorinfo["RED_VMax"])),
int(colorinfo["RED_LOWER_BOUND"]), int(colorinfo["RED_UPPER_BOUND"]),
int(colorinfo["RED_LEFT_BOUND"]), int(colorinfo["RED_RIGHT_BOUND"]))
```

```
cv2.imwrite(f"processed_img/{current_time}_red.png", red)
red_mask = red > 0
red_pixels = np.count_nonzero(red_mask)
```

Для забезпечення більш високої точності результатів було вирішено використати два внутрішні стандарти. Червоний і білий кольори обрано завдяки їх рідкісній присутності на ґрунті, що мінімізує ймовірність помилкових визначень. Проте навіть за таких умов деякі фонові пікселі могли впливати на розрахунки. Щоб усунути цей шум, було введено параметри, що задаються користувачем, для обмеження поля зору і маскування фону. Ці параметри дозволяють користувачу налаштовувати область аналізу зображення, щоб виключити небажані елементи, що можуть вплинути на результати. Візуалізація результатів цього підходу наведена на (рисунок 3.5.).

Після первинного оброблення зображення скрипт здійснює вилучення зелених, червоних та білих пікселів. Спочатку зображення конвертується у колірний простір HSV (відтінок, насиченість, яскравість), що дозволяє точніше ідентифікувати необхідні кольори. Далі встановлюються порогові значення для кожного кольору, які визначають, які пікселі будуть вважатися відповідними червоному, білому чи зеленому кольору. Для визначення цих порогів я використовував Python-скрипт, розроблений користувачем nathancy з платформи stackoverflow.com.

Після вилучення пікселів, скрипт обчислює середню кількість пікселів для внутрішніх стандартів – червоного і білого. Ці стандарти слугують референсними точками для подальших розрахунків. Виходячи з цих значень, скрипт визначає кількість зелених пікселів і переводить ці дані у площу зеленої зони в квадратних сантиметрах. Такий підхід забезпечує високу точність і надійність вимірювання площі рослинного покриття, що є критично важливим для моніторингу росту рослин та оцінки їх стану.

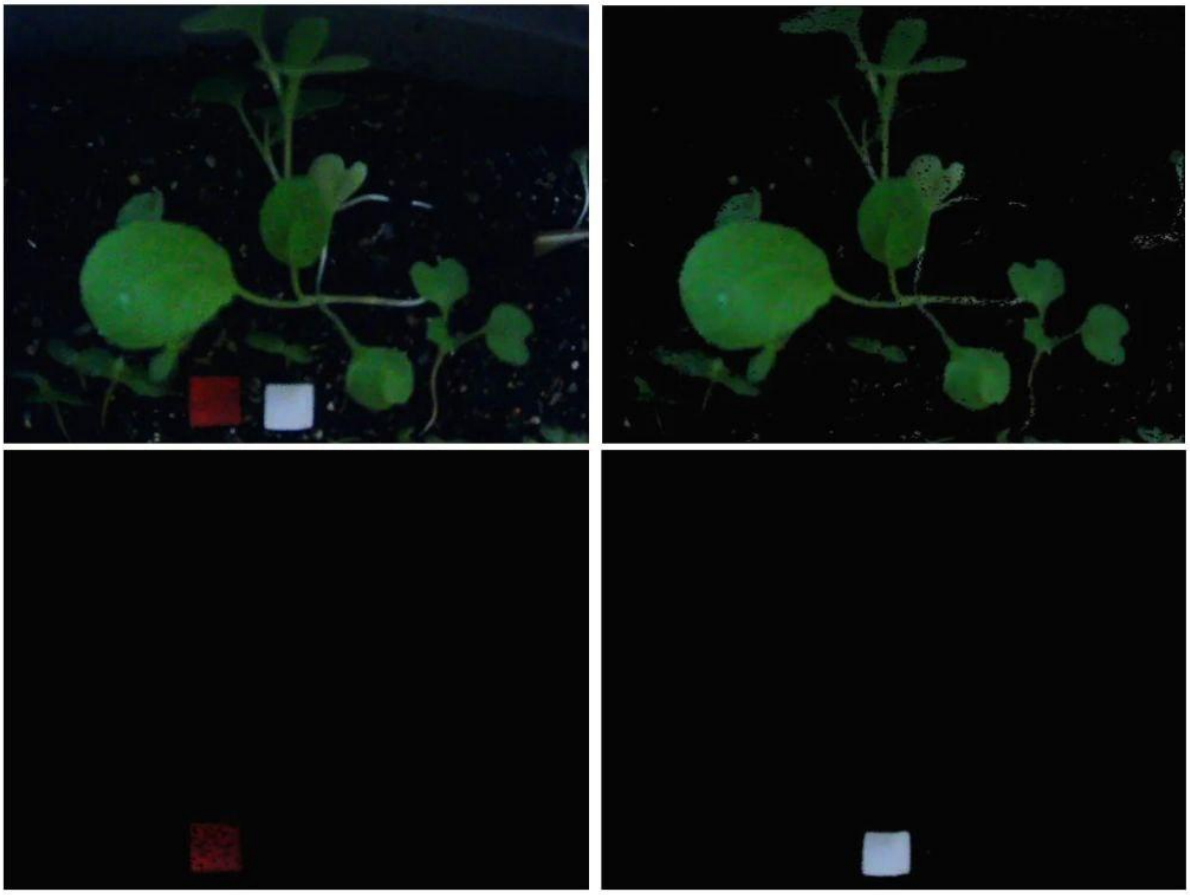


Рисунок 3.5 – Сценарій генерує фрагменти зелених пікселів, червоного та білого стандартів.

pixel_density_for_1_square_cm = (white_pixels + red_pixels)/2

green_area = green_pixels/pixel_density_for_1_square_cm

Після аналізу зображення та обчислення кількості зелених пікселів, наступним важливим кроком є передача результатів на платформу AWS за допомогою протоколу MQTT [25]. Цей етап є ключовим для забезпечення інтеграції з хмарною інфраструктурою та подальшої обробки даних [51].

Спочатку отримані дані упаковуються у формат JSON, створюючи змінну під назвою MESSAGE. Ця змінна містить усю необхідну інформацію, включаючи кількість зелених пікселів, що відповідає площі рослинного покриву. Формат

JSON забезпечує стандартизований і зручний для інтерпретації формат даних, який легко передавати та обробляти.

Наступним кроком є створення MQTT-клієнта. Цей клієнт відповідає за встановлення зв’язку з AWS IoT. Використовуючи необхідні файли підключення, які містять сертифікати та ключі безпеки, клієнт встановлює захищене з'єднання з сервером AWS IoT. Безпечне з'єднання є критичним для забезпечення конфіденційності та цілісності даних під час їх передачі.

Після успішного встановлення з'єднання, скрипт публікує повідомлення на визначеній темі MQTT. Тема, зазвичай, має формат device/DEVICE_ID/data, де DEVICE_ID - це унікальний ідентифікатор пристрою. Цей ідентифікатор дозволяє системі AWS IoT точно визначити, від якого пристрою надійшли дані, і відповідно їх обробити.

Передача даних на платформу AWS за допомогою протоколу MQTT забезпечує безперервний моніторинг і зберігання інформації про стан рослинного покриву. Це дозволяє в реальному часі отримувати актуальні дані, що є важливим для аналізу та прийняття рішень щодо росту та стану рослин.

```
TOPIC = f'device/{awsinfo["DEVICE_ID"]}/data'
MESSAGE = {"green_area": green_area}

myAWSIoTMQTTClient = AWSIoTPyMQTT.AWSIoTMQTTClient(CLIENT_ID)
myAWSIoTMQTTClient.configureEndpoint(ENDPOINT, 8883)
myAWSIoTMQTTClient.configureCredentials(PATH_TO_AMAZON_ROOT_CA_1,
PATH_TO_PRIVATE_KEY, PATH_TO_CERTIFICATE)

myAWSIoTMQTTClient.connect()
print('Begin Publish')


myAWSIoTMQTTClient.publish(TOPIC, json.dumps(MESSAGE), 1)
print("Published: '" + json.dumps(MESSAGE) + "' to the topic: " + f"{TOPIC}")

print('Publish End')
myAWSIoTMQTTClient.disconnect()
```

Для забезпечення максимальної гнучкості та зручності налаштувань нашої системи, ключові параметри, такі як DEVICE_ID, пороги кольорів, endpoint, client_id, а також шляхи до файлів підключення, можуть бути визначені у спеціальному конфігураційному файлі. Це рішення дозволяє легко адаптувати систему до різних умов та потреб, мінімізуючи необхідність внесення змін у основний код скрипта.

DEVICE_ID є унікальним ідентифікатором пристрою в системі, що дозволяє AWS IoT точно ідентифікувати джерело даних. Це особливо важливо в масштабованих системах, де використовується багато пристроїв, кожен з яких повинен бути однозначно розпізнаний.

Пороги кольорів визначають діапазони значень у просторі кольорів HSV, які відповідають різним кольорам, що цікавлять нас, зокрема зеленим, червоним та білим пікселям. Точне налаштування цих порогів дозволяє системі точно ідентифікувати відповідні кольорові області на зображенні, що критично важливо для аналізу стану рослин.

Endpoint – це адреса сервера AWS IoT, до якого підключається наш пристрій для передачі даних. Client_id використовується як ідентифікатор клієнта при підключенні до сервера, забезпечуючи правильну ідентифікацію та взаємодію з сервером.

Шляхи до файлів підключення включають місця зберігання сертифікатів та ключів безпеки, які необхідні для встановлення захищеного з'єднання з AWS IoT. Ці файли зберігаються в окремій папці під назвою aws_key, що дозволяє зручно організувати їх зберігання та легко знаходити необхідні файли під час налаштування з'єднання.

Використання конфігураційного файлу значно спрощує процес налаштування системи, дозволяючи швидко змінювати параметри без необхідності внесення змін до основного коду. Такий підхід забезпечує високу гнучкість та ефективність, що особливо важливо в умовах швидких змін та необхідності адаптації до нових вимог. Таким чином, ми можемо зосередитися на

аналізі даних та прийнятті обґрунтованих рішень щодо моніторингу росту рослин, не витрачаючи зайвий час на технічні налаштування системи.

3.4 Налаштування Raspberry Pi

Для налагодження Raspberry Pi у системі моніторингу росту рослин з використанням AWS потрібно виконати ряд важливих кроків.

Спочатку, слід встановити операційну систему на Raspberry Pi. Рекомендовано використовувати Raspberry Pi OS, спеціально оптимізовану для пристроїв цієї лінійки. Завантажте останню версію Raspberry Pi OS з офіційного веб-сайту та запишіть образ на SD-карту.

Після успішного завантаження Raspberry Pi OS виконайте початкове налаштування системи. Підключіть до Raspberry Pi монітор, клавіатуру, мишу та інші необхідні пристрої. Після цього підключіть Raspberry Pi до мережі через Ethernet або налаштуйте Wi-Fi-з'єднання. Проведіть оновлення операційної системи та встановлених пакетів.

Наступним кроком буде встановлення необхідного програмного забезпечення. Зокрема, слід встановити Python та необхідні бібліотеки за допомогою pip. Для взаємодії з AWS також встановіть AWS CLI.

Після встановлення програмного забезпечення налаштуйте підключення до AWS IoT. Завантажте сертифікати та ключі з AWS IoT Console та створіть конфігураційний файл з параметрами підключення.

Нарешті, запустіть основний скрипт Python для обробки зображень та передачі даних. Переконайтеся, що скрипт працює коректно і надсилає дані на AWS.

Далі описано покрокову інструкцію. Підключіть пристрій Raspberry Pi до доступної мережі Інтернет. Після цього скористайтеся протоколом SSH для віддаленого входу в систему з іншого пристрою.

```
# if the console says that "REMOTE HOST IDENTIFICATION HAS CHANGED!"
# run this command to reset it
# ssh-keygen -R raspberrypi.local

ssh pi@raspberrypi.local
```

Створіть на пристрої Raspberry Pi папку під назвою "plant". Впевніться, що всі значення у файлі config.ini є вірними та актуальними. Перенесіть скрипт Python, тестове зображення, файли автентифікації AWS, конфігураційний файл та всі інші необхідні файли до папки проекту на Raspberry Pi за допомогою команди scp.

```
# on your host desktop

cd [your_desktop_pi_plant_growth_project_path]
scp -r ./* pi@raspberrypi.local:~/plant/
```

Після цього встановіть необхідні бібліотеки Python за допомогою інструменту pip. Наприклад, бібліотеку MQTT можна встановити таким чином.

```
pip install AWSIoTPythonSDK
```

3.5 Представлення результатів тестування ПЗ

Тестування є одним з найважливіших етапів розробки мікроконтролерної системи моніторингу росту рослин з використанням AWS. У процесі тестування проводиться комплексна перевірка всіх модулів системи для забезпечення їх коректного функціонування та взаємодії. Мета тестування полягає у виявленні та виправленні помилок, оцінці продуктивності та надійності системи, а також у підтвердженні відповідності проєкту встановленим вимогам та очікуванням.

Перед початком тестування необхідно підготувати всі необхідні компоненти системи, включаючи мікроконтролери, датчики, камери та інші апаратні засоби. Крім того, слід налаштувати програмне забезпечення, зокрема скрипти для обробки зображень та передачі даних на AWS, а також конфігураційні файли.

Процес тестування включає кілька етапів:

1) перевірка апаратних компонентів: перевірка правильності підключення датчиків, камер та інших пристроїв до мікроконтролера. Оцінка стабільності та надійності їх роботи;

2) тестування програмного забезпечення: запуск основного скрипта Python для перевірки коректності обробки зображень, вилучення пікселів та обчислення площі рослинного покриву. Верифікація правильності налаштувань у конфігураційних файлах;

3) передача даних на AWS: тестування з'єднання з AWS IoT, передача даних за допомогою протоколу MQTT, перевірка коректності відправлених повідомлень;

4) аналіз результатів: оцінка отриманих даних, їх відповідність очікуванням, виявлення можливих відхилень та помилок.

У ході тестування можуть бути виявлені різні помилки та недоліки у функціонуванні системи. До основних типів помилок належать:

– апаратні помилки: проблеми з підключенням або роботою датчиків та камер;

– програмні помилки: некоректна обробка зображень, помилки у коді, неправильні налаштування;

– мережеві помилки: проблеми з підключенням до AWS IoT, нестабільність з'єднання.

Кожна виявлена помилка ретельно аналізується, після чого вживаються заходи для її усунення. Виправлення помилок включає зміну конфігураційних налаштувань, корекцію коду, заміну або налаштування апаратних компонентів.

Після внесення всіх необхідних виправлень проводиться повторне тестування системи для підтвердження усунення помилок та перевірки її стабільності та надійності. Позитивні результати підсумкового тестування свідчать про готовність системи до використання в реальних умовах.

Давайте проведемо тестування нашої настройки. Відкрийте клієнт тесту MQTT в консолі IoT. Введіть device/[DEVICE_ID]/data у поле фільтра тем і натисніть кнопку "Підписатися". Тема повинна з'явитися в панелі підписок (рис. 3.6.). Залиште вікно відкритим.

Відкрийте консоль DynamoDB в іншій вкладці браузера. Клацніть "Дослідження елементів" і виберіть таблицю pi_green (рис. 3.6.). Також залиште вікно відкритим.

Поверніться на наш Raspberry Pi. Виконайте наступну команду. Вона тестує код та підключення до AWS за допомогою стандартного зображення.

```
cd /home/pi/plant/ && python /home/pi/plant/robot.py leaf_standard.jpeg
```

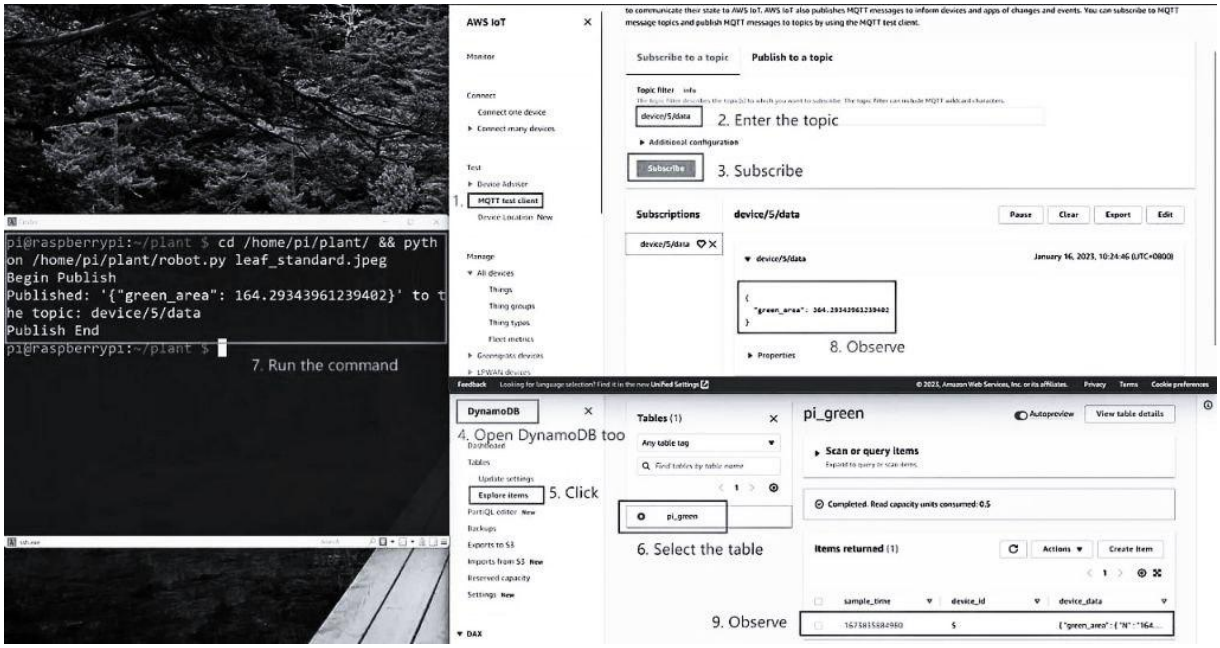


Рисунок 3.6 – Перевірка проекту зростання рослин за допомогою зображення за замовчуванням

Якщо все пройшло успішно, в консолі IoT автоматично з'явиться пункт (рисунок 3.6.). На вкладці DynamoDB також з'явиться пункт після того, як ми натиснемо кнопку оновлення. Це означає, що AWS тепер може отримувати дані про зелену зону з нашого Raspberry Pi.

3.6 Інструкції для користувачів

Настав час встановити камеру над рослинами. Ми краще закріпимо місце та кут камери так, щоб вона спостерігала за тією ж ділянкою землі протягом тривалого періоду (рисунок 1). Розмістіть внутрішні стандарти на нижній частині виду (рисунок 6). Мій скрипт camera_test.py може допомогти вам розмістити камеру, показуючи прямий відеопотік з неї.

Після налаштування апаратної частини ми можемо налаштувати значення порогів кольорів. Ідеальний набір порогів має виділяти цільовий колір та приховувати всі інші. Для внутрішніх стандартів ми можемо уникнути фонового шуму, обмеживши поле огляду лише нижньою областю.

Після завершення всіх налаштувань ми можемо знову протестувати нашу настройку за допомогою прямого відеопотоку.

```
cd /home/pi/plant/ && python /home/pi/plant/robot.py
```

Якщо результат не проходить перевірку на адекватність, нам потрібно знову налаштувати апаратну та програмну частини. Також, ми повинні перевірити, чи отримує DynamoDB результати.

Нарешті, якщо ми задоволені всією системою, ми повинні запланувати вимірювання за допомогою cron. Відкрийте редактор cron за допомогою наступної команди.

```
crontab -e
```

Наприклад, бажано проводити нові вимірювання щодня о 17:00. Цей час підійде, оскільки світло ще досить яскраве, але не таке сильне, щоб завадити роботі моєї білої стандартної основи.

```
0 17 * * * cd /home/pi/plant/ && python /home/pi/plant/robot.py >>
/home/pi/Documents/sensor.log 2>&1
```

3.7 Висновки

У даному розділі дипломної роботи було представлено детальний опис програмно-апаратної реалізації мікроконтролерної системи моніторингу росту рослин з використанням AWS. Починаючи з простої архітектури, що включає Raspberry Pi 4, вебкамеру, обліковий запис AWS та основний комп'ютер для програмування, було розглянуто кроки налаштування та конфігурації AWS, встановлення необхідного програмного забезпечення та інтеграцію з Raspberry Pi. Значна увага була приділена опису основного Python скрипта, який відповідає за зйомку зображень, аналіз кольорів та передачу даних на AWS.

Окремий підпункт був присвячений опису налаштування Raspberry Pi, включаючи встановлення операційної системи, необхідного програмного забезпечення та підключення до AWS. Детально розглянуто кроки налаштування та тестування системи, включаючи використання MQTT для передачі даних на AWS та перевірку результатів на DynamoDB.

Завершальний пункт розділу стосується випробування системи та планування продукції. Вказано на необхідність систематичного тестування для впевненості в роботі системи та можливість автоматизації вимірювань за допомогою cron.

У цілому, даний розділ роботи надає повний огляд процесу розробки та налаштування мікроконтролерної системи моніторингу росту рослин з використанням AWS.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було створено систему моніторингу росту рослин з використанням мікроконтролера Raspberry Pi та послуг AWS.

У першому розділі проведено аналіз сучасних тенденцій у сільському господарстві та проблем, пов'язаних з моніторингом росту рослин. Виявлено потребу в розробці ефективних та доступних технологій для відстеження розвитку рослин. Зазначено, що існують різні підходи до створення мікроконтролерних систем для моніторингу росту рослин, включаючи системи на базі Arduino та Raspberry Pi, а також комерційні рішення. Проаналізовано переваги та недоліки кожного з цих підходів.

У другому розділі розглянуто процес проектування програмно-технічного засобу для системи моніторингу росту рослин. Визначено функціональні та нефункціональні вимоги до системи, розроблено архітектуру програмно-апаратного забезпечення та проведено верифікацію вимог. Зазначено важливість дотримання встановлених вимог для успішного виконання проекту та подальшого розвитку програмно-технічного засобу.

У третьому розділі описано реалізацію мікроконтролерної системи моніторингу росту рослин з використанням AWS. Надано детальний опис апаратної та програмної складових системи, процесу налаштування та тестування, а також висновки щодо ефективності розробленого рішення. Зазначено необхідність систематичного тестування для впевненості в роботі системи та можливість автоматизації вимірювань за допомогою cron.

Загальний висновок полягає у тому, що розроблена система є ефективним інструментом для моніторингу росту рослин, що може бути успішно використана у сільському господарстві та наукових дослідженнях. Робота включає в себе комплексний аналіз, проектування та реалізацію системи, яка відповідає вимогам та потребам користувачів у сільському господарстві.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Решетюк В. М., Лендєл Т. І., Куляк Б. В. Алгоритм прийняття рішень для проведення крапельного зрошення в теплиці із використанням інформації про стан рослини. *Науковий вісник НУБіП України. Серія: Техніка та енергетика АПК,* 2017. Вип. 261. С. 111–120.
2. Andreas W., Michael W. Amazon Web Services in Action, Third Edition: An In-depth Guide to AWS. Simon and Schuster, 2023. 552 p.
3. Brad S. Amazon Unbound: Jeff Bezos and the Invention of a Global Empire. Simon and Schuster, 2022. 512 p.
4. Brian D. Bezonomics: How Amazon Is Changing Our Lives and What the World's Best Companies Are Learning from It. Simon and Schuster, 2020. 336 p.
5. Поджаренко В.О., Кучерук В.Ю., Севастьянов В.М. Основи мікроконтролерної техніки. Вінниця: 2006. 23 с.
6. Шарапа О. В., Бердников А. Г. Модель системи управління технологічним процесом в тепличному агропромисловому комплексі. Вісник Харківського національного університету імені В.Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. Вип. 47, 2020. 135 с.
7. Ясінський Р.В., Осухівська Г.М., Паламар А.М., Величко Д.В. Комп’ютерна система для контролю параметрів мікроклімату теплиць на основі інтернету речей. Актуальні задачі сучасних технологій : збірник тез доповідей XI міжнародної науково-технічної конференції молодих учених та студентів, Тернопіль: ФОП Паляниця В. А, 2022. 177. с.
8. Заєць Н. А., Дудник А. О., Якименко І. Ю. Експериментально-статистичне дослідження теплиці як об’єкта керування з метою підвищення ресурсоефективності виробництва. *Енергетика і автоматика,* 2017. № 4. С. 200–211.

9. Науковці з Данії розробили розумну систему зрошення для теплиць. URL: https://superagronom.com/news/13793-naukovtsi-z-daniyi-rozrobili-rozumnu-sistemuzroshennya-dlya-teplits (дата звернення: 22.04.2024).
10. IntelliGrow – Cloud-based Greenhouse Crop Management Application. URL: https://autogrow.com/our-products-solutions/intelligrow (дата звернення: 24.04.2024).
11. Майданюк В.П., Пєтух А.М. Інтерфейс Користувач-комп'ютер: навч. посіб. Вінниця: 1999. 66 с.
12. Paul J. J. Distributed Serverless Architectures on AWS. Berkeley, CA: 2023. 172 p.
13. Сметанка Ю. А. Архітектура системи автентифікації користувачів інформаційної системи на основі віддаленого виділеного сервера. Тернопіль: 2019. 133 с.
14. Abhishek S., Chaman S., Nadeem K., Rohit C., Gurjeet S. EPRA *International Journal of Multidisciplinary Research.* A review paper on aws*,* 2024. № 10. P. 0-6.
15. Sudip C., Aithal P. S. *International Journal of Case Studies in Business, IT, and Education*. Let Us Create an IoT Inside the AWS Cloud, 2023. P. 211- 219.
16. John C., Mike R. Programming AWS Lambda: Build and Deploy Serverless Applications with Java, 2020. 278 с.
17. Amol K. Amazon Athena : Serverless Architecture and Troubleshooting. *International Journal of Computer Trends and Technology*. 2023. P. 57-61.
18. Suresh P. K., Jitendra K. Performance Analysis of SSL/TLS Crypto Libraries: Based on Operating Platform. *Journal of Scientific Research.* 2022. P. 1-10.
19. Muhammad T., Mishal S., Hajar H. Analysis of research on amazon AWS cloud computing seller data security. *International Journal of Research in Engineering and Innovation*. 2020. P. 131-136.
20. George Reese. Java Database Best Practices — O'Reilly Media, 2003. URL: http://oreilly.com/catalog/9780596005221/ index.html (дата звернення: 16.03.2024).

21. Villa-Henriksen A., Edwards G.T., Pesonen L.A., Green O., Sørensen C.A.G., Internet of Things in arable farming: Implementation, applications, challenges and potential, 2020. P. 60–84 URL: https://doi.org/10.1016/j.biosystemseng.2019.12.013 (дата звернення: 29.03.2024).
22. Nayyar A., Puri V., Smart farming: IoT based smart sensors agriculture stick for live temperature and moisture monitoring using Arduino, cloud computing & solar technology. In Proc. of The International Conference on Communication and Computing Systems, 2016. 121 p. URL: https://doi.org/10.1201/9781315364094-121 (дата звернення: 01.04.2024).
23. Tanveer, A., Choudhary, A., Pal, D., Gupta, R., Husain, F. Automated farming using microcontroller and sensors, 2015. P. 21–30.
24. Elazhary, Hanan. Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. *Journal of Network and Computer Applications*, 2019. P. 105-140.
25. Haripriya, A. P., Kulothungan K. Secure-MQTT: an efficient fuzzy logic-based approach to detect DoS attack in MQTT protocol for internet of things. Eurasip: *Journal on Wireless Communications and Networking,* 2019. 90 p.
26. Isaac W.A., Varshney S., Khan E. An IoT based system for remote monitoring of soil characteristics. In the International Conference on Information Technology. India: 2016. P. 316-320. URL: https://doi.org/10.1109/INCITE.2016.7857638 (дата звернення: 10.04.2024).
27. Suma N., Samson S.R., Saranya S., Shanmugapriya G., Subhashri R. IoT based smart agriculture monitoring system. *International Journal on Recent and Innovation Trends in Computing and Communication:* P. 177-181
28. Thamaraimanalan T., Vivekk S.P., Satheeshkumar G., Saravanan P. Smart garden monitoring system using IoT. *Asian Journal of Applied Science and Technology*: 2018. P. 186-192

29. Kasiviswanathan S., Ramalingam D. Development and application of user review quality model for embedded system. Microprocessors and Microsystems: 2020. 74. p. URL: https://doi.org/10.1016/j.micpro.2020.103029 (дата звернення: 12.04.2024).
30. Kavitha B. C., Vallikannu R., Sankaran K.S. Delay-aware concurrent data management method for IoT collaborative mobile edge computing environment. Microprocessors and Microsystems: 2020. 74 p. URL: https://doi.org/10.1016/j.micpro.2020.103021 (дата звернення: 14.04.2024).
31. Jacksonet R. D. Canopy temperature as a crop water stress indicator. *Water Resources Research*: 1981. P. 1133-1138.
32. Gavito M. E. *I*nteractive effects of soil temperature, atmospheric carbon dioxide and soil N on root development, biomass and nutrient uptake of winter wheat during vegetative growth. *Journal of Experimental Botany:* 2001. P. 32-38.
33. Nisha, G., Megala, J. Wireless sensor network based automated irrigation and crop field monitoring system. *Sixth International Conference on Advanced Computing* , 2014. P. 189–194.
34. Tanveer, A., Choudhary, A., Pal, D., Gupta, R., Husain, F. Automated farming using microcontroller and sensors. 2015. P. 21–30.
35. Jiao, J., Ma, H.M., Qiao Y., Du Y.L., Kong, W., Wu Z.C. Design of Farm Environmental Monitoring System Based on the Internet of Things. *Advance Journal of Food Science and Technology*:2014. P. 368–373.
36. Pavithra D.S., Srinath M.S. GSM based automatic irrigation control system for efficient use of resources and crop planning by using an android mobile. *Journal. Mech. Civil*: 2014. P. 49–55.
37. Naik P. Arduino based automatic irrigation system using IoT. *Journal*. Sci:2017. P. 881–886.
38. Ankit K. V., Bhagavan, K., Akhil, V., Amrita, S. Wireless network based smart irrigation system using IOT. *Journal*. 2018. P. 342–345.

39. Rao R.N., Sridhar B. IoT based smart crop-field monitoring and automation irrigation system. *Second International Conference on Inventive Systems and Control.* 2018. P. 478–483.
40. Banumathi P., Saravanan D., Sathiyapriya M., Saranya V. An android based automatic irrigation system using bayesian network with SMS and voice alert. *Journal*. 2017. P. 573–578.
41. Bharathi G., Prasunamba C.G. Automatic irrigation system for smart city using PLC AND SCADA. *Journal* 2917. P. 309–314.
42. Kumar B.D., Srivastava P., Agrawal R., Tiwari V. Microcontroller based automatic plant irrigation system. *Journal* 2017. P. 1436–1439.
43. Nayyar A., Puri V. Smart farming: IoT based smart sensors agriculture stick for live temperature and moisture monitoring using Arduino, cloud computing & solar technology. In Proc. of The International Conference on Communication and Computing Systems URL: https://doi.org/10.1201/9781315364094-121 (дата звернення: 20.04.2024).
44. Prathibha S. R., Hongal A., Jyothi M. P. IoT based monitoring system in smart agriculture. In 2017 international conference on recent advances in electronics and communication technology, 2017. URL: https://doi.org/10.1109/ICRAECT.2017.52 (дата звернення: 21.04.2024).
45. Kaburuan E.R., Jayadi R. A design of IoT-based monitoring system for intelligence indoor micro-climate horticulture farming in Indonesia. *Procedia Computer Science*, 2019. P. 459–464. URL: https://doi.org/10.1016/j.procs.2019.09.001 (дата звернення: 22.04.2024).
46. Mishra D., Pande T., Agrawal K.K., Abbas A., Pandey A.K., Yadav R.S. Smart agriculture system using IoT, in Proceedings of the Third International Conference on Advanced Informatics for Computing Research, 2019. P. 1–7. URL: https://doi.org/10.1145/3339311.3339350 (дата звернення: 25.04.2024).
47. Yoon C., Huh M., Kang S.G., Park J., Lee C. Implement smart farm with IoT technology, in 2018 20th International Conference on Advanced Communication

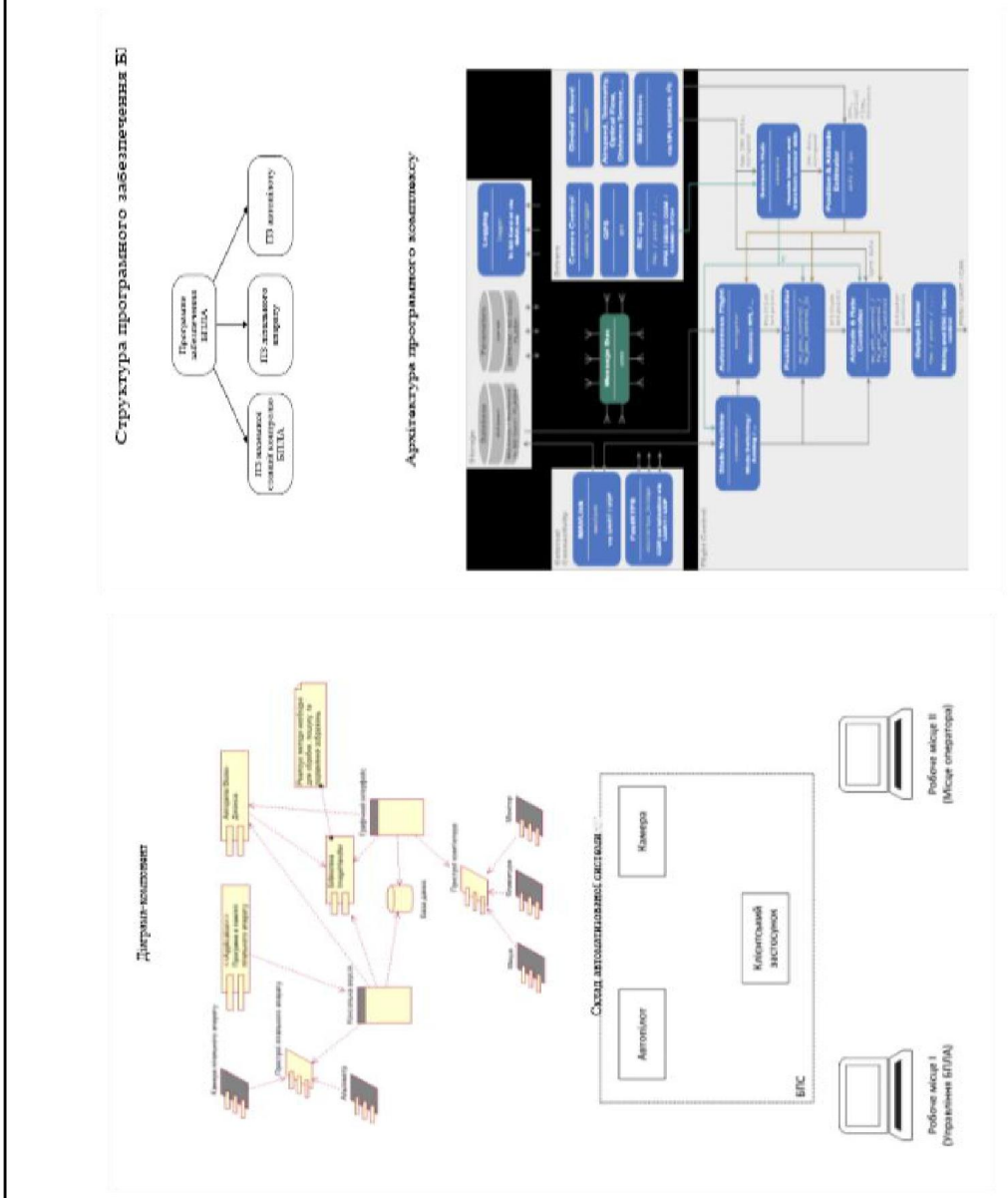
Technology. Chuncheon, Korea: 2018. P. 749–752. URL: https://doi.org/10.23919/ICACT.2018.8323908 (дата звернення: 09.05.2024).

48. Ignatius P. Penggunaan Raspberry Pi Sebagai Web Server Pada Rumah Untuk Sistem Pengendalia Lampu Jarak Jauh dan Pemantauan Suhu. *Jurnal Ilmiah Elektroteknika*, 2014. № 1 P. 111-124.
49. Kekre A., Gawre S. K. Solar photovoltaic remote monitoring system using IOT. *International conference on recent innovations in signal processing and embedded systems*, 2017. P. 619- 623.
50. Purusothaman S. D., Rajesh R., Bajaj K. K., Vijayaraghavan V. Implementation of Arduino-based multi-agent system for rural Indian microgrids. *Innovative Smart Grid Technologies.* Asia: 2019. P. 1-5.
51. MQTT: The Standard for IoT Messaging. URL: http://mqtt.org/

Додаток А

(обов’язковий)

Копія креслення «Архітектура ПЗ проєкту»



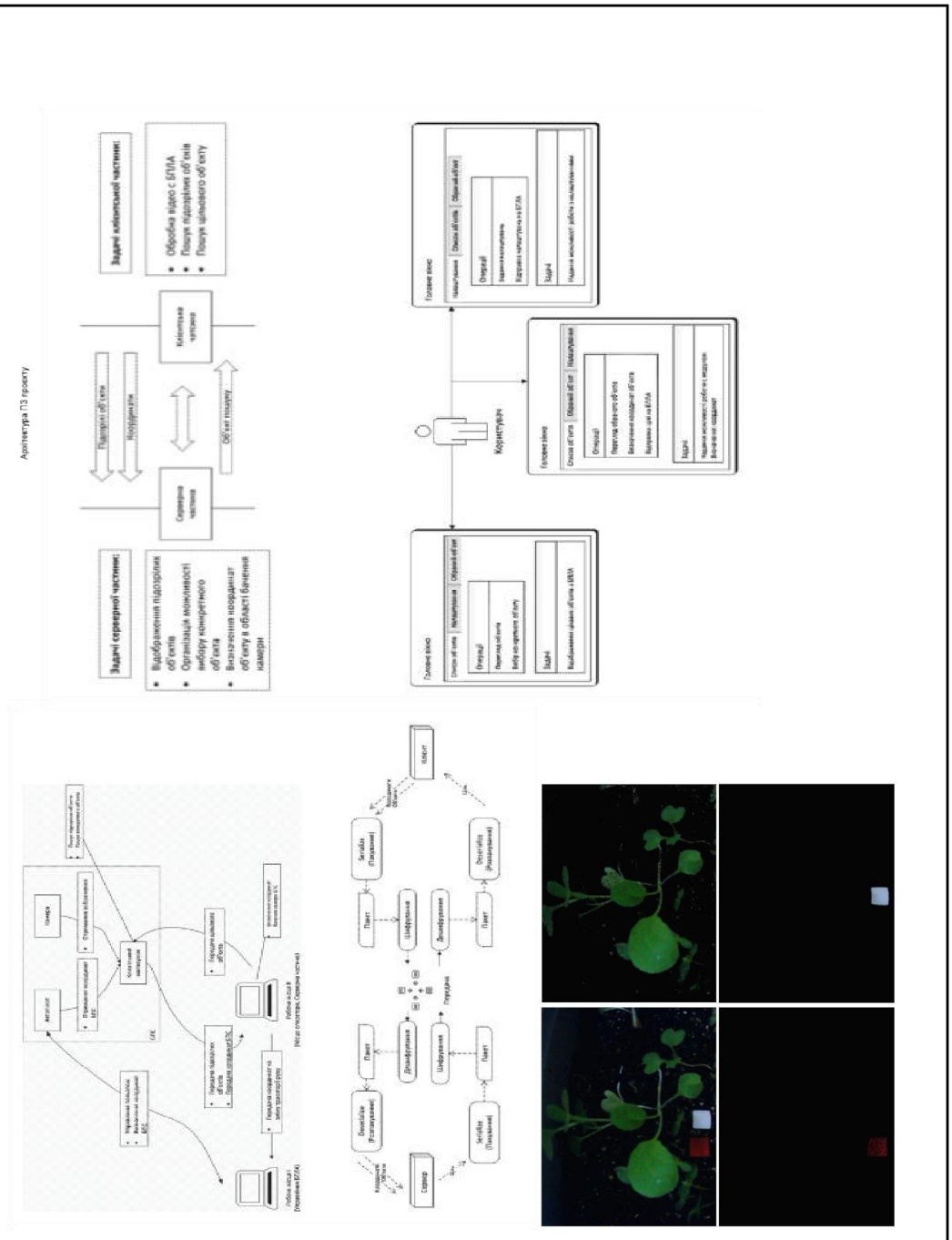
					КвРКІ 200236.20.02.11 ПЗ	
Зм.	Арк.	№ докум.	Підпис	Дата		
Затвердив		Говорущенко Т.О.			Мікроконтролерна система моніторингу росту рослин з використанням AWS Архітектура ПЗ проєкту	Стадія: РП; Аркуш: 1; Аркушів: 1
Перевірив		Говорущенко Т.О.				
Н.Контроль		Засорінова І.О.				ХНУ, ГР, КІ2-20-2
Розробив		Лозова Х. Р.				

КвРКІ 200236.20.02.11 ПЗ

Додаток Б

(обов’язковий)

Копія креслення «Архітектура ПЗ для мікроконтролерної системи»



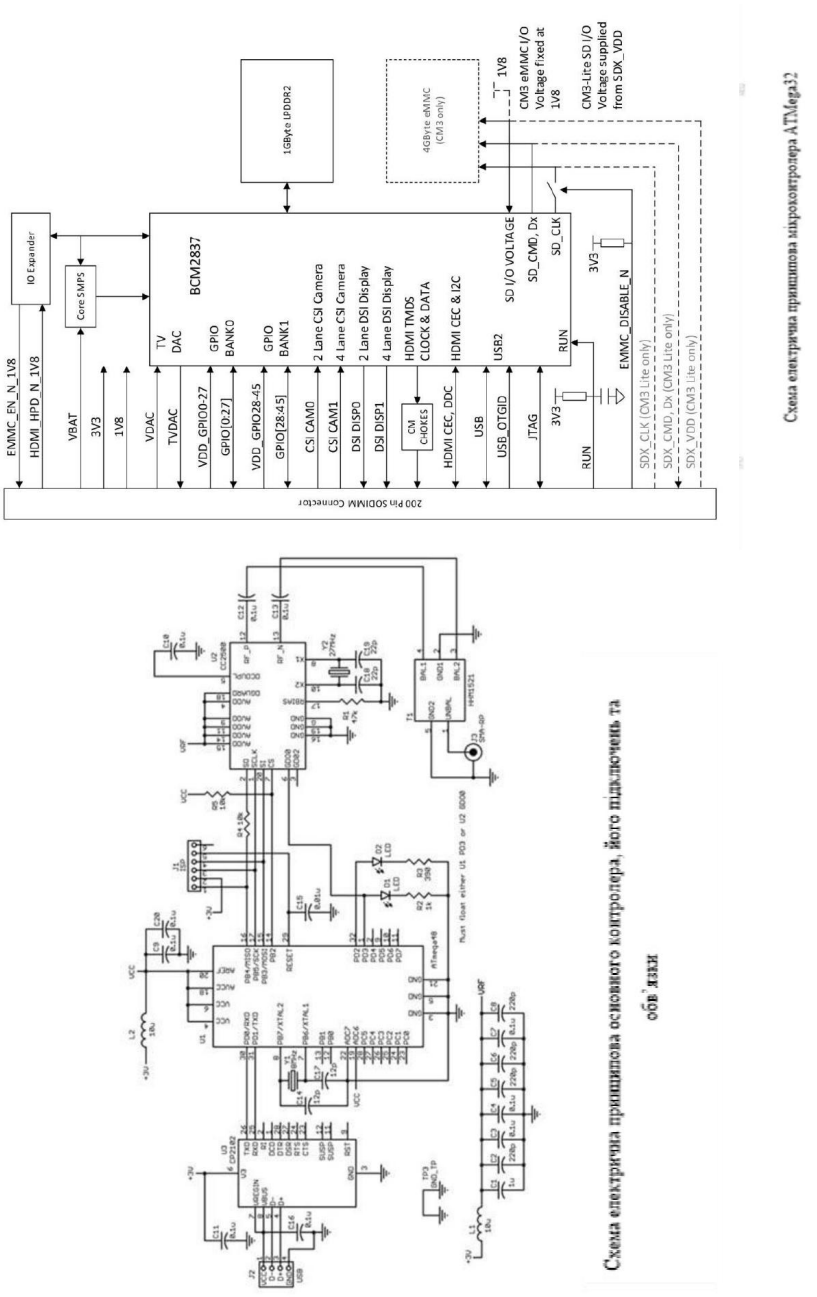
КвРКІ 200236.20.02.11 ПЗ

Зм.	Арк.	№ докум.	Підпис	Дата	КвРКІ 200236.20.02.11 ПЗ			
Затвердив		Говорущенко Т.О.			Мікроконтролерна система моніторингу росту рослин з використанням AWS Архітектура ПЗ для мікроконтролерної системи	Стадія	Аркуш	Аркушів
Перевірив		Говорущенко Т.О.				РП	1	1
Н.Контроль		Засорнова І.О.				ХНУ, ГР, КІ2-20-2		
Розробив		Лозова Х. Р.						

Додаток В

(обов'язковий)

Копія креслення «Апаратне забезпечення проєкту»



					КвРКІ 200236.20.02.11 ПЗ	
Зм.	Арк.	№ докум.	Підпис	Дата		
Затвердив		Говорущенко Т.О.			Мікроконтролерна система моніторингу росту рослин з використанням AWS Апаратне забезпечення проєкту	Стадія: РП; Аркуш: 1; Аркушів: 1
Перевірив		Говорущенко Т.О.				
Н.Контроль		Засорнова І.О.				ХНУ, ГР, КІ2-20-2
Розробив		Лозова Х. Р.				

КвРКІ 200236.20.02.11 ПЗ

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016328991

Дата перевірки:
06.06.2024 19:28:17 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.06.2024 19:29:08 EEST

ID користувача:
100005591

Назва документа: **Лозова_Мікроконтролерна система моніторингу росту рослин з використанням AWS**

Кількість сторінок: **65** Кількість слів: **11905** Кількість символів: **95779** Розмір файлу: **2.34 MB** ID файлу: **1016128438**

15.1%
Схожість

Найбільша схожість: **2.75%** з Інтернет-джерелом (https://dokumen.pub/computational-intelligence-in-data-science-third.

14.4% Джерела з Інтернету	635	Сторінка 67
2.14% Джерела з Бібліотеки	119	Сторінка 70

1.59% Цитат

Цитати	9	Сторінка 71
Посилання	1	Сторінка 71

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	1

Thu Jun 06 17:42:02 EEST 2024, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

ID: 128890 Назва: БКР Мікроконтролерна система моніторингу росту рослин з використанням AWS Додано в БД: 2024-06-06 Автора: Х. Р. Лозова Керівники: В. В. Яцків Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	82349	692	2590 (3%)	28 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Лозова Христина Романівна

Тема: Мікроконтролерна система моніторингу росту рослин з використанням AWS.

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___57___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розробка мікроконтролерної системи моніторингу росту рослин з використанням AWS.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області (проаналізовано теорію мікроконтролерної системи моніторингу росту рослин з використанням AWS, а також розроблення і опис схем електричних функціональної і принципової) та виконано постановку задачі дослідження. В другому розділі кваліфікаційної роботи проведено моделювання та проектування програмно-технічного засобу для системи моніторингу росту рослин, а саме: формалізований опис програмно-технічного засобу для системи моніторингу росту рослин з використанням мікроконтролерів та AWS; розроблено змістовну схему алгоритму, що визначає основні кроки моніторингу та збір даних;роведено кодування вершин змістовної схеми алгоритму; розроблено мікрооперацийну схему алгоритму; розроблено мікрокоманди для управління мікроконтролером; розроблено мікрокомандну схему алгоритму; побудовано основну таблицю функціонування системи; спроєктовано граф-схему переходів мікроконтролера; складено систему рівнянь переходів; визначено необхідну кількість елементів пам'яті для реалізації функцій системи; проведено кодування внутрішніх

станів мікроконтролера; розроблено структурну схему мікроконтролерної системи; побудовано схему мікроконтролерної системи. В третьому розділі кваліфікаційної роботи виконано апаратну реалізацію мікроконтролерної системи моніторингу росту рослин з використанням AWS, а саме: розроблено ефективне програмне забезпечення для взаємодії мікроконтролера з AWS; налаштовано систему для автоматичного вимірювання та передачі даних в хмарне середовище AWS.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: Недостатньо уваги приділено аналізу хмарних сервісів.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: немає

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Бойко Юлій Миколайович, професор кафедри технічної кібернетики, інформаційних технологій, ЛКУ

"07" 06 2024 р. (підпис)

Завідувачу кафедри КІСП
д-ру техн.наук, проф. Говорущенко Т. О.

Лозової Х. Р.
ПІБ здобувача вищої освіти

ФПКТС, 4 курсу, групи КІ2-20-2

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність плагіату ознайомлений(а) та надаю свою згоду на обробку та зберження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

06.06.24
дата

підпис

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Мікроконтролерна система моніторингу росту рослин з використанням AWS

Автор: Лозова Христина Романівна

Спеціальність: 123 – Компютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Яцків Василь Васильович, д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 15.1% і адресується до 754 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ В. В. Яцків

Гарант ОП _____ С. М. Лисенко

Завідувач кафедри КІІС _____ Т. О. Говорущенко