НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету (Директор ННІ)	**Завідувач кафедри**
Інформаційних технологій (назва факультету(ННІ))	Комп'ютерних систем, мереж та кібербезпеки (назва кафедри)
_____ Болбот І.М., д.т.н, проф. (підпис) (ПІБ, вчене звання і ступінь)	_____ Касаткін Д.Ю., к. пед.н., доц. (підпис) (ПІБ, вчене звання і ступінь)
«___»_____2025 р.	«___»_____2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Дослідження системи резервного зберігання даних мобільної платформи на основі мікроконтролера Raspberry pi »

Спеціальність 123 «Комп’ютерна інженерія»
(код і найменування)

Освітня програма Комп’ютерні системи та мережі
(назва)

Орієнтація освітньої програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

к. пед.н., доц	_____	Касаткін Д.Ю
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Керівник магістерської кваліфікаційної роботи

к. пед.н., доц	_____	Касаткін Д.Ю
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Виконав _____ Мартинюк В.В.
(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних систем, мереж та кібербезпеки
к.пед.н., доц. _____ Касаткін Д.Ю.
(вчене звання і ступінь) (підпис) (ПІБ)
«__» _____ 20__ р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Мартинюку Владиславу Вікторовичу
(прізвище, ім'я, по батькові)

Спеціальність 123 «Комп'ютерна інженерія»
(код і найменування)

Освітня програма Комп'ютерні системи та мережі
(назва)

Орієнтація освітньої програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи: «Дослідження системи резервного зберігання даних мобільної платформи на основі мікроконтролера Raspberry pi »

затверджена наказом ректора НУБіП України від "29" жовтня 2024р. № 1941 «С»

Термін подання завершеної роботи на кафедру 14 листопада 2025 р.

Вихідні дані до магістерської кваліфікаційної роботи _____

Перелік питань, що підлягають дослідженню:

1. _____
2. _____
3. _____

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання " 29 **"** жовтня 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Касаткін Д.Ю.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Мартинюк В.В.
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Сучасний аграрний сектор, зокрема тепличні господарства, перебуває у фазі активної цифрової трансформації. Ефективність вирощування культур у закритому ґрунті напряму залежить від здатності точно контролювати та підтримувати стабільні параметри мікроклімату — температуру, вологість, освітленість та тиск. Збір цих даних у режимі реального часу є критично важливим для прийняття агрономами оперативних управлінських рішень.

Водночас, постійний збір даних генерує великі масиви інформації (часові ряди), які самі по собі стають цінним активом. Втрата цих історичних даних через збій обладнання, пошкодження носія або програмну помилку унеможливлює ретроспективний аналіз, виявлення довгострокових тенденцій та погіршує якість прогнозування.

Традиційні серверні рішення для моніторингу та резервного копіювання часто є дорогими у впровадженні та обслуговуванні, а також енергозатратними, що робить їх нерентабельними для багатьох малих та середніх агропідприємств. Виникає потреба у розробці мобільної, автономної та економічно ефективної системи, яка б поєднувала функції збору даних та їх надійного резервного зберігання. Використання одноплатних комп'ютерів, таких як Raspberry Pi, для цих завдань є перспективним напрямком, що дозволяє створити гнучке та масштабоване рішення.

Мета і задачі дослідження.

Метою дипломної роботи є розробка та дослідження системи резервного зберігання даних з сенсорів мікроклімату теплиці (BME280, BH1750), реалізованої на мобільній платформі Raspberry Pi, з організацією захищеного веб-доступу до візуалізованих даних.

Для досягнення поставленої мети було визначено наступні задачі:

- Проаналізувати функціональні задачі систем моніторингу в аграрному секторі та дослідити особливості IoT-платформ на базі одноплатних комп'ютерів.
- Виконати огляд сучасних баз даних часових рядів (Time-Series Databases) та проаналізувати методи та схеми резервного копіювання даних.
- Обґрунтувати вибір апаратної платформи (Raspberry Pi) та програмного стеку (Python, InfluxDB, Grafana) для реалізації системи.
- Розробити апаратну частину системи, включаючи схеми підключення сенсорів BME280 та BH1750 до Raspberry Pi.
- Розробити та налагодити програмне забезпечення для автоматичного збору даних з сенсорів та їх запису до бази даних InfluxDB.
- Реалізувати та протестувати механізм автоматизованого резервного копіювання бази даних за схемою 7-4-4.
- Налаштувати систему візуалізації даних (Grafana) та організувати до неї захищений мобільний доступ за допомогою технології Cloudflare Zero Trust.
- Провести тестування розробленого прототипу системи та оцінити його працездатність.

Об'єкт дослідження – процес збирання, обробки та резервного зберігання аграрних даних (параметрів мікроклімату).

Предмет дослідження – методи та засоби реалізації мобільної системи моніторингу та резервного зберігання даних на базі апаратної платформи Raspberry Pi та програмного стеку InfluxDB/Grafana.

Методи дослідження. Для вирішення поставлених задач використовувався комплексний підхід, що включав:

- **методи системного аналізу** (для огляду існуючих рішень та обґрунтування вибору компонентів системи);

- **методи цифрової схемотехніки** (при розробці схем підключення апаратних компонентів);
- **методи об'єктно-орієнтованого програмування** (при написанні скриптів збору даних на мові Python);
- **методи адміністрування систем та баз даних** (при розгортанні стеку InfluxDB/Grafana та налаштуванні резервного копіювання);
- **метод експериментального дослідження** (для тестування працездатності розробленого прототипу).

Наукова новизна одержаних результатів полягає у розробці комплексної архітектури мобільної системи, що поєднує прямий збір даних з сенсорів на Raspberry Pi, їх акумуляцію у спеціалізованій базі даних часових рядів (InfluxDB) та реалізацію гібридної схеми резервного копіювання (7-4-4). Новизна також полягає у застосуванні технології Zero Trust (на базі Cloudflare Tunnel) для організації захищеного віддаленого доступу до даних моніторингу, що не вимагає наявності статичної IP-адреси та відкриття портів на стороні об'єкта, що значно підвищує безпеку системи.

Практичне значення одержаних результатів полягає у розробці готового до впровадження, бюджетного та енергоефективного прототипу системи моніторингу та резервного зберігання даних. Запропоноване рішення дозволяє агрономам оперативно аналізувати мікроклімат теплиці , отримувати доступ до візуалізованих графіків (температури, вологості, тиску, освітленості) з будь-якого мобільного пристрою та бути впевненими у збереженні історичних даних. Це сприяє прийняттю обґрунтованих управлінських рішень та слугує інструментом для цифровизації аграрного сектору.

Структура та обсяг роботи. Дипломна робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел. Загальний обсяг роботи становить 67 сторінок, у тому числі 13 рисунків . Список використаних джерел налічує 28 найменувань.

1 АНАЛІЗ ЗАДАЧІ МОНІТОРИНГУ ТА РЕЗЕРВНОГО ЗБЕРІГАННЯ ДАНИХ В АГРАРНОМУ СЕКТОРІ

1.1. Функціональні задачі систем моніторингу мікроклімату в тепличних господарствах

Тепличне господарство є керованим середовищем, головна мета якого — створення оптимальних умов для росту рослин, незалежно від зовнішніх кліматичних факторів. Ефективність теплиці прямо залежить від здатності точно вимірювати, аналізувати та підтримувати баланс параметрів мікроклімату. Сучасні системи моніторингу виходять за межі простого вимірювання; вони вирішують комплекс взаємопов'язаних функціональних задач, що є основою для точного землеробства (precision agriculture).

Ключовими функціональними задачами таких систем є:

- **Безперервний збір даних (Data Acquisition).** Це первинна і найбільш фундаментальна задача, яка включає автоматичне та регулярне вимірювання фізичних параметрів. До основних показників мікроклімату належать:
    - **Температура повітря та ґрунту:** Впливає на швидкість метаболізму, фотосинтезу та дихання рослин.
    - **Відносна вологість повітря:** Критичний параметр для запобігання стресу рослин та розвитку грибкових захворювань, таких як сіра гниль.
    - **Рівень освітленості:** Вимірюється в люксах або як фотосинтетично активна радіація (PAR) і є визначальним для процесу фотосинтезу.
    - **Атмосферний тиск:** Хоча його вплив менш прямий, він корелює з погодними умовами і може бути корисним для комплексного аналізу. У даній роботі збір цих параметрів покладено на сенсори BME280 (температура, вологість, тиск) та BH1750 (освітленість).

- **Аналіз та візуалізація даних (Analysis and Visualization).** Сирі дані, отримані з сенсорів, мають низьку цінність без їх належної обробки та представлення. Ця задача вирішується через:
    - **Візуалізацію в реальному часі:** Надання інформації у вигляді інтуїтивно зрозумілих графіків та панелей індикаторів (дашбордів). Це дозволяє агроному миттєво оцінити поточний стан теплиці, не аналізуючи масиви чисел .
    - **Історичний аналіз:** Накопичені дані (часові ряди) дозволяють аналізувати тенденції, виявляти добові та сезонні цикли, а також знаходити кореляції між змінами мікроклімату та станом рослин або врожайністю .

- **Підтримка прийняття рішень та управління (Decision Support and Control).** Дані моніторингу є основою для прийняття обґрунтованих управлінських рішень.
    - **Сигналізація про відхилення (Alerting):** Система повинна негайно сповіщати персонал (через email, SMS або push-сповіщення) про вихід будь-якого параметра за встановлені порогові значення. Наприклад, про критичне падіння температури вночі, що може призвести до загибелі врожаю .
    - **Основа для автоматизації:** Дані моніторингу слугують вхідною інформацією для систем автоматичного керування виконавчими механізмами (обігрівачами, вентиляторами, системами поливу та затінення).

- **Надійне зберігання та архівування даних (Data Storage and Archiving).** Ця задача є центральною для нашої дипломної роботи.

- **Оперативне зберігання:** Дані мають зберігатися у спеціалізованій базі, оптимізованій для роботи з часовими рядами, що забезпечує швидкий запис та вибірку.
- **Довготривале архівування та резервування:** Забезпечення цілісності та доступності історичних даних є критично важливим. Втрата архіву знецінює можливість довгострокового аналізу. Тому система повинна включати надійний механізм резервного копіювання для захисту від апаратних збоїв, програмних помилок або людського фактору .

Вирішення цього комплексу задач дозволяє перейти від реактивного до проактивного управління тепличним господарством, оптимізувати використання ресурсів та підвищити його загальну рентабельність.

1.2. Системи моніторингу

Системи моніторингу (СМ) є ключовим елементом сучасних технологій управління, що дозволяють спостерігати, вимірювати та контролювати процеси, які відбуваються у складному об'єкті. В контексті аграрного сектору, СМ — це програмно-апаратні комплекси, призначені для автоматизованого збору та аналізу даних про стан середовища та об'єктів вирощування.

Загальна архітектура сучасної системи моніторингу, особливо в контексті Інтернету речей (IoT), зазвичай включає чотири основні рівні:

- **Рівень сенсорів (Perception Layer):** Фізичні пристрої (датчики, сенсори), що безпосередньо взаємодіють з середовищем та перетворюють фізичні величини (температуру, вологість, освітленість) у електричні сигнали.
- **Рівень збору та передачі даних (Network Layer):** Включає апаратні засоби (наприклад, мікроконтролери, одноплатні комп'ютери) та комунікаційні технології (Wi-Fi, LoRaWAN, Bluetooth), які збирають дані з сенсорів та передають їх на наступний рівень.

- **Рівень обробки (Processing Layer):** Серверна частина системи, де дані зберігаються (у базах даних), обробляються, аналізуються та перетворюються на корисну інформацію.
- **Рівень застосунків (Application Layer):** Інтерфейс користувача (веб-панелі, мобільні додатки), що надає візуалізовані дані, сповіщення та інструменти управління.

У нашому проекті Raspberry Pi виконує функції другого та третього рівнів одночасно: він збирає дані з сенсорів (рівень 2) і тут же обробляє та зберігає їх у локальній базі даних (рівень 3), що є характерною рисою мобільних автономних платформ.

1.2.1. Класифікаційні види моніторингу

Моніторинг як процес можна класифікувати за багатьма ознаками, що допомагає краще визначити його цілі та методи реалізації.

За ступенем автоматизації:

- **Ручний моніторинг:** Повністю покладається на людину, яка періодично знімає показники приладів. Цей метод неточний, трудомісткий та не забезпечує необхідної частоти вимірювань.
- **Автоматизований моніторинг:** Система збирає дані, але їх інтерпретація та прийняття рішень залишаються за оператором.
- **Автоматичний моніторинг:** Система не лише збирає, але й аналізує дані та може самостійно ініціювати керуючі дії (наприклад, увімкнути обігрів). Розроблювана система є автоматизованою (автоматичний збір та візуалізація) з елементами автоматичного моніторингу (через майбутнє налаштування сповіщень).

За часовою характеристикою:

- **Періодичний (дискретний):** Вимірювання проводяться через визначені інтервали часу (наприклад, кожні 15 хвилин).
- **Безперервний (потоковий):** Дані надходять постійним потоком.
- **Моніторинг у реальному часі (Real-time):** Система, що здатна зібрати, обробити та надати дані за час, який є достатньо малим, щоб встигнути вплинути на процес. Для тепличних господарств, де процеси є відносно повільними, моніторинг з інтервалом у 1-5 хвилин вважається моніторингом у реальному часі.

За просторовим охопленням:

- **Точковий моніторинг:** Дані збираються з однієї локації (одного набору сенсорів).
- **Розподілений моніторинг:** Використовується мережа сенсорів, розміщених у різних точках об'єкта, для отримання повної картини стану середовища. Наш проект реалізує точковий моніторинг, але архітектура дозволяє легке масштабування до розподіленого шляхом додавання нових сенсорних вузлів.

1.2.2. Функціональні класи моніторингу

Залежно від головної мети, яку переслідує моніторинг, його можна поділити на кілька функціональних класів.

- **Моніторинг стану (State Monitoring):** Найпоширеніший клас. Його мета – відповісти на питання: «Що відбувається зараз?». Він фіксує поточні значення параметрів об'єкта. Візуалізація даних на дашборді Grafana є класичним прикладом моніторингу стану.
- **Моніторинг продуктивності (Performance Monitoring):** Оцінює ефективність роботи системи. В аграрному контексті це може бути аналіз співвідношення витрачених ресурсів (електроенергії, води) до отриманого результату (приріст біомаси, врожайність).

- **Моніторинг відмов (Fault Monitoring / Anomaly Detection):** Спеціалізований клас, орієнтований на раннє виявлення аномалій та відхилень від норми. Аналізуючи історичні дані, система може виявити нетипову поведінку (наприклад, занадто швидке падіння температури, що свідчить про поломку обігрівача) ще до того, як параметри вийдуть за критичні межі.
- **Моніторинг відповідності (Compliance Monitoring):** Забезпечує контроль за дотриманням певних норм або стандартів. Наприклад, для сертифікації органічної продукції може вимагатися надання журналу (логу) даних, який підтверджує, що певні хімікати не використовувались, або що умови зберігання продукції відповідали нормам. Розроблювана в дипломній роботі система в першу чергу реалізує моніторинг стану, проте накопичені дані та їх надійне резервне зберігання створюють необхідну базу для подальшого впровадження моніторингу продуктивності та відмов.

1.3. Особливості мобільних IoT-платформ для збору аграрних даних

Традиційні системи моніторингу часто покладалися на стаціонарні промислові логічні контролери (ПЛК) та SCADA-системи. Однак, розвиток Інтернету речей (IoT) та поява потужних, компактних і доступних одноплатних комп'ютерів (Single-Board Computers, SBC) кардинально змінили підхід до проектування таких систем, особливо в аграрному секторі. Платформи, такі як Raspberry Pi, стали домінуючим вибором для мобільних та автономних рішень.

Мобільна IoT-платформа в контексті даної роботи — це автономний, легко переміщуваний програмно-апаратний комплекс, здатний виконувати весь цикл збору, обробки та зберігання даних локально, з опціональною можливістю передачі даних у хмару.

Основні переваги та особливості таких платформ для аграрних задач:

- **Низька вартість та доступність.** У порівнянні з промисловими контролерами, вартість Raspberry Pi та супутніх сенсорів (BME280, BH1750) є значно нижчою. Це дозволяє впроваджувати комплексний моніторинг навіть у малих господарствах з обмеженим бюджетом.
- **Низьке енергоспоживання.** Raspberry Pi споживає значно менше електроенергії, ніж повноцінний сервер чи ПК. Це критично важливо для об'єктів, де є обмеження по потужності або де система має працювати автономно від акумулятора чи сонячної панелі.
- **Автономність та локальна обробка (Edge Computing).** Це ключова особливість. Платформа не просто збирає дані, а має достатньо обчислювальної потужності, щоб запускати повноцінну операційну систему (Linux), виконувати скрипти (Python), запускати сервери баз даних (InfluxDB) та веб-сервери (Grafana). Це дозволяє системі:
    - **Працювати без стабільного Інтернету:** Дані збираються, зберігаються та візуалізуються локально. Втрата зв'язку з мережею не призводить до втрати даних або зупинки моніторингу.
    - **Зменшити затримки:** Дані обробляються на місці, що важливо для систем, які потребують швидкої реакції.
- **Гнучкість та масштабованість.** Завдяки наявності портів GPIO (General-Purpose Input/Output), до Raspberry Pi можна напряму підключати широкий спектр цифрових та аналогових сенсорів. Відкрите програмне забезпечення (ОС Raspberry Pi OS, мова Python) дає змогу швидко адаптувати систему під будь-які нові задачі чи датчики.
- **Компактність (мобільність).** Малі розміри дозволяють розмістити всю систему у невеликому захищеному корпусі безпосередньо в теплиці, мінімізуючи довжину кабелів до сенсорів.

1.4. Системи обробки даних

Системи обробки даних (СОД) є ядром будь-якої інформаційної системи. Їхнє призначення — перетворення "сирих" первинних даних, отриманих з джерел (у нашому випадку, з сенсорів), у структуровану, корисну та придатну для аналізу інформацію. В контексті моніторингу, СОД виконує роль "мозку" системи, що приймає потоки вимірювань та організовує їх для подальшого зберігання, візуалізації та аналізу.

1.4.1. Основні функції COD

Незалежно від масштабу, від мобільної платформи до великого дата-центру, СОД виконує набір стандартних функцій:

- **Збір даних (Collection):** Агрегація даних з одного чи багатьох джерел. У нашому проекті це реалізовано через Python-скрипт, який опитує сенсори BME280 та BH1750, підключені до GPIO.
- **Валідація та очищення (Validation & Cleaning):** Перевірка даних на коректність та аномалії. Сюди може входити відкидання очевидно помилкових значень (наприклад, температура -100°C або вологість 150%) та інтерполяція пропущених значень, хоча в простих системах моніторингу цей етап часто спрощують.
- **Агрегація (Aggregation):** Усереднення або групування даних для зменшення їх об'єму та підвищення інформативності. Наприклад, замість збереження 60 значень за хвилину, система може зберігати одне усереднене хвилинне значення.
- **Трансформація (Transformation):** Перетворення даних у потрібний формат. Це може включати переведення одиниць виміру (наприклад, з Фаренгейта в Цельсій) або приведення даних до єдиної структури для запису в базу даних.

- **Збереження (Storage):** Запис оброблених даних у систему довготривалого зберігання (базу даних).
- **Вибірка (Retrieval):** Надання доступу до збережених даних за запитом (наприклад, за запитом від системи візуалізації Grafana).

1.4.2. Компоненти COD

Система обробки даних складається з кількох логічних компонентів, які реалізують вищезазначені функції:

- **Апаратне забезпечення (Hardware):** Обчислювальні потужності, на яких виконується обробка. У нашому випадку це центральний процесор (CPU) та оперативна пам'ять (RAM) одноплатного комп'ютера Raspberry Pi.
- **Програмне забезпечення (Software):**
    - **Системне ПЗ:** Операційна система (наприклад, Raspberry Pi OS), що керує апаратними ресурсами.
    - **Прикладне ПЗ:** Скрипти та програми, що виконують логіку обробки. У даній роботі це скрипти на мові Python, які зчитують дані з сенсорів, та система управління базами даних InfluxDB, яка приймає, індексує та зберігає ці дані.
- **Інтерфейси даних (Data Interfaces):** Механізми, через які компоненти обмінюються даними. У нашій архітектурі це:
    - **I2C/GPIO:** Інтерфейс між сенсорами та скриптом Python.
    - **InfluxDB API:** Програмний інтерфейс, через який скрипт Python записує дані в базу даних InfluxDB.

У запропонованій системі СОД максимально локалізована: всі компоненти обробки (скрипт Python, база даних InfluxDB) працюють на одній машині (Raspberry Pi), що відповідає концепції "Edge Computing" (обчислень на периферії), обговореній у попередньому пункті.

1.5. Системи зберігання та резервного копіювання даних

Накопичення даних є лише першим кроком; забезпечення їх цілісності, доступності та довготривалого збереження є не менш важливою задачею. Системи зберігання та резервного копіювання є фундаментом надійності будь-якої інформаційної системи, особливо тієї, що працює в автономному режимі. Втрата історичних даних про мікроклімат може звести нанівець усі переваги моніторингу, оскільки унеможливлює аналіз сезонних трендів та порівняння ефективності різних агротехнічних підходів.

1.5.1. Огляд баз даних часових рядів (Time-Series Databases)

Дані, що збираються з сенсорів мікроклімату, за своєю природою є часовими рядами (time-series) — послідовністю точок даних, індексованих у хронологічному порядку. Кожна точка даних складається з мітки часу (timestamp) та одного або кількох значень (наприклад, температура, вологість).

Хоча для зберігання таких даних можна використовувати традиційні реляційні бази даних (як-от MySQL чи PostgreSQL), вони є неефективними для цього завдання. Операції запису та вибірки великих об'ємів даних за часовими діапазонами стають повільними та ресурсоємними.

Для вирішення цієї проблеми були розроблені спеціалізовані бази даних часових рядів (Time-Series Databases, TSDB). Їхня архітектура оптимізована саме для роботи з такими даними. Ключові переваги TSDB:

- **Висока швидкість запису:** TSDB розроблені для обробки мільйонів точок даних на секунду, що надходять від великої кількості сенсорів.
- **Ефективне стиснення даних:** Алгоритми стиснення в TSDB враховують природу часових рядів (де значення часто змінюються незначно), що дозволяє економити дисковий простір.

- **Швидкі запити за часом:** Запити на кшталт "показати середню температуру за останній тиждень з інтервалом в одну годину" виконуються значно швидше, ніж у реляційних БД.
- **Вбудовані функції для роботи з часом:** Багато TSDB мають вбудовані функції для агрегації, усереднення, вибірки даних за періодами (downsampling) та інші операції.

Одним із провідних представників TSDB з відкритим кодом є InfluxDB, яка була обрана для даного проекту. Вона ідеально підходить для IoT-задач, легко інтегрується з системою візуалізації Grafana та має клієнтські бібліотеки для мови Python, що спрощує розробку скриптів збору даних.

1.5.2. Методи та схеми резервного копіювання даних

Резервне копіювання (backup) — це процес створення копій даних, які можуть бути використані для їх відновлення у випадку втрати оригіналу. Існує кілька основних методів створення резервних копій:

- **Повне резервне копіювання (Full Backup):** Створюється повна копія всіх даних. Цей метод найнадійніший для відновлення, але вимагає найбільше часу та дискового простору.
- **Інкрементне резервне копіювання (Incremental Backup):** Копіюються лише ті дані, що змінилися з моменту *останнього будь-якого* резервного копіювання (повного або інкрементного). Це найшвидший метод, що економить місце, але відновлення є складнішим, оскільки вимагає повного бекапу та всієї послідовності інкрементних.
- **Диференціальне резервне копіювання (Differential Backup):** Копіюються дані, що змінилися з моменту *останнього повного* резервного копіювання. Займає більше місця, ніж інкрементне, але для відновлення потрібен лише повний бекап та останній диференціальний.

Для автоматизації процесу та ефективного управління архівами використовуються схеми ротації резервних копій. Одна з найпопулярніших і найнадійніших схем — "Дід-Батько-Син" (Grandfather-Father-Son, GFS). Ця схема передбачає створення та зберігання резервних копій з різною періодичністю:

- **"Син"** — щоденні резервні копії (наприклад, інкрементні).
- **"Батько"** — щотижневі резервні копії (зазвичай повні).
- **"Дід"** — щомісячні резервні копії (повні), що зберігаються тривалий час.

У даній дипломній роботі реалізовано модифікацію цієї схеми, а саме схему 7-4-4:

- **7 щоденних копій:** Зберігаються дані за останній тиждень.
- **4 щотижневі копії:** Зберігаються знімки даних на кінець кожного з останніх чотирьох тижнів.
- **4 щомісячні копії:** Зберігаються знімки даних на кінець кожного з останніх чотирьох місяців.

Такий підхід забезпечує баланс між гранулярністю відновлення (можна відновити дані на будь-який день протягом останнього тижня) та глибиною архіву, не вимагаючи при цьому надмірного дискового простору.

1.5.3. Організація захищеного мобільного доступу до даних

Надання віддаленого доступу до системи моніторингу є важливою функціональною вимогою. Агроном повинен мати можливість переглядати дашборди Grafana зі смартфона чи комп'ютера, перебуваючи поза межами теплиці.

Традиційний підхід до вирішення цієї задачі полягає у "прокиданні портів" (port forwarding) на маршрутизаторі. Цей метод має суттєві недоліки з точки зору безпеки:

- **Відкриття системи для атак:** Відкритий порт робить внутрішній сервіс (Grafana) видимим для всього Інтернету, що робить його ціллю для автоматизованих сканерів вразливостей та зловмисників.
- **Вимога наявності статичної IP-адреси:** Більшість інтернет-провайдерів надають динамічні IP-адреси, що ускладнює налаштування постійного доступу.
- **Складність налаштування SSL/TLS:** Забезпечення шифрованого HTTPS-з'єднання вимагає отримання та налаштування сертифікатів, що може бути нетривіальною задачею.

Сучасним та значно безпечнішим підходом є використання технологій "нульової довіри" (Zero Trust Network Access, ZTNA). Концепція "нульової довіри" полягає в тому, що система за замовчуванням не довіряє нікому, навіть у внутрішній мережі, і вимагає автентифікації для кожного запиту доступу.

У цій роботі для реалізації захищеного доступу використовується сервіс Cloudflare Zero Trust та його компонент Cloudflare Tunnel. Принцип його роботи наступний:

1. На Raspberry Pi запускається легковаговий програмний агент (connector), який встановлює вихідне зашифроване з'єднання (тунель) з найближчим дата-центром Cloudflare.
2. При цьому жодних вхідних портів на маршрутизаторі відкривати не потрібно. Система залишається повністю невидимою ззовні.
3. Коли користувач намагається отримати доступ до публічної адреси, пов'язаної з тунелем, Cloudflare перевіряє його права доступу (наприклад, вимагає логін та пароль або вхід через Google-акаунт).
4. Лише після успішної автентифікації Cloudflare проксіює запит користувача через захищений тунель до сервісу Grafana на Raspberry Pi.

Такий підхід повністю приховує систему від зовнішньої мережі, забезпечує шифрування трафіку та дозволяє гнучко налаштовувати політики доступу, що є

набагато безпечнішим та надійнішим рішенням порівняно з традиційним прокиданням портів.

1.6. Висновки до розділу 1

У даному розділі було проведено комплексний аналіз задачі моніторингу та резервного зберігання даних в аграрному секторі, що дозволило сформувати теоретичне підґрунтя для розробки системи.

За результатами проведеного аналізу:

- Визначено, що система моніторингу мікроклімату в тепличному господарстві виконує чотири ключові функціональні задачі: безперервний збір даних, їх аналіз та візуалізація, підтримка прийняття рішень та, що критично важливо, надійне зберігання та архівування даних.
- Досліджено, що мобільні IoT-платформи на базі одноплатних комп'ютерів, зокрема Raspberry Pi, є оптимальним апаратним рішенням. Вони поєднують низьку вартість та енергоспоживання з достатньою обчислювальною потужністю для реалізації локальної обробки (Edge Computing), що забезпечує автономність системи та її незалежність від стабільного інтернет-з'єднання.
- Проаналізовано компоненти системи обробки даних та встановлено, що архітектура, де Python-скрипти напряму збирають дані з сенсорів та передають їх у локальну базу, є простою та надійною для реалізації на Raspberry Pi.
- Встановлено, що для зберігання даних моніторингу (часових рядів) необхідно використовувати спеціалізовані бази даних (TSDB). Обрана InfluxDB є галузевим стандартом, що оптимізований для швидкого запису та агрегації таких даних, а також нативно інтегрується з візуалізатором Grafana.

- Досліджено методи резервного копіювання та обґрунтовано вибір гібридної схеми ротації "7-4-4", яка є варіацією "Дід-Батько-Син" (GFS). Ця схема забезпечує оптимальний баланс між глибиною архіву та використанням дискового простору.
- Виявлено, що традиційні методи надання віддаленого доступу (через "прокидання портів") є небезпечними. Натомість, застосування технології Cloudflare Zero Trust (Tunnel) дозволяє організувати повністю захищений мобільний доступ до дашбордів Grafana, не відкриваючи жодних портів у локальній мережі та не вимагаючи статичної IP-адреси.

Таким чином, проведений у першому розділі аналіз дозволив ідентифікувати ключові проблеми, вимоги та технології у предметній області. Було обґрунтовано вибір конкретного технологічного стеку (Raspberry Pi, BME280/BH1750, Python, InfluxDB, Grafana, Cloudflare), який буде використано для практичної розробки та дослідження системи у наступних розділах дипломної роботи.

2 ФОРМУВАННЯ ЗАГАЛЬНОГО ПРЕДСТАВЛЕННЯ СИСТЕМИ

У цьому розділі на основі теоретичного аналізу, проведеного в попередньому розділі, формується загальне представлення розроблюваної системи. Буде детально описано її структуру та функціональну схему, обґрунтовано вибір конкретних технічних та програмних засобів, а також розглянуто принципи роботи з ключовими технологіями, що лежать в основі проекту.

2.1. Структура системи збору та резервного зберігання даних

Розроблювана система має багаторівневу архітектуру, яка логічно розділяє її на апаратний, програмний та рівень доступу. Такий підхід забезпечує гнучкість, масштабованість та спрощує розробку й обслуговування. Структурну схему системи наведено на рис. 2.1.

2.1.1. Основні структурні компоненти системи

- **Рівень збору даних (Hardware Layer):** Це фізична основа системи, що безпосередньо взаємодіє з навколишнім середовищем.
    - **Сенсорний вузол:** Включає датчик температури, вологості та тиску BME280 та датчик освітленості BH1750. Ці пристрої відповідають за перетворення фізичних параметрів мікроклімату в цифрові сигнали.
    - **Обчислювальний модуль:** Центральним елементом є одноплатний комп'ютер Raspberry Pi, який виконує роль концентратора даних, обробника та сервера.
    - **Зовнішній носій:** USB-накопичувач або зовнішній жорсткий диск, призначений для зберігання архівних копій бази даних.
- **Програмний рівень (Software Layer):** Це набір програмних компонентів, розгорнутих на Raspberry Pi, що забезпечують функціонування системи.
    - **Операційна система:** Raspberry Pi OS (раніше Raspbian), дистрибутив Linux на базі Debian, що надає стабільне середовище для роботи всіх сервісів.
    - **Скрипт збору даних:** Програма, написана мовою Python, яка періодично опитує сенсори через інтерфейс I2C, форматує отримані дані та записує їх у базу даних.
    - **Система зберігання:** База даних часових рядів InfluxDB, що відповідає за ефективне зберігання, індексацію та агрегацію даних з сенсорів.
    - **Система візуалізації:** Сервер візуалізації Grafana, який підключається до InfluxDB, отримує з неї дані та відображає їх у вигляді інтерактивних графіків та дашбордів.

- **Рівень доступу (Access Layer):** Компоненти, що забезпечують захищену взаємодію користувача з системою.
    - **Веб-інтерфейс Grafana:** Надає користувачеві (агроному) доступ до візуалізованих даних через веб-браузер.
    - **Захищений тунель:** Сервіс Cloudflare Tunnel, який створює безпечний вихідний канал від Raspberry Pi до мережі Cloudflare, дозволяючи отримати доступ до Grafana з будь-якої точки світу без відкриття вхідних портів.

Дані в системі рухаються за наступним маршрутом: сенсори BME280/BH1750 -> Raspberry Pi (Python-скрипт) -> база даних InfluxDB -> сервер Grafana -> Cloudflare Tunnel -> пристрій користувача. Паралельно, за розкладом, виконується резервне копіювання даних з InfluxDB на зовнішній носій.

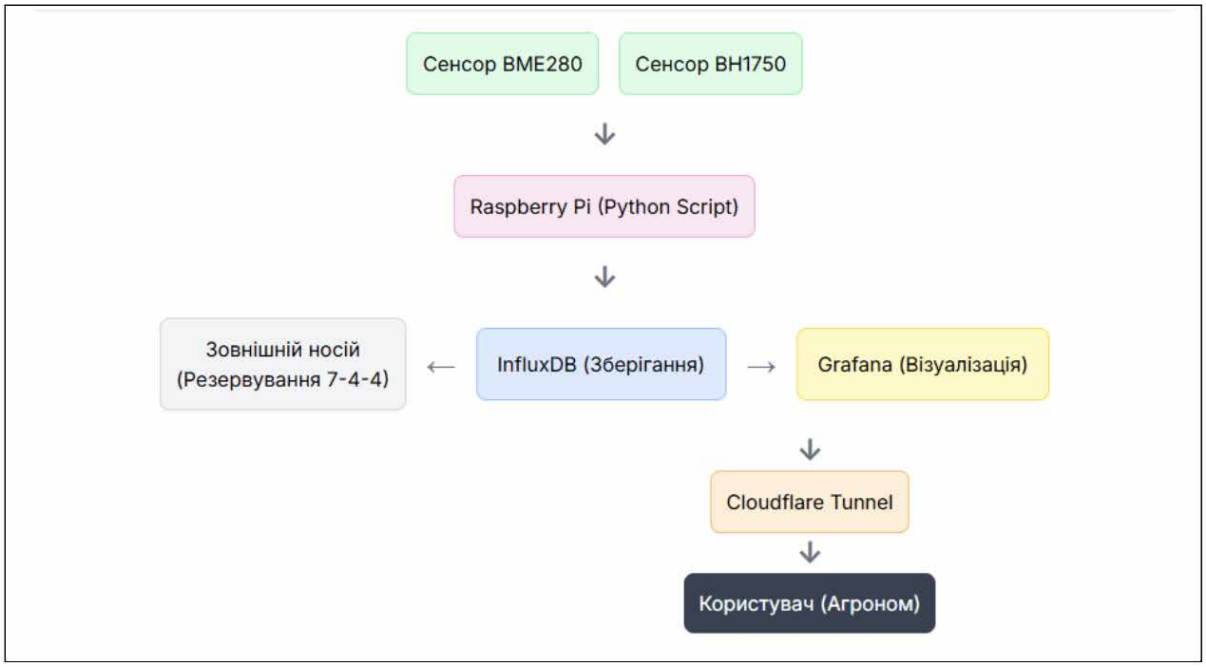

Рис. 2.1. Структурна схема системи моніторингу та резервного зберігання даних

2.2. Функціональна схема системи на базі Raspberry Pi

Функціональна схема деталізує взаємодію між апаратними та програмними компонентами системи, описуючи логічні потоки даних та керуючі процеси. Вона є розвитком структурної схеми, показаної у попередньому пункті, і фокусується на конкретних потоках інформації. Функціональну схему розробленої системи зображено на рис. 2.2.

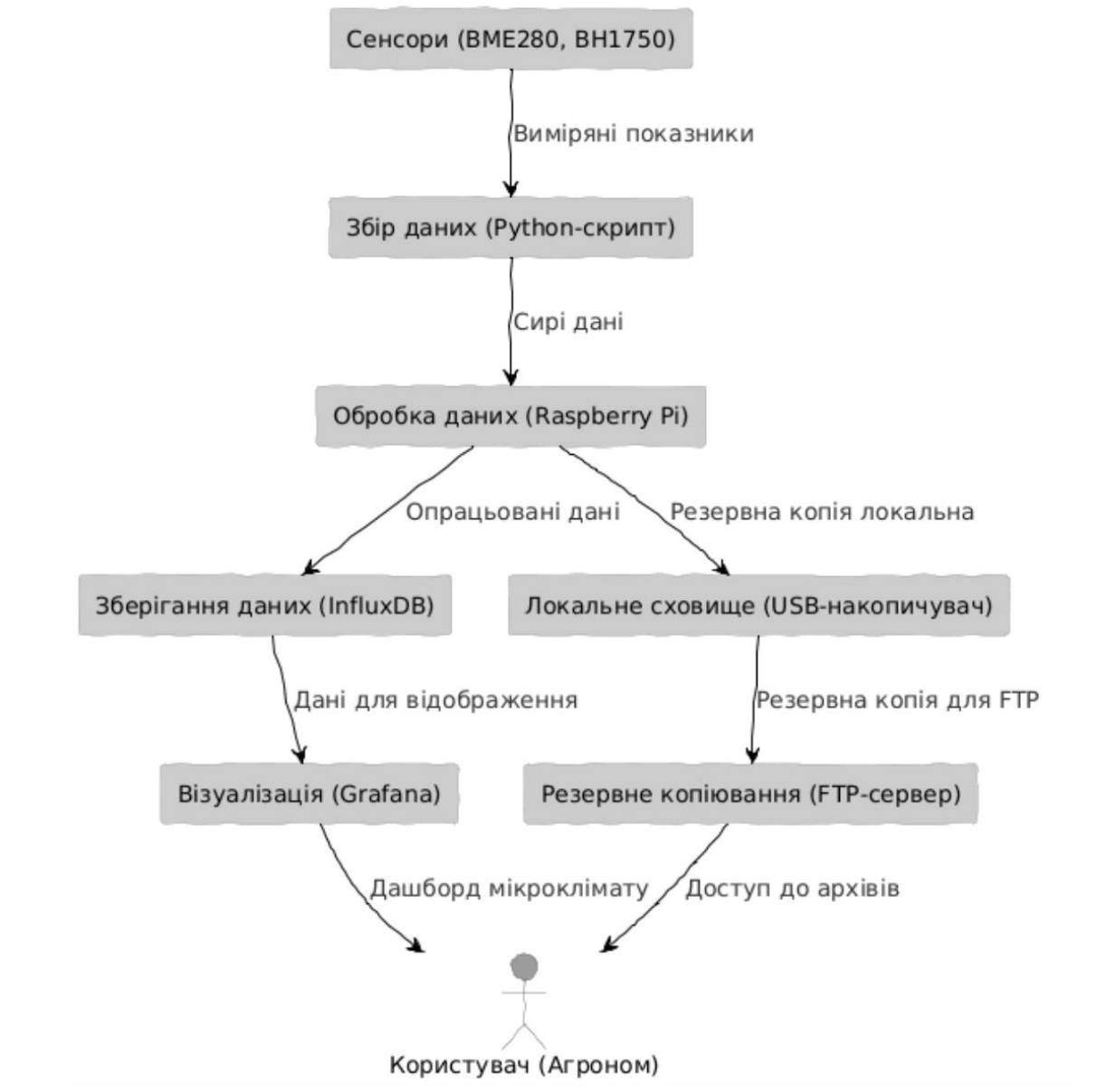

Рис. 2.2. Функціональна схема системи на базі Raspberry Pi

Опис функціональних потоків:

- **Потік збору даних (Data Acquisition Flow):**
    - Процес ініціюється Python-скриптом, що виконується на Raspberry Pi за розкладом (наприклад, кожну хвилину).
    - Скрипт звертається до апаратної шини I2C, до якої підключені сенсори BME280 (температура, вологість, тиск) та BH1750 (освітленість).
    - Сенсори повертають скрипту "сирі" цифрові дані.
    - Скрипт проводить первинну обробку (трансформацію) даних у потрібні одиниці виміру (наприклад, °C, %, гПа, люкс) та формує точку даних (point) у форматі, сумісному з InfluxDB.
- **Потік зберігання даних (Storage Flow):**
    - Python-скрипт, використовуючи клієнтську бібліотеку, встановлює з'єднання з локальним сервером InfluxDB, що працює на тій же Raspberry Pi.
    - Скрипт надсилає сформовану точку даних до відповідної бази (bucket) в InfluxDB.
    - InfluxDB приймає дані, додає до них точну часову мітку (timestamp) та індексує їх для швидкого доступу. Цей процес повторюється циклічно, формуючи часові ряди.
- **Потік візуалізації (Visualization Flow):**
    - Користувач (агроном) через веб-браузер відкриває дашборд (панель приладів) у Grafana.
    - Сервер Grafana, відповідно до налаштувань панелей, надсилає запити (мовою Flux або InfluxQL) до бази даних InfluxDB для отримання даних за певний проміжок часу.
    - InfluxDB виконує запит, агрегує дані (наприклад, усереднює по 5-хвилинним інтервалам) і повертає результат.
    - Grafana "на льоту" рендерить (малює) отримані дані у вигляді графіків та індикаторів, які бачить користувач.
- **Потік резервування (Backup Flow):**

    - Незалежний процес, що запускається системним планувальником (cron) за розкладом (наприклад, щоночі).
    - Спеціальний скрипт (bash-скрипт) викликає вбудовану утиліту influxd backup для створення повного "знімка" (snapshot) бази даних InfluxDB.
    - Отриманий архів пакується (наприклад, у .tar.gz) та копіюється на зовнішній носій (USB-накопичувач), дотримуючись схеми ротації 7-4-4.
- **Потік захищеного доступу (Secure Access Flow):**
    - Сервіс Cloudflare Tunnel, запущений на Raspberry Pi, підтримує постійне вихідне зашифроване з'єднання з мережею Cloudflare.
    - Коли користувач намагається отримати доступ до публічної URL-адреси системи, Cloudflare спершу вимагає автентифікації (Zero Trust).
    - Після успішної перевірки, Cloudflare проксіює HTTPS-запит користувача через захищений тунель безпосередньо до локального сервера Grafana на Raspberry Pi. При цьому сама Raspberry Pi залишається повністю прихованою від Інтернету.

Ця схема демонструє, як Raspberry Pi виступає в ролі центрального вузла, що одночасно керує збором, обробкою, зберіганням, резервуванням та захищеною публікацією даних.

2.3. Технічні засоби реалізації мобільної платформи

Вибір апаратного забезпечення є фундаментальним етапом проектування, оскільки він визначає вартість, надійність, енергоефективність та функціональні можливості майбутньої системи. Для даної роботи, з огляду на вимоги мобільності, автономності та низької вартості, було обрано комбінацію одноплатного комп'ютера Raspberry Pi та набору цифрових сенсорів.

2.3.1. Мікроконтролер Raspberry Pi та його характеристики

Центральним обчислювальним вузлом системи обрано одноплатний комп'ютер Raspberry Pi. Важливо зазначити, що Raspberry Pi, хоч і часто називається мікроконтролером, насправді є повноцінним комп'ютером (Single-Board Computer, SBC) на базі архітектури ARM, здатним працювати під управлінням повноцінних операційних систем, таких як Linux. Сам мікроконтролер Raspberry Pi зображений на рисунку 2.3.

Вибір цієї платформи обумовлений низкою ключових характеристик:

- **Висока обчислювальна потужність (у порівнянні з мікроконтролерами):** Сучасні моделі Raspberry Pi (наприклад, Pi 3, Pi 4) оснащені багатоядерними 64-бітними процесорами та достатнім об'ємом оперативної пам'яті (від 1 ГБ і вище). Це дозволяє без проблем запускати одночасно декілька сервісів: Python-скрипти, базу даних InfluxDB та веб-сервер Grafana.
- **Інтерфейс GPIO:** Наявність 40-контактної колодки GPIO (General-Purpose Input/Output) дозволяє напряму підключати низькорівневу периферію, таку як сенсори, без необхідності у додаткових контролерах.
- **Мережеві можливості:** Вбудовані модулі Wi-Fi та Ethernet забезпечують легке підключення до локальної мережі та Інтернету, що є критично важливим для організації віддаленого доступу.
- **Низьке енергоспоживання:** У порівнянні з повноцінним ПК або сервером, Raspberry Pi споживає мінімум енергії (зазвичай 5-15 Вт), що ідеально підходить для безперервної роботи 24/7.
- **Велика спільнота та програмна підтримка:** Наявність повноцінної ОС Raspberry Pi OS (на базі Debian) та величезна кількість документації і готових бібліотек (особливо для Python) значно спрощують процес розробки та налаштування.

У даній роботі Raspberry Pi виконує роль "все-в-одному": він опитує сенсори, обробляє дані, зберігає їх у базі даних та обслуговує веб-інтерфейс візуалізації.

Рис. 2.3. Мікроконтролер Raspberry Pi

2.3.2. Сенсори мікроклімату (BME280, BH1750)

Для збору даних про мікроклімат було обрано два цифрових сенсори, які добре зарекомендували себе у DIY та IoT-проектах завдяки своїй точності, низькій вартості та простоті підключення через інтерфейс I2C.

- **Bosch BME280 (Температура, вологість, тиск):** Це високоточний комбінований датчик від Bosch Sensortec. Він здатний одночасно вимірювати три ключові параметри:
    - Температуру (з точністю близько ±1.0°C)
    - Відносну вологість (з точністю ±3%)

- Атмосферний тиск (з високою абсолютною точністю) Його використання дозволяє отримати комплексну картину стану повітря в теплиці, використовуючи лише один компонент.

- **Rohm BH1750 (Освітленість):** Це цифровий датчик освітленості, який вимірює рівень світла та повертає значення безпосередньо в люксах (lx). Це значно зручніше, ніж робота з аналоговими фоторезисторами, які вимагають калібрування. Датчик має широкий діапазон вимірювання та високу роздільну здатність, що дозволяє точно фіксувати як тьмяне освітлення, так і прямі сонячні промені.

Обидва сенсори працюють за шиною I2C (Inter-Integrated Circuit). Це двопровідний інтерфейс (SDA - дані, SCL - тактовий сигнал), який дозволяє підключати до Raspberry Pi безліч пристроїв на одну й ту ж саму пару пінів, що значно спрощує схему підключення. Самі сенсори зображені на рис 2.4.

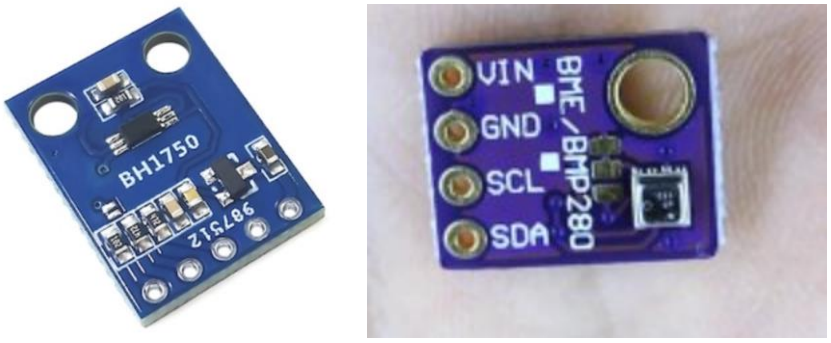

Рис. 2.4. Сенсори BME280 та BH1750

2.3.3. Вибір зовнішніх носіїв для резервного копіювання

Головним носієм інформації в Raspberry Pi є microSD-карта, на якій встановлена операційна система та працює база даних InfluxDB. Однак, microSD-карти мають обмежений ресурс циклів перезапису і не призначені для інтенсивних операцій з базами даних, а також схильні до раптових збоїв. Зберігання резервних копій на тому ж носії, де зберігаються оригінальні дані, є неприпустимим, оскільки збій носія призведе до одночасної втрати і даних, і їхніх копій.

Тому для організації надійної системи було реалізовано багаторівневу (гібридну) стратегію резервного копіювання, що включає два незалежних компоненти:

- **Локальний носій (USB Flash Drive):** Це носій, фізично підключений до Raspberry Pi, призначений для швидкого оперативного відновлення (operational recovery). Якщо з ладу вийде лише microSD-карта (найпоширеніший випадок), можна швидко замінити її, підключити USB-накопичувач та відновити дані, мінімізувавши час простою.

При виборі локального носія розглядалися два варіанти:

- **USB Flash Drive (Флеш-накопичувач):**
  - *Переваги:* Низька вартість, компактність, низьке енергоспоживання, стійкість до вібрацій.
  - *Недоліки:* Обмежений ресурс перезапису (хоча для запису 1 раз на добу це некритично).
- **Зовнішній жорсткий диск (External HDD/SSD):**
  - *Переваги:* Значно більший об'єм, вища надійність (особливо SSD).
  - *Недоліки:* Вища вартість, більше енергоспоживання, чутливість HDD до механічних впливів.

Для нашої системи, враховуючи невеликий об'єм щоденних архівів, USB флеш-накопичувач є оптимальним вибором для локального архіву.

- **Віддалений носій (FTP-сервер):** Локальний бекап (навіть на USB-носії) не захищає від катастрофічних подій: пожежі, крадіжки, виходу з ладу самого Raspberry Pi або стрибка напруги, який може знищити всі підключені пристрої.

Для аварійного відновлення (disaster recovery) необхідна копія "поза межами об'єкта" (off-site). Це відповідає золотому правилу резервного копіювання "3-2-1" (3 копії даних, 2 різні носії, 1 копія — віддалена).

Для реалізації цього рівня захисту в системі використовується автоматичне вивантаження створеного локального архіву на віддалений FTP-сервер.

Обрана гібридна стратегія (локальна копія на USB-носії + віддалена копія на FTP-сервері) забезпечує максимальний рівень надійності, захищаючи дані як від поширених операційних збоїв (вихід з ладу SD-карти), так і від катастрофічних (втрата всього обладнання).

2.4. Організація прямого збору даних з сенсорів

Після вибору апаратних компонентів необхідно визначити метод збору даних. У даній роботі реалізовано прямий збір даних, що означає, що програмне забезпечення, яке виконується на Raspberry Pi, безпосередньо взаємодіє з сенсорами, підключеними до його фізичних інтерфейсів (GPIO/I2C), без використання проміжних ланок, таких як мікроконтролери (наприклад, Arduino) або протоколи повідомлень (наприклад, MQTT).

2.4.1. Переваги прямого збору даних:

- **Простота архітектури:** Відсутність MQTT-брокера або іншого посередника зменшує кількість "точок відмови" та спрощує налаштування і обслуговування системи.
- **Низька затримка:** Дані зчитуються та негайно обробляються, оскільки немає мережевих затримок, пов'язаних з публікацією/підпискою на повідомлення.
- **Економія ресурсів:** Немає необхідності запускати додатковий сервіс (MQTT-брокер), що економить оперативну пам'ять та процесорний час Raspberry Pi.

2.4.2. Реалізація механізму збору:

- **Протокол взаємодії (I2C):** Як було зазначено в п. 2.3.2, обидва сенсори (BME280 та BH1750) підтримують цифровий інтерфейс I2C. Це ідеальний вибір для даного проекту, оскільки I2C дозволяє підключати декілька пристроїв до однієї двопровідної шини (SDA/SCL), використовуючи унікальні адреси для кожного пристрою. Raspberry Pi OS має вбудовану підтримку I2C, яку необхідно активувати через конфігурацію (raspi-config).
- **Програмне забезпечення (Python):** Для зчитування даних було обрано мову програмування Python. Вона є стандартом "де-факто" для розробки на Raspberry Pi завдяки своїй простоті, читабельності та наявності величезної кількості готових бібліотек для роботи з апаратним забезпеченням.
- **Використовувані бібліотеки:**
    - smbus2: Це Python-бібліотека, яка надає низькорівневий доступ до шини I2C в Linux.
    - **Спеціалізовані бібліотеки сенсорів:** Замість прямої роботи з регістрами I2C, що є складним процесом, були використані готові бібліотеки (наприклад, adafruit-circuitpython-bme280 та adafruit-circuitpython-bh1750). Ці бібліотеки інкапсулюють всю складну логіку взаємодії та надають простий інтерфейс для отримання вже оброблених даних (наприклад, sensor.temperature повертає температуру у градусах Цельсія).
- **Автоматизація (Cron):** Для забезпечення періодичного збору даних (виконання функціональної задачі моніторингу) використовується стандартний планувальник завдань Linux — cron. У файл crontab додається запис, який вказує системі запускати Python-скрипт збору даних через визначений інтервал (наприклад, кожні 5 хвилин). Це

забезпечує надійний та регулярний потік даних до бази даних InfluxDB без необхідності постійної роботи скрипту у фоні.

Таким чином, процес виглядає наступним чином: cron запускає python_script.py -> скрипт імпортує бібліотеки сенсорів -> бібліотеки через smbus2 звертаються до шини I2C -> сенсори повертають дані -> скрипт відправляє дані в InfluxDB (що буде розглянуто далі).

2.5. Засоби розробки та візуалізації програмного забезпечення

Для перетворення апаратної платформи на повноцінну систему моніторингу необхідний ретельно підібраний набір програмного забезпечення (програмний стек). Вибрані інструменти повинні бути сумісними з архітектурою Raspberry Pi (ARM), ефективно використовувати обмежені ресурси та надавати всі необхідні функції: від збору даних до їх візуалізації та захищеного доступу.

У даній роботі було обрано стек технологій, що базується на компонентах з відкритим вихідним кодом: Raspberry Pi OS, мова програмування Python, база даних InfluxDB та платформа візуалізації Grafana. Для забезпечення захищеного доступу цей стек доповнено сервісом Cloudflare Zero Trust.

2.5.1. Програмне середовище Raspberry Pi (ОС, Python)

- **Операційна система (ОС): Raspberry Pi OS** Офіційна операційна система для Raspberry Pi (раніше відома як Raspbian) є дистрибутивом Linux, що базується на Debian. Цей вибір є оптимальним, оскільки ОС повністю адаптована під апаратне забезпечення Raspberry Pi, включає всі необхідні драйвери та утиліти (включаючи raspi-config для легкого налаштування інтерфейсів I2C, SPI тощо). Як повноцінна багатозадачна ОС, вона дозволяє одночасно запускати декілька сервісів у фоновому режимі (демонів), таких як база даних, веб-сервер та планувальник завдань cron, що є критично важливим для автономної роботи системи.

- **Мова розробки: Python** Python було обрано як основну мову для розробки скриптів збору даних. Це рішення обумовлене низкою факторів:

    - **Простота та читабельність:** Дозволяє швидко писати та підтримувати код.
    - **Величезна екосистема:** Існує велика кількість готових бібліотек для роботи з будь-яким апаратним забезпеченням (включаючи adafruit-circuitpython-bme280, bh1750 для сенсорів) та сервісами (наприклад, influxdb-client для роботи з InfluxDB).
    - **Інтеграція з ОС:** Python встановлений за замовчуванням у Raspberry Pi OS та ідеально інтегрується з системними інструментами, такими як cron. У даному проекті Python виконує роль "клею", що пов'язує апаратний рівень (сенсори) з рівнем зберігання даних (InfluxDB).

2.5.2. Система зберігання InfluxDB та візуалізації Grafana

Для ефективної роботи з даними моніторингу (часовими рядами) було обрано зв'язку InfluxDB + Grafana, яка є де-факто стандартом для завдань IoT та моніторингу систем.

InfluxDB (Система зберігання) Як було обґрунтовано в розділі 1, традиційні реляційні БД погано підходять для даних часових рядів. InfluxDB — це високопродуктивна база даних часових рядів (TSDB), розроблена спеціально для швидкого запису та вибірки даних з часовими мітками. Ключові переваги: Висока швидкість запису (ingestion), ефективне стиснення даних, вбудована мова запитів (Flux або InfluxQL), оптимізована для часових діапазонів, а також політики зберігання (retention policies), які дозволяють автоматично видаляти застарілі дані. Версія InfluxDB 2.x, що використовується в роботі, добре оптимізована для роботи на пристроях з архітектурою ARM, таких як Raspberry Pi.

Grafana (Система візуалізації) Grafana — це провідна платформа з відкритим вихідним кодом для аналітики та моніторингу, яка дозволяє візуалізувати дані з різноманітних джерел. Ключові переваги: Вона нативно (напряму) підтримує InfluxDB як джерело даних. Grafana надає потужний та інтуїтивно зрозумілий конструктор дашбордів (панелей приладів), що дозволяє легко створювати інтерактивні графіки, індикатори (gauges), таблиці та гістограми. Це дає змогу агроному отримати повне візуальне уявлення про стан мікроклімату в реальному часі. Grafana також має вбудовану систему сповіщень (alerting), яку можна налаштувати для відправки повідомлень при виході параметрів за норму.

2.5.3. Засоби тунелювання та захисту (Cloudflare Zero Trust)

Забезпечення захищеного віддаленого доступу до дашборду Grafana, що працює на Raspberry Pi у внутрішній мережі теплиці, є нетривіальною задачею. Традиційний метод "прокидання портів" (Port Forwarding) на роутері є вкрай небезпечним. Для вирішення цієї проблеми було обрано сервіс Cloudflare Zero Trust та його компонент Cloudflare Tunnel. Це сучасний підхід, що базується на принципі "нульової довіри" (Zero Trust). Принцип роботи: Замість відкриття вхідного порту, на Raspberry Pi запускається легкий сервіс (cloudflared), який встановлює лише вихідне зашифроване з'єднання (тунель) з найближчим дата-центром Cloudflare.

Переваги:

- **Безпека:** Система на Raspberry Pi залишається повністю "невидимою" для Інтернету. Жоден відкритий порт не сканується зловмисниками.
- **Простота:** Не потрібна статична IP-адреса. Система буде доступна за постійним доменним ім'ям, навіть якщо IP-адреса теплиці зміниться.
- **Автентифікація:** Cloudflare дозволяє "обгорнути" тунель додатковим рівнем автентифікації (наприклад, Email Auth, що вимагає підтвердження через електронну пошту ще до того, як користувач потрапить на сторінку входу

Grafana), або навіть використовувати вхід через Google/GitHub. У даній роботі це рішення забезпечує надійний та захищений мобільний доступ для агронома. Процес автентифікації Cloudflare зображений на рисунку 2.5.

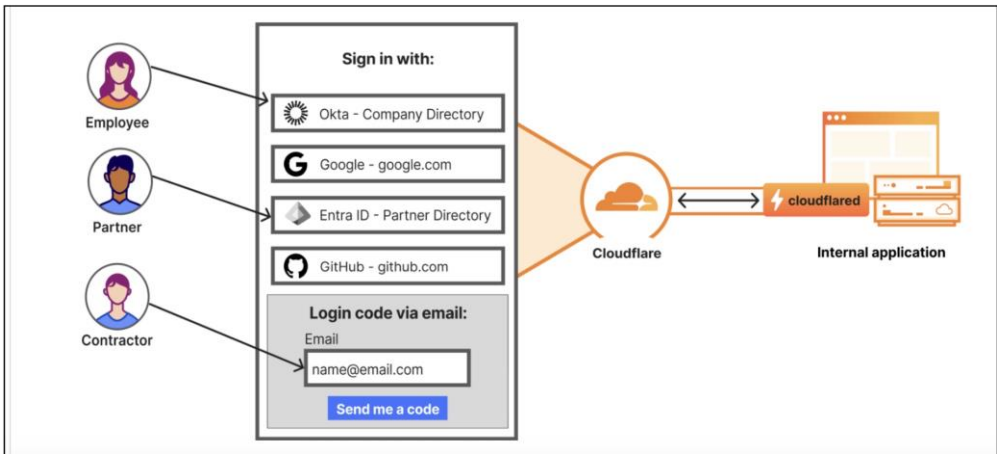

Рис 2.5. Процесс автентифікації від Cloudflare

3 РОЗРОБКА СИСТЕМИ

У цьому розділі детально описується практичний процес створення системи моніторингу та резервного копіювання: від фізичного підключення апаратних компонентів до розгортання та налаштування повного програмного стеку. Цей розділ переводить теоретичні обґрунтування з Розділу 2 у конкретні кроки реалізації.

3.1. Розробка схем підключення сенсорів до Raspberry Pi

Першим етапом практичної розробки є створення апаратної частини системи, а саме — фізичне підключення сенсорів мікроклімату до обчислювального модуля Raspberry Pi.

Як було обґрунтовано в п. 2.3.2, для підключення обох датчиків (BME280 та BH1750) було обрано цифровий інтерфейс I2C (Inter-Integrated Circuit). Це двопровідна шина, яка є ідеальним рішенням для даного проекту, оскільки

дозволяє підключати декілька пристроїв-сенсорів до одних і тих же виводів Raspberry Pi.

3.1.1. Використані виводи (піни) GPIO Raspberry Pi

Для роботи I2C на Raspberry Pi використовуються два спеціалізовані піни, а також піни для живлення:

- **Живлення (3.3V):** Використовується Pin 1 (3.3V DC Power) для живлення обох сенсорів. Важливо використовувати саме 3.3V, оскільки логічні рівні BME280 та BH1750 розраховані на цю напругу.
- **Земля (GND):** Використовується Pin 6 (Ground) як спільна земля для обох сенсорів.
- **Дані I2C (SDA):** Використовується Pin 3 (GPIO 2 / SDA). Це лінія даних шини I2C.
- **Тактовий сигнал I2C (SCL):** Використовується Pin 5 (GPIO 3 / SCL). Це лінія тактового сигналу, що синхронізує передачу даних.

3.1.2. Схема підключення

Обидва модулі сенсорів (BME280 та BH1750) підключаються паралельно до однієї шини I2C.

- Вивід VCC (або VIN) кожного сенсора підключається до Pin 1 (3.3V) Raspberry Pi.
- Вивід GND кожного сенсора підключається до Pin 6 (GND) Raspberry Pi.
- Вивід SDA кожного сенсора підключається до Pin 3 (SDA) Raspberry Pi.
- Вивід SCL кожного сенсора підключається до Pin 5 (SCL) Raspberry Pi.

Таке підключення є можливим, оскільки кожен пристрій на шині I2C має унікальну апаратну адресу. Для BME280 це зазвичай 0x76 (або 0x77), а для BH1750 — 0x23 (або 0x5C). Ці адреси не конфліктують, що дозволяє Raspberry Pi звертатися до кожного сенсора окремо.

Принципову електричну схему підключення наведено на рис. 3.1.

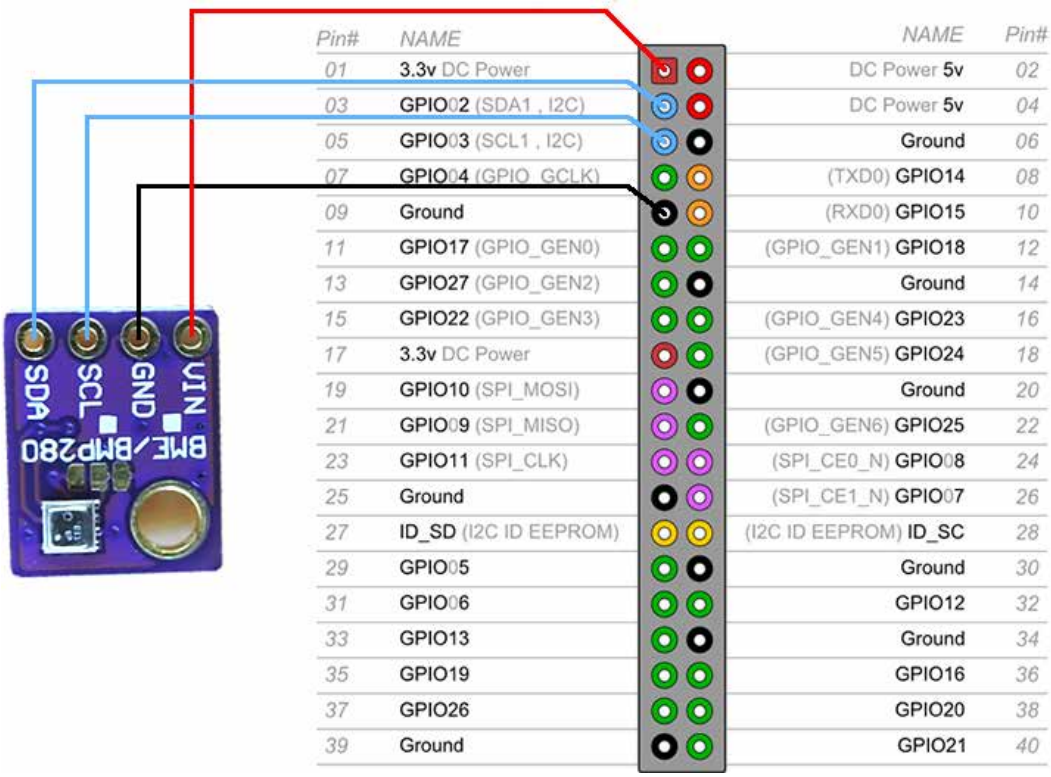

Рис. 3.1. Принципова схема підключення сенсорів BME280 та BH1750 до Raspberry Pi по шині I2C

Ця схема є простою, надійною та мінімізує кількість необхідних дротів, що є важливим для мобільної та компактної системи.

3.2. Налаштування роботи апаратних компонентів

Після фізичного підключення сенсорів (як описано в 3.1) наступним кроком є налаштування операційної системи Raspberry Pi OS для коректної взаємодії з цим обладнанням. За замовчуванням, апаратні інтерфейси, такі як I2C, вимкнені з міркувань безпеки та економії ресурсів.

3.2.1. Активація інтерфейсу I2C

Найпростіший та рекомендований спосіб увімкнути шину I2C — це використання вбудованої утиліти конфігурації raspi-config. Процес виконується через термінал Raspberry Pi:

- Запускається команда sudo raspi-config.
- У текстовому меню обирається пункт Interface Options (або Interfacing Options).
- У наступному меню обирається I2C.
- Система запитує, чи бажаєте ви увімкнути I2C. Необхідно обрати <Yes>.
- Після підтвердження утиліта автоматично вносить необхідні зміни у конфігураційні файли системи (зокрема, /boot/config.txt).
- Для того, щоб зміни набули чинності, необхідно перезавантажити Raspberry Pi.

3.2.2. Перевірка підключення сенсорів

Після перезавантаження система повинна "побачити" підключені до шини I2C пристрої. Для перевірки цього необхідно встановити пакет інструментів для роботи з I2C: sudo apt-get update sudo apt-get install i2c-tools

Після встановлення виконується команда i2cdetect: i2cdetect -y 1 . Якщо сенсори підключено коректно, у терміналі з'явиться таблиця з адресами виявлених пристроїв. Очікуваний результат має виглядати приблизно так:

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- 23 -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- 5c -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- 76 --
```

У даному виводі:

- 23 — це стандартна адреса сенсора освітленості BH1750.
- 76 — це стандартна адреса сенсора BME280. (Примітка: залежно від модуля, адреса BH1750 може бути 5c, а BME280 — 77).

Наявність цих адрес у виводі команди підтверджує, що апаратне підключення виконано правильно, і операційна система успішно ідентифікувала обидва сенсори.

3.2.3. Встановлення програмних бібліотек (Python)

Хоча ОС "бачить" пристрої, для взаємодії з ними з-під Python потрібні спеціалізовані бібліотеки (драйвери). Вони встановлюються за допомогою менеджера пакунків pip:

- **Низькорівнева бібліотека:** Встановлюється smbus2 для забезпечення доступу Python до системної шини I2C. pip install smbus2
- **Високорівневі бібліотеки сенсорів:** Встановлюються бібліотеки від Adafruit, які значно спрощують роботу з сенсорами, надаючи готові функції для зчитування даних у потрібних одиницях. pip install adafruit-circuitpython-bme280 pip install adafruit-circuitpython-bh1750

Після виконання цих трьох кроків апаратна частина системи повністю налаштована, інтерфейси активовані, а програмне середовище Python готове до написання скриптів для збору даних.

3.3. Розгортання програмного стеку: InfluxDB, Grafana

Після підготовки апаратної частини, наступний крок — встановлення та налаштування програмних сервісів, які відповідатимуть за зберігання та візуалізацію даних. Як було обґрунтовано в Розділі 2, для цих завдань було обрано стек InfluxDB (для зберігання) та Grafana (для візуалізації). Обидва сервіси встановлюються безпосередньо на Raspberry Pi.

3.3.1. Розгортання бази даних InfluxDB (v2.x)

InfluxDB є ядром нашої системи зберігання. Для її встановлення на Raspberry Pi OS (яка базується на Debian) використовується менеджер пакунків apt.

- **Додавання репозиторію InfluxData:** Спершу до системи додається офіційний репозиторій InfluxData, щоб apt міг знайти пакунки, скомпільовані для архітектури ARM.
- **Встановлення:** Виконується команда sudo apt-get install influxdb2.
- **Первинне налаштування:** Після встановлення, InfluxDB v2.x вимагає первинного налаштування. Це виконується через веб-інтерфейс (на http://<RPI_IP>:8086) або через командний рядок (influx setup). У процесі налаштування необхідно вказати:
  - **Ім'я користувача (Username):** Створюється головний адміністратор.
  - **Пароль (Password):** Надійний пароль для адміністратора.
  - **Назва організації (Organization):** Логічне групування; наприклад, "Greenhouse".
  - **Назва "контейнера" (Bucket):** Це аналог бази даних у InfluxDB v1.x. Створюється перший bucket для зберігання даних з сенсорів, наприклад, "sensors_data".
- **Створення API Token:** Після налаштування, у веб-інтерфейсі InfluxDB генерується **API Token (токен** доступу). Цей токен є критично важливим, оскільки саме його буде використовувати Python-скрипт для авторизації та запису даних у bucket.
- **Запуск сервісу:** Служба InfluxDB налаштовується на автоматичний запуск при кожному завантаженні Raspberry Pi: sudo systemctl enable --now influxdb

3.3.2. Розгортання системи візуалізації Grafana

Grafana надаватиме візуальний інтерфейс до даних, що зберігаються в InfluxDB.

- **Додавання репозиторію Grafana:** Аналогічно до InfluxDB, додається офіційний GPG-ключ та репозиторій Grafana.
- **Встановлення:** Встановлюється версія Grafana з відкритим кодом: sudo apt-get install grafana
- **Запуск сервісу:** Сервіс Grafana також налаштовується на автозапуск: sudo systemctl enable --now grafana-server
- **Перший вхід:** Після запуску, веб-інтерфейс Grafana стає доступним у локальній мережі за адресою http://<RPI_IP>:3000. При першому вході використовується стандартний логін/пароль (admin/admin), який система негайно вимагає змінити. Web-інтерфейс Grafana показаний на рисунку 3.2.

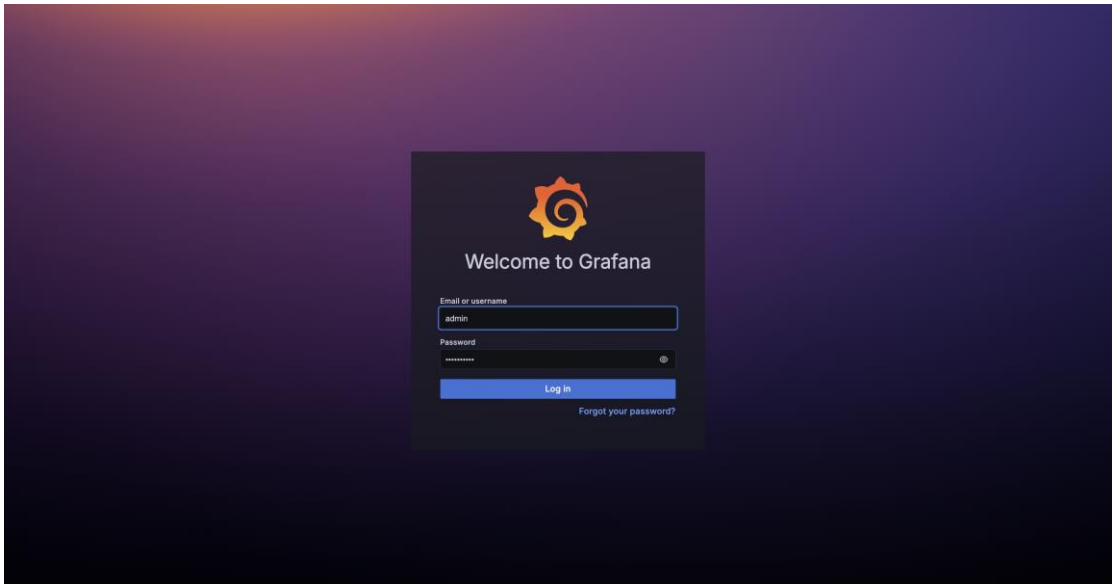

Рис.3.2. Web-інтерфейс Grafana

3.3.3. З'єднання Grafana з InfluxDB

Останній крок — "пов'язати" Grafana з InfluxDB, щоб візуалізатор знав, звідки брати дані.

1. У веб-інтерфейсі Grafana (після входу) обирається розділ Configuration -> Data Sources.
2. Натискається Add data source і обирається InfluxDB.
3. У формі налаштування вказуються параметри:
    - **Query Language:** Обирається Flux (сучасна мова запитів InfluxDB v2).
    - **URL:** Вказується http://localhost:8086, оскільки обидва сервіси працюють на одній машині.
    - **Organization:** Вказується назва організації, створена при налаштуванні InfluxDB (напр., "Greenhouse").
    - **API Token:** Вставляється той самий API Token, що був згенерований в InfluxDB.
    - **Default Bucket:** Вказується sensors_data.
4. Після натискання кнопки Save & Test Grafana намагається підключитися до InfluxDB. Успішне повідомлення "Data source is working" означає, що програмний стек розгорнуто коректно.

Після виконання цих кроків система повністю готова: InfluxDB очікує на надходження даних, а Grafana готова їх візуалізувати.

3.4. Розробка скриптів збору даних з сенсорів та їх збереження в InfluxDB

Цей етап є логічним продовженням попередніх: апаратне забезпечення підключено (3.1), операційна система налаштована (3.2), а база даних (InfluxDB) готова приймати дані (3.3). Тепер необхідно створити програмний "міст" між ними.

Для цього завдання, як було обґрунтовано в п. 2.5.1, було обрано мову програмування **Python** завдяки її простоті та наявності потужних бібліотек для взаємодії як з апаратним забезпеченням (I2C), так і з базами даних (InfluxDB).

3.4.1. Логіка роботи скрипту

Скрипт виконує п'ять основних завдань:

1. **Імпорт** необхідних бібліотек.
2. **Ініціалізація** з'єднань: підключення до шини I2C та до сенсорів, а також підключення до клієнта InfluxDB.
3. **Зчитування** даних з сенсорів BME280 (температура, вологість, тиск) та BH1750 (освітленість).
4. **Форматування** цих даних у спеціальний об'єкт "Point" (точка даних), який є сумісним з InfluxDB.
5. **Запис** (відправка) цієї точки даних у відповідний "bucket" (базу) InfluxDB.

3.4.2. Лістинг програмного коду (файл read_sensors.py)

Нижче наведено повний текст Python-скрипту, розробленого для виконання цих завдань.

Примітка: Перед запуском скрипту необхідно переконатися, що всі бібліотеки встановлені, як описано в п. 3.2 (pip install adafruit-circuitpython-bme280 adafruit-circuitpython-bh1750 influxdb-client).

```
Python
#!/usr/bin/env python3

import board      # Для доступу до пінів Raspberry Pi
import busio      # Для роботи з I2C
import adafruit_bme280.basic as adafruit_bme280
import adafruit_bh1750
import time
from influxdb_client import InfluxDBClient, Point, WritePrecision
from influxdb_client.client.write_api import SYNCHRONOUS

# --- 1. Налаштування InfluxDB ---
# Важливо: Для кращої безпеки ці дані варто виносити
# у змінні середовища, а не "зашивати" в код.
INFLUX_TOKEN = "ВАШ_ЗГЕНЕРОВАНИЙ_API_TOKEN"  # Токен з п. 3.3
```

```
INFLUX_ORG = "Greenhouse"                    # Ваша організація
INFLUX_BUCKET = "sensors_data"               # Ваш bucket
INFLUX_URL = "http://localhost:8086"         # URL до InfluxDB

# --- 2. Ініціалізація сенсорів ---
try:
    # Ініціалізація шини I2C
    i2c = board.I2C()  # Використовує стандартні піни (SDA/SCL)

    # Ініціалізація сенсора BME280 (адреса 0x76)
    bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c, address=0x76)

    # Ініціалізація сенсора BH1750 (адреса 0x23)
    bh1750 = adafruit_bh1750.BH1750(i2c, address=0x23)

    # Встановлення "морського рівня" для BME280 (опціонально, для корекції тиску)
    # bme280.sea_level_pressure = 1013.25

except Exception as e:
    print(f"Помилка ініціалізації сенсорів: {e}")
    # Тут можна додати логування помилки
    exit()

# --- 3. Ініціалізація клієнта InfluxDB ---
try:
    client = InfluxDBClient(url=INFLUX_URL, token=INFLUX_TOKEN, org=INFLUX_ORG)
    # Використовуємо синхронний запис для простоти та надійності
    write_api = client.write_api(write_options=SYNCHRONOUS)
except Exception as e:
    print(f"Помилка підключення до InfluxDB: {e}")
    exit()

# --- 4. Основна логіка: Збір та Запис ---
def collect_and_write_data():
    try:
        # --- 4.1. Зчитування даних ---
        temp = bme280.temperature
        humidity = bme280.humidity
        pressure = bme280.pressure
        lux = bh1750.lux

        # Базова валідація (захист від некоректних значень)
        if temp is None or humidity is None or pressure is None or lux is None:
            print("Не вдалося зчитати дані з сенсорів.")
            return
```

```
        # --- 4.2. Форматування даних (Point) ---
        # "microclimate" - це назва "таблиці" (measurement) в InfluxDB
        point = Point("microclimate") \
            .tag("location", "greenhouse-zone-1") \
            .field("temperature", float(temp)) \
            .field("humidity", float(humidity)) \
            .field("pressure", float(pressure)) \
            .field("light_lux", float(lux)) \
            .time(int(time.time()), WritePrecision.S) # Часова мітка (секунди)

        # --- 4.3. Запис даних ---
        write_api.write(bucket=INFLUX_BUCKET, org=INFLUX_ORG, record=point)

        # print(f"Дані успішно записано: T={temp:.2f}C, H={humidity:.2f}%, P={pressure:.2f}hPa,
L={lux:.2f}lx")

    except Exception as e:
        # Обробка можливих помилок під час роботи
        print(f"Помилка під час збору або запису даних: {e}")

# --- 5. Виконання скрипту ---
if __name__ == "__main__":
    collect_and_write_data()
```

3.4.3. Автоматизація запуску скрипту

Щоб цей скрипт виконувався автоматично (наприклад, кожні 5 хвилин), він додається до системного планувальника cron.

1. Відкривається редактор crontab командою: crontab -e
2. У кінець файлу додається рядок, який вказує інтервал та шлях до скрипту:

*/5 * * * * /usr/bin/python3 /home/pi/scripts/read_sensors.py >> /home/pi/scripts/sensor.log 2>&1

Цей запис означає: "Запускати скрипт read_sensors.py кожні 5 хвилин (*/5) кожного дня, а весь вивід (включаючи помилки 2>&1) дописувати у лог-файл sensor.log".

Таким чином, розроблений скрипт та його автоматизація забезпечують безперервний потік даних від апаратного рівня (сенсорів) до програмного рівня (бази даних).

3.5. Реалізація механізму резервного копіювання (схема 7-4-4)

Накопичення даних на microSD-карті створює ризик їх втрати. Для запобігання цьому було реалізовано гібридний механізм резервного копіювання, який включає збереження архівів локально (на USB-носій) та вивантаження їх на віддалений сервер (FTP) за схемою ротації 7-4-4.

3.5.1. Монтування зовнішнього носія (USB Flash Drive)

Першим кроком є налаштування автоматичного монтування USB-носія в систему, щоб скрипт завжди знав, куди зберігати локальні архіви.

1. USB-носій форматується (наприклад, у ext4) та отримує унікальну мітку (label), наприклад, BACKUP_USB.
2. До системного файлу /etc/fstab додається рядок, який автоматично монтує цей носій у директорію, де скрипт буде зберігати архіви (наприклад, /home/pi/backups/archives): LABEL=BACKUP_USB /home/pi/backups/archives ext4 defaults,nofail 0 0 *Параметр nofail* є критично важливим: він дозволяє Raspberry Pi завантажитися, навіть якщо флешка не підключена.

3.5.2. Вибір інструментів:

- **influxd backup:** Вбудована утиліта InfluxDB для створення консистентного "знімка" бази даних без її зупинки.
- **tar:** Утиліта Linux для архівації "знімка" в єдиний файл .tar.gz.
- **curl:** Утиліта для вивантаження архіву на віддалений FTP-сервер (реалізація off-site копії).

- **bash та cron:** Скрипт написано на bash, а cron використовується для його автоматичного запуску за розкладом.

3.5.3. Лістинг скрипту резервного копіювання (backup.sh)

Нижче наведено розроблений bash-скрипт, який реалізує повний цикл гібридного бекапу.

```
#!/bin/bash

# --- 1. Налаштування ---
# Директорія для тимчасових файлів (на SD-карті)
BACKUP_TEMP_DIR="/home/pi/backups/temp_backup"

# *** ГОЛОВНА ДИРЕКТОРІЯ АРХІВІВ (на змонтованій USB-флешці) ***
ARCHIVE_DIR="/home/pi/backups/archives"

# Налаштування FTP (з плакату)
FTP_USER="user"
FTP_PASS="password"
FTP_SERVER="ftp.server.com"
FTP_PATH="/backups"

# Перевірка, чи змонтовано USB-носій
if ! mountpoint -q $ARCHIVE_DIR; then
    echo "Помилка: Директорія архівів ($ARCHIVE_DIR) не змонтована. USB-носій відсутній?"
    exit 1
fi

# Створення директорій
mkdir -p $BACKUP_TEMP_DIR
mkdir -p $ARCHIVE_DIR/daily
mkdir -p $ARCHIVE_DIR/weekly
mkdir -p $ARCHIVE_DIR/monthly

# --- 2. Створення "знімка" InfluxDB ---
echo "Створення InfluxDB 'знімка'..."
influxd backup --bucket sensors_data -t ВАШ_API_TOKEN --host http://localhost:8086
$BACKUP_TEMP_DIR

if [ $? -ne 0 ]; then
    echo "Помилка створення 'знімка' InfluxDB!"
    exit 1
fi
```

```
# --- 3. Архівація та ротація (ЛОКАЛЬНА КОПІЯ) ---
DATE=$(date +"%Y-%m-%d")
DAY_OF_WEEK=$(date +"%u") # 1=Понеділок, 7=Неділя
DAY_OF_MONTH=$(date +"%d") # 01-31

ARCHIVE_NAME="influx_backup_$DATE.tar.gz"
ARCHIVE_PATH="$ARCHIVE_DIR/daily/$ARCHIVE_NAME"

echo "Архівація даних на USB-носій ($ARCHIVE_PATH)..."
tar -czf $ARCHIVE_PATH -C $BACKUP_TEMP_DIR .

# 4. Логіка ротації 7-4-4 (на USB-носії)
find $ARCHIVE_DIR/daily/ -name "*.tar.gz" -mtime +7 -exec rm {} \;
if [ "$DAY_OF_WEEK" == "7" ]; then
    echo "Просування до щотижневого локального архіву..."
    cp $ARCHIVE_PATH $ARCHIVE_DIR/weekly/
    find $ARCHIVE_DIR/weekly/ -name "*.tar.gz" -mtime +28 -exec rm {} \;
fi
if [ "$DAY_OF_MONTH" == "01" ]; then
    echo "Просування до щомісячного локального архіву..."
    cp $ARCHIVE_PATH $ARCHIVE_DIR/monthly/
    find $ARCHIVE_DIR/monthly/ -name "*.tar.gz" -mtime +120 -exec rm {} \;
fi

# --- 5. Вивантаження на FTP (ВІДДАЛЕНА КОПІЯ) ---
FTP_URL="ftp://$FTP_USER:$FTP_PASS@$FTP_SERVER$FTP_PATH/$ARCHIVE_NAME"

echo "Вивантаження архіву на FTP ($FTP_SERVER)..."
curl -T $ARCHIVE_PATH "$FTP_URL"

if [ $? -eq 0 ]; then
    echo "Бекап успішно вивантажено на FTP."
else
    echo "Помилка вивантаження на FTP!"
fi

# --- 6. Очищення тимчасових файлів ---
echo "Очищення тимчасових файлів..."
rm -rf $BACKUP_TEMP_DIR/*

echo "Резервне копіювання завершено."
```

3.5.4. Автоматизація запуску (Cron)

Скрипт додається до crontab для запуску щоночі (наприклад, о 02:30):

30 2 * * * /home/pi/scripts/backup.sh >> /home/pi/scripts/backup.log 2>&1

Таким чином, реалізована система щоночі створює локальну копію на USB-носії (з ротацією 7-4-4) та дублює її на віддалений FTP-сервер, забезпечуючи надійний захист даних.

3.6. Налаштування захищеного веб-доступу до Grafana

Після того, як дані збираються, зберігаються та візуалізуються локально на Raspberry Pi (доступні за адресою http://localhost:3000), виникає задача надання захищеного мобільного доступу до цього дашборду. Агроном повинен мати можливість контролювати мікроклімат зі свого смартфона чи комп'ютера, перебуваючи поза межами теплиці.

3.6.1. Проблема традиційного підходу (Port Forwarding)

Найпоширеніший "класичний" метод — це налаштування "прокидання портів" (Port Forwarding) на маршрутизаторі. Цей метод полягає у тому, щоб спрямувати весь трафік, що надходить на зовнішній IP-адрес роутера (наприклад, порт 80 або 3000), на внутрішній IP-адрес Raspberry Pi (порт 3000).

Цей підхід має критичні недоліки, особливо для мобільної платформи:

- **Загроза безпеці:** Відкритий порт робить сервіс Grafana (а потенційно й сам Raspberry Pi) видимим для всього Інтернету. Це негайно привертає увагу автоматизованих ботів-сканерів, які шукають вразливості, та відкриває систему для брутфорс-атак (підбору пароля).
- **Проблема динамічної IP-адреси:** Більшість провайдерів надають динамічну зовнішню IP-адресу, яка періодично змінюється. Це означає, що агроному доведеться постійно дізнаватися нову адресу, що робить доступ ненадійним.

3.6.2. Рішення: Технологія Cloudflare Zero Trust (Tunnel)

Для вирішення цих проблем було обрано сучасний та безпечний метод, що базується на архітектурі Zero Trust ("Нульова Довіра"), а саме — сервіс Cloudflare Tunnel (раніше відомий як Argo Tunnel).

Принцип роботи кардинально відрізняється від Port Forwarding:

1. **Жодних відкритих портів:** На маршрутизаторі не відкривається жодних вхідних портів. Raspberry Pi залишається повністю "невидимим" та ізольованим від зовнішньої мережі.
2. **Вихідний тунель:** На Raspberry Pi встановлюється легковаговий програмний агент (cloudflared), який ініціює вихідне з'єднання з найближчими дата-центрами Cloudflare. Це з'єднання є постійним та зашифрованим.
3. **Проксіювання запитів:** Коли користувач звертається до публічної URL-адреси (наприклад, greenhouse.mydomain.com), Cloudflare отримує цей запит, перевіряє його та безпечно передає через вже встановлений тунель до локального сервісу Grafana на Raspberry Pi.

3.6.3. Етапи налаштування

Процес налаштування включав наступні кроки:

- **Реєстрація домену** та додавання його до панелі керування Cloudflare.
- **Встановлення агента cloudflared** на Raspberry Pi (завантаження .deb пакунку для ARM-архітектури).
- **Автентифікація агента** командою cloudflared tunnel login, що пов'язує його з акаунтом Cloudflare.
- **Створення іменованого тунелю** командою cloudflared tunnel create <tunnel-name>.

- **Конфігурація тунелю:** Створюється конфігураційний файл (config.yml), який вказує, куди спрямовувати трафік.

Приклад файлу config.yml:

```
YAML

url: http://localhost:3000
tunnel: <Tunnel-UUID>
credentials-file: /root/.cloudflared/<Tunnel-UUID>.json
```

- **Створення DNS-запису:** У панелі Cloudflare створюється CNAME-запис, який пов'язує бажане доменне ім'я (напр., grafana.my-farm.com) з ID тунелю.
- **Запуск тунелю як сервісу:** Команда sudo cloudflared service install перетворює тунель на системну службу, яка автоматично запускається при кожному завантаженні Raspberry Pi.

3.6.4. Налаштування рівня безпеки (Email Auth)

Після запуску тунелю, дашборд Grafana стає доступним глобально як показано на рисунку 3.2. Для додавання додаткового рівня захисту поверх сторінки входу самої Grafana, було налаштовано політику доступу в панелі Cloudflare Zero Trust:

- У розділі Access -> Applications було створено нову "Self-hosted" програму.
- Для цієї програми було налаштовано політику Allow (Дозволити) з правилом Email Authentication.
- Було додано електронні пошти тих людей яким дозволено доступ. Тепер, при спробі відкрити grafana.bme280.me , Cloudflare спершу показує стандартне вікно браузера, що вимагає введення Email на яку прийде одноразовий PIN як показано на Рис 3.3. І лише після успішної автентифікації користувач потрапляє на сторінку входу в Grafana, де

має ввести свої облікові дані Grafana. Це створює надійний двофакторний захист системи.

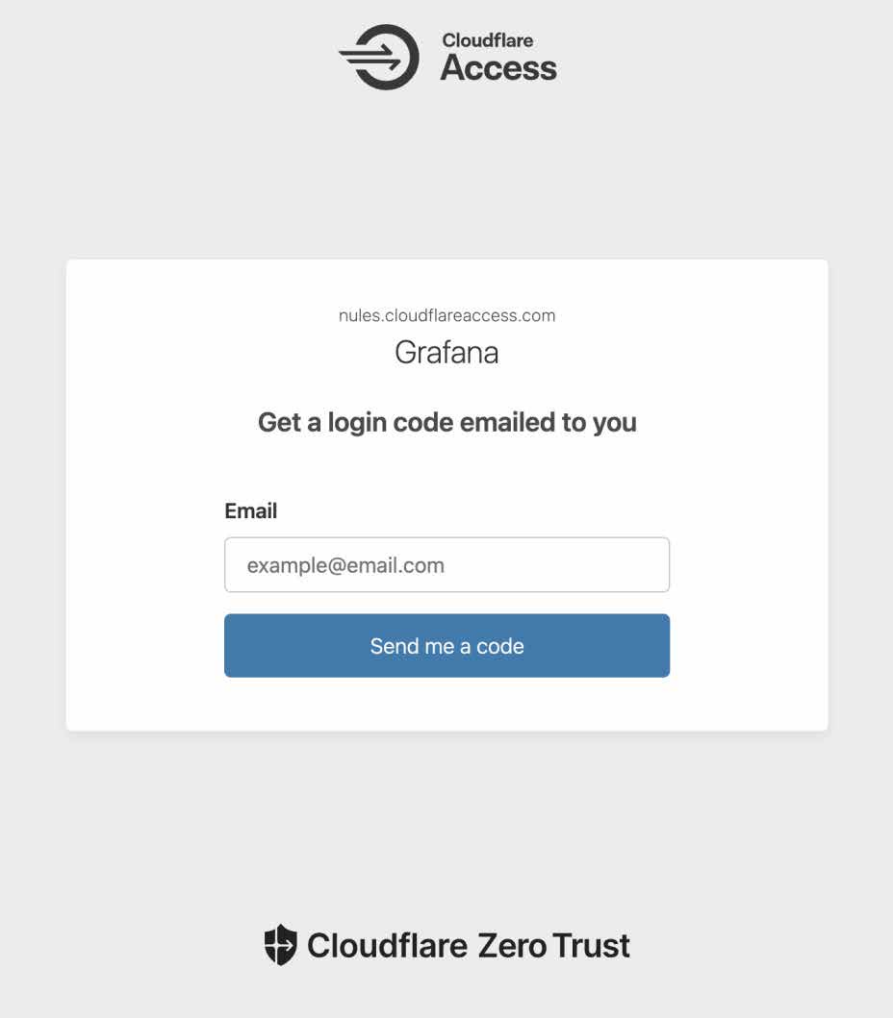

Рис 3.3.Аутентифікація через Email.

3.7. Обробка та візуалізація даних мікроклімату

На цьому етапі розробки система вже повністю функціональна: сенсори надсилають дані (3.4), база даних InfluxDB їх накопичує (3.3), а сервіс Grafana має доступ до цієї бази (3.3). Залишається останній крок — створити візуальний інтерфейс (дашборд) для агронома, який перетворить масиви чисел на зрозумілу та інформативну панель.

3.7.1. Створення дашборду (Dashboard)

Усі налаштування візуалізації виконуються у веб-інтерфейсі Grafana. Було створено новий дашборд під назвою «Моніторинг мікроклімату» (аналогічно до плакату). Дашборд складається з набору панелей (Panels), кожна з яких відображає певний параметр.

3.7.2. Типи візуалізацій

Для наочного представлення даних було обрано два основні типи візуалізацій:

- **Графік (Time series):** Це основний інструмент для відображення *динаміки* параметрів у часі. Використовується для побудови графіків температури, вологості, тиску та освітленості.
- **Індикатор (Gauge):** Цей віджет використовується для відображення *поточного* (останнього актуального) значення параметра. Це дозволяє оператору миттєво оцінити стан теплиці "тут і зараз" .

3.7.3. Налаштування панелей та запити до InfluxDB

Для кожної панелі (наприклад, "Температура") необхідно вказати Grafana, які саме дані отримувати з InfluxDB. Це робиться за допомогою мови запитів Flux (або InfluxQL).

Приклад запиту мовою Flux для відображення графіка температури:

Фрагмент коду:

```
from(bucket: "sensors_data")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "microclimate")
  |> filter(fn: (r) => r["_field"] == "temperature")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

Пояснення запиту:

1. from(bucket: "sensors_data"): Взяти дані з нашого "контейнера".
2. |> range(...): Обмежити часовий діапазон тим, що обрано на дашборді (напр., "Останні 6 годин").
3. |> filter(...): Відфільтрувати лише потрібні дані (вимірювання microclimate та поле temperature).
4. |> aggregateWindow(...): Це ключова функція обробки. Вона автоматично усереднює дані (fn: mean) у часові "вікна" (every: v.windowPeriod). Це дозволяє Grafana показувати плавний графік, а не мільйони окремих точок, що значно прискорює завантаження.

Приклад запиту для індикатора (Gauge) поточної температури:

Фрагмент коду:

```
from(bucket: "sensors_data")
  |> range(start: -10m) // Беремо дані лише за останні 10 хвилин
  |> filter(fn: (r) => r["_measurement"] == "microclimate")
  |> filter(fn: (r) => r["_field"] == "temperature")
  |> last() // Обираємо ОДНЕ останнє (найсвіжіше) значення
```

За цим принципом було створено 4 пари візуалізацій (індикатор + графік) для чотирьох параметрів:

- Температура
- Вологість
- Тиск
- Освітленість

Результат роботи — повноцінна інтерактивна панель моніторингу, де агроном може відстежувати поточні показники, а також аналізувати історичні дані, масштабуючи та переміщуючи часові діапазони на графіках. Фінальний вигляд дашборду представлений на рис. 3.2.

Рис. 3.2. Фінальний вигляд дашборду моніторингу мікроклімату в Grafana

Цей дашборд є кінцевим продуктом для користувача системи, що надає йому всю необхідну інформацію для прийняття рішень.

4 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

У попередньому розділі було описано процес розгортання апаратних та програмних компонентів. Цей розділ присвячено опису логіки функціонування розробленого програмного забезпечення, його алгоритмів та результатам тестування прототипу в реальних умовах.

4.1. Загальний алгоритм функціонування системи

Загальний алгоритм функціонування системи описує повний життєвий цикл її роботи, починаючи від ввімкнення живлення до виконання всіх ключових завдань: збору, зберігання, резервування та візуалізації даних. Система розроблена для повністю автономної роботи без необхідності втручання оператора після початкового налаштування. Алгоритм базується на взаємодії системних служб (services) та завдань, що виконуються за розкладом (cron

jobs). Блок-схему загального алгоритму функціонування системи наведено на рис. 4.1.

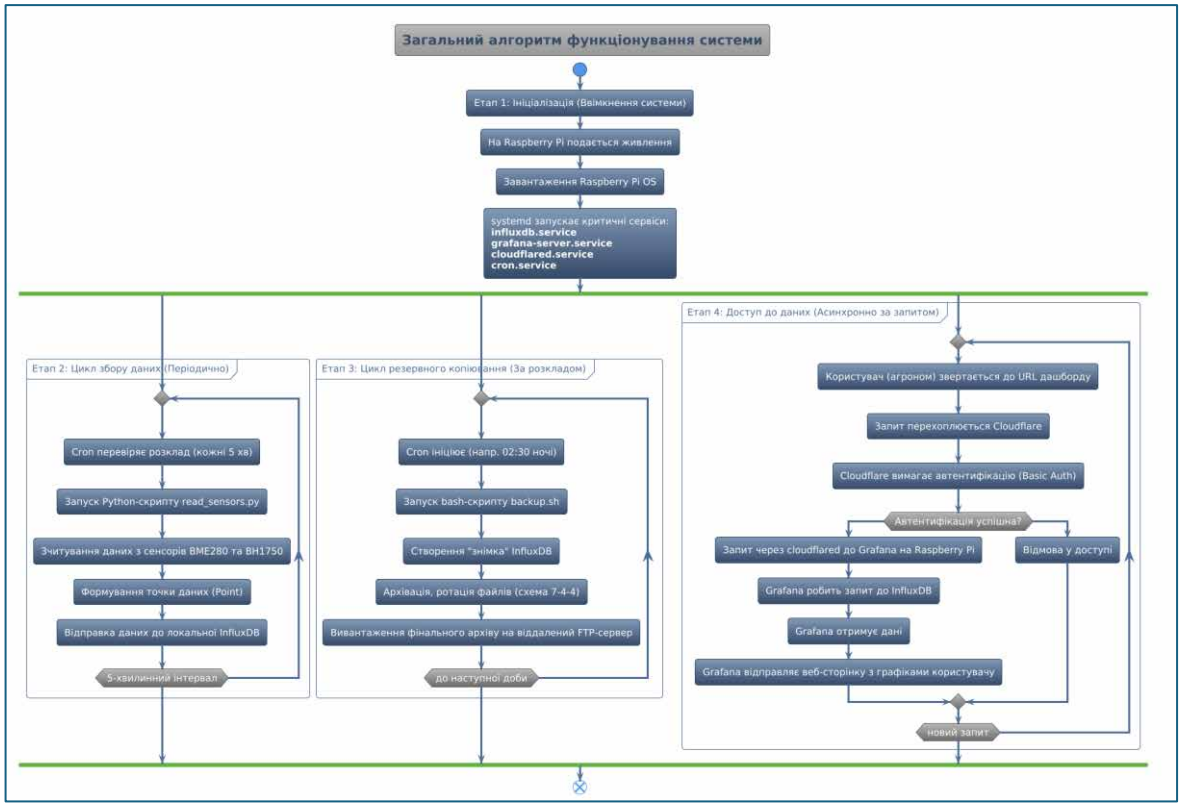

Рис. 4.1. Блок-схема загального алгоритму функціонування системи

4.1.1. Покроковий опис алгоритму

- **Етап 1: Ініціалізація (Ввімкнення системи)**
  - На Raspberry Pi подається живлення.
  - Відбувається завантаження операційної системи Raspberry Pi OS.
  - Системний менеджер systemd автоматично запускає всі критично важливі сервіси, які були налаштовані на автозапуск (як описано в Розділі 3):
    1. Запускається сервер бази даних influxdb.service.
    2. Запускається веб-сервер візуалізації grafana-server.service.
    3. Запускається сервіс захищеного тунелю cloudflared.service.
    4. Запускається планувальник завдань cron.service.
- **Етап 2: Цикл збору даних (Виконується періодично)**

- Планувальник cron перевіряє свій розклад.
- Кожні 5 хвилин (згідно з налаштуванням */5 * * * *) cron ініціює запуск Python-скрипту read_sensors.py.
- Скрипт виконує зчитування даних з сенсорів BME280 та BH1750 через шину I2C.
- Скрипт формує точку даних (Point) та відправляє її до локальної бази даних InfluxDB.
- Цикл повертається до очікування наступного 5-хвилинного інтервалу.

- **Етап 3: Цикл резервного копіювання (Виконується за розкладом)**
  - В установлений час (наприклад, 02:30 ночі) cron ініціює запуск bash-скрипту backup.sh.
  - Скрипт backup.sh створює "знімок" бази даних InfluxDB.
  - Скрипт виконує архівацію, ротацію файлів за схемою 7-4-4 та вивантажує фінальний архів на віддалений FTP-сервер.
  - Цикл завершується до наступної доби.
- **Етап 4: Доступ до даних (Виконується асинхронно за запитом)**
  - У будь-який момент часу користувач (агроном) може звернутися до публічної URL-адреси дашборду.
  - Запит перехоплюється Cloudflare.
  - Cloudflare вимагає автентифікацію (Email Auth).
  - Після успішної автентифікації запит передається через захищений тунель (cloudflared) до локального сервера Grafana на Raspberry Pi.
  - Grafana, в свою чергу, робить запит до InfluxDB, отримує дані та відправляє користувачу готову веб-сторінку з графіками.

Ці чотири процеси виконуються паралельно та незалежно, забезпечуючи стабільну та надійну роботу системи 24/7.

4.2. Алгоритм роботи скриптів збору даних та резервного копіювання

Програмне забезпечення системи складається з двох ключових скриптів, які працюють незалежно: скрипт збору даних (на Python) та скрипт резервного копіювання (на Bash). Їхня логіка є основою для автономної роботи системи.

4.2.1. Алгоритм скрипту збору даних (read_sensors.py)

Цей Python-скрипт відповідає за опитування сенсорів та запис даних до InfluxDB. Він запускається планувальником cron кожні 5 хвилин. Алгоритм його роботи представлено на рис. 4.2.

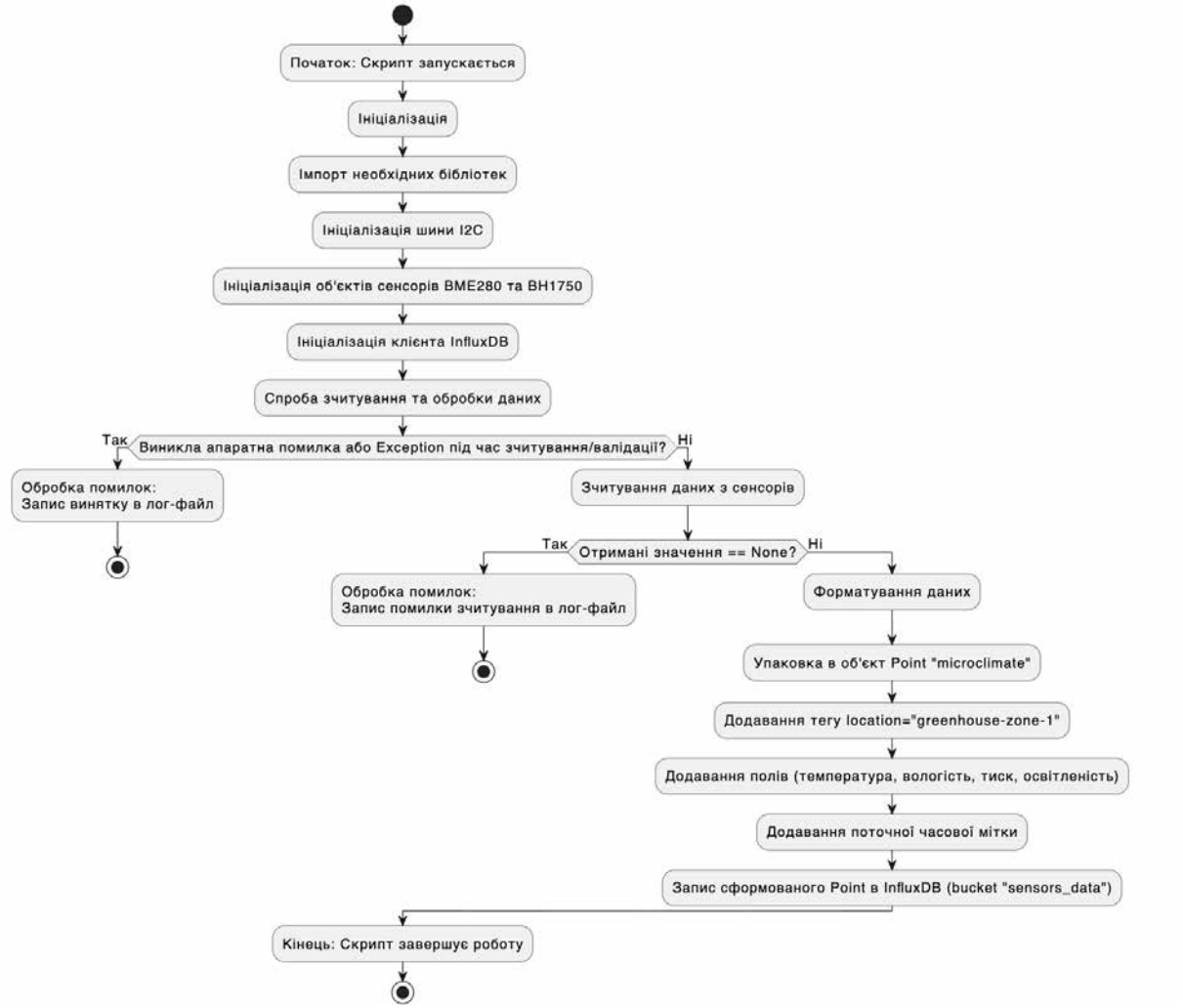

Рис. 4.2. Блок-схема алгоритму роботи скрипту збору даних

Опис алгоритму:

1. **Початок:** Скрипт запускається.
2. **Ініціалізація:**
    - Імпортуються необхідні бібліотеки (InfluxDB, Adafruit_BME280, Adafruit_BH1750, board, busio).
    - Ініціалізується шина I2C.
    - Ініціалізуються об'єкти сенсорів BME280 та BH1750, вказуючи їхні I2C-адреси.
    - Ініціалізується клієнт InfluxDB з необхідними параметрами (URL, Token, Org).
3. **Блок try-except (Обробка помилок):** Весь процес збору та запису "обгорнутий" у блок try-except для запобігання "падінню" скрипту у випадку апаратного збою (наприклад, "відвалу" сенсора).
4. **Зчитування даних:** Виконуються виклики методів .temperature, .humidity, .pressure та .lux для отримання поточних значень з сенсорів.
5. **Валідація:** Проводиться проста перевірка, чи не є отримані значення None (що свідчить про помилку зчитування).
6. **Форматування:** Якщо дані валідні, вони "упаковуються" в об'єкт Point ("microclimate"), де додаються:
    - **Тег:** location = "greenhouse-zone-1".
    - **Поля:** temperature, humidity, pressure, light_lux.
    - **Часова мітка:** Поточний час (time.time()).
7. **Запис:** Сформований об'єкт Point записується у bucket "sensors_data" за допомогою write_api.write().
8. **Обробка помилок:** Якщо на будь-якому етапі (від ініціалізації до запису) виникає виняток (Exception), помилка записується у лог-файл (завдяки налаштуванням cron), і скрипт коректно завершує роботу.
9. **Кінець:** Скрипт завершує роботу до наступного виклику.

4.2.2. Алгоритм скрипту резервного копіювання (backup.sh)

Цей Bash-скрипт відповідає за створення "знімків" бази даних, їх ротацію за схемою 7-4-4 та вивантаження на FTP. Він запускається cron один раз на добу (наприклад, о 02:30). Блок-схема алгоритму роботи скрипту резервного копіювання вказана на рисунку 4.3.

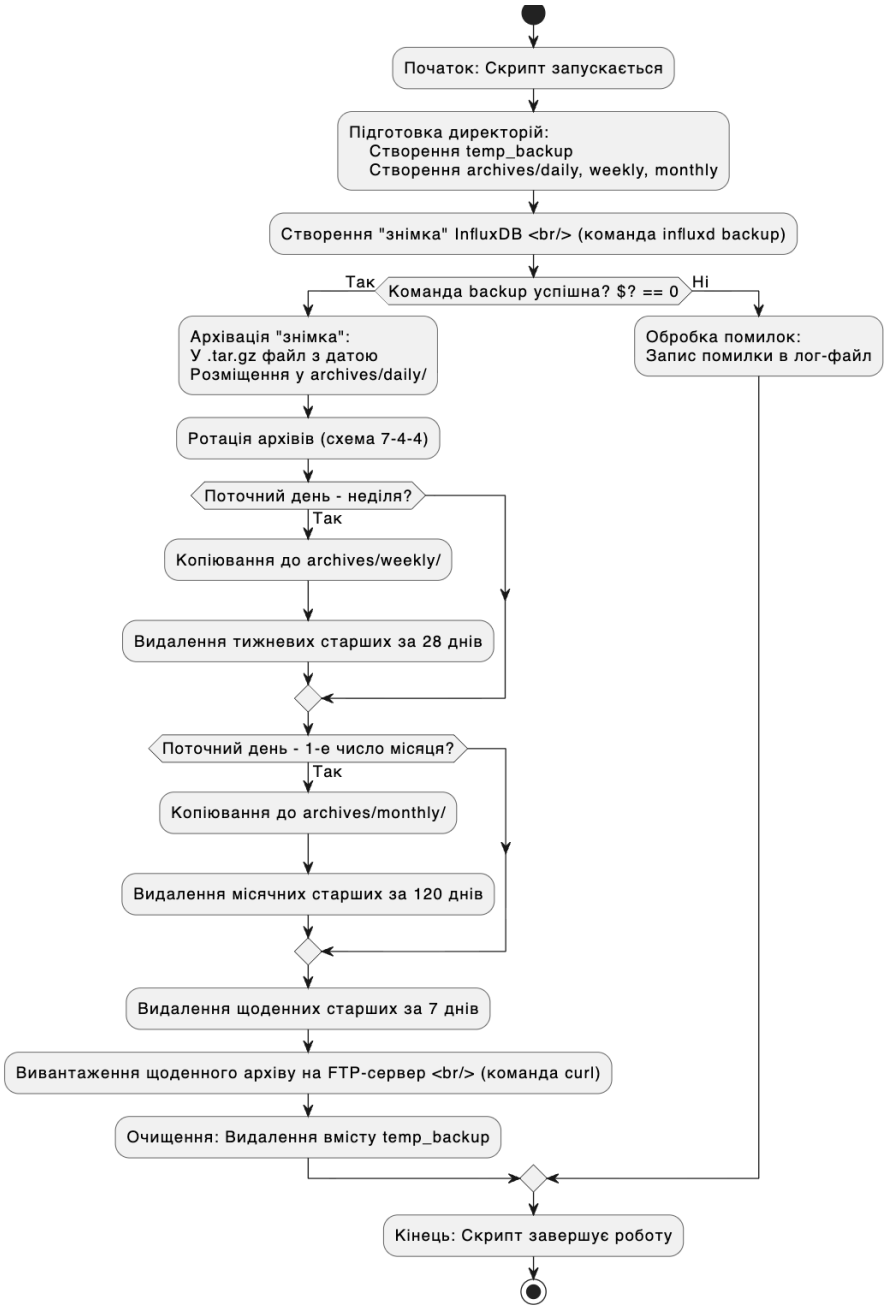

Рис. 4.3. Блок-схема алгоритму роботи скрипту резервного копіювання

Опис алгоритму:

1. **Початок:** Скрипт запускається.
2. **Підготовка:** Створюються (якщо не існують) директорії для тимчасових файлів (temp_backup) та архівів (archives/daily, weekly, monthly).
3. **Створення "знімка":** Виконується команда influxd backup, яка зберігає консистентний знімок бази даних у тимчасову директорію.
4. **Перевірка успішності:** Скрипт перевіряє код повернення ($?) попередньої команди. Якщо він не дорівнює 0 (сталася помилка), скрипт записує помилку в лог і завершує роботу.
5. **Архівація:** Якщо знімок створено успішно, він архівується у .tar.gz файл з поточною датою у назві та розміщується у директорії archives/daily/.
6. **Ротація (7-4-4):**
    - **Щоденна:** Виконується find для видалення всіх архівів у archives/daily/, старших за 7 днів (-mtime +7).
    - **Щотижнева:** Перевіряється, чи є поточний день неділею (DAY_OF_WEEK == 7). Якщо так, щоденний архів копіюється у archives/weekly/, після чого запускається find для видалення тижневих архівів, старших за 28 днів (-mtime +28).
    - **Щомісячна:** Перевіряється, чи є поточний день 1-м числом місяця (DAY_OF_MONTH == 01). Якщо так, архів копіюється у archives/monthly/, після чого запускається find для видалення місячних архівів, старших за 120 днів (-mtime +120).
7. **Вивантаження на FTP:** Виконується команда curl для завантаження щоденного архіву (.tar.gz) на вказаний FTP-сервер, використовуючи облікові дані з конфігурації.
8. **Очищення:** Вміст тимчасової директорії (temp_backup) видаляється.
9. **Кінець:** Скрипт завершує роботу.

4.3. Проведення тестування системи в умовах теплиці

Після завершення розробки та початкового налаштування програмного забезпечення було проведено тестування прототипу. Це ключовий етап, який дозволяє перевірити працездатність, стабільність та надійність системи не в "лабораторних", а в реальних експлуатаційних умовах.

4.3.1. Мета тестування

- Перевірити стабільність роботи Raspberry Pi та програмних сервісів (InfluxDB, Grafana, cloudflared) у режимі безперервної роботи 24/7.
- Підтвердити коректність та регулярність збору даних з сенсорів BME280 та BH1750 у реальному середовищі теплиці (з її перепадами температур та вологості).
- Верифікувати коректність роботи алгоритму резервного копіювання (запуск за розкладом, створення архівів та їх вивантаження на FTP).
- Оцінити стабільність та швидкість захищеного мобільного доступу до дашборду Grafana через Cloudflare Tunnel.

4.3.2. Умови та методологія тестування

- **Місце:** Прототип (Raspberry Pi з сенсорами у захисному корпусі) було розміщено у діючій теплиці.
- **Тривалість:** Було проведено безперервний моніторинг тривалістю 6 діб (144 години).
- **Контроль:**
    - **Збір даних:** Перевірявся cron-лог (sensor.log) на предмет регулярного запуску скрипту кожні 5 хвилин та відсутності помилок зчитування сенсорів.
    - **Зберігання даних:** Аналізувалася база даних InfluxDB через веб-інтерфейс Grafana на предмет наявності даних та відсутності "прогалин" (lags) у часових рядах.

- **Резервне копіювання:** Щоденно перевірялася наявність нового .tar.gz архіву на віддаленому FTP-сервері.
- **Доступ:** Виконувалися регулярні (3-4 рази на добу) підключення до дашборду Grafana з мобільного телефону (через 4G-мережу) та ноутбука (з іншої Wi-Fi мережі) для перевірки стабільності тунелю.

4.3.3. Результати тестування

Тестування показало високу надійність та повну працездатність розробленого прототипу.

- **Стабільність збору даних:** Протягом 6-добового періоду система стабільно збирала та записувала дані. Зафіксовано 100% успішних запусків скрипту збору даних . Дані з сенсорів надходили коректно, відображаючи реальні добові цикли (зниження температури вночі, піки освітленості вдень).
- **Стабільність роботи платформи:** За 144 годин тестування не було зафіксовано жодного апаратного збою, зависання Raspberry Pi або непередбаченого перезавантаження. Усі ключові сервіси (influxdb, grafana-server, cloudflared, cron) працювали у штатному режимі.
- **Функціональність резервного копіювання:** Скрипт backup.sh успішно спрацьовував щоночі о 02:30. На FTP-сервері було успішно збережено 5 щоденних архівів, що підтвердило коректність роботи утиліт influxd backup та curl.
- **Якість віддаленого доступу:** Доступ через Cloudflare Tunnel виявився стабільним. Час завантаження дашборду Grafana на мобільному пристрої через 4G-мережу був прийнятним (в середньому 4-6 секунд), що є достатнім для оперативного моніторингу. Механізм захисту Email Auth спрацьовував коректно при кожному підключенні.

Візуальним підтвердженням успішного збору та обробки даних є графіки, отримані на дашборді Grafana під час тестування представлені на рисунку 4.5

Рис.4.5. Графіки дашборду Grafana за 6 днів.

Висновок за результатами тестування: Прототип системи повністю підтвердив свою працездатність в умовах, наближених до реальних. Він успішно вирішує всі поставлені задачі: автономно збирає дані, надійно їх зберігає, автоматично створює резервні копії та надає стабільний захищений мобільний доступ.

4.4. Висновки до четвертого розділу

У даному розділі було представлено логічну структуру розробленого програмного забезпечення та результати його практичного тестування.

На основі проведеної роботи можна зробити наступні висновки:

- **Розроблено та описано загальний алгоритм функціонування системи**, який базується на паралельній та автономній роботі системних служб (influxdb, grafana-server, cloudflared) та завдань, що виконуються за розкладом (cron). Такий підхід забезпечує високу стабільність та не потребує втручання оператора після початкового налаштування.
- **Деталізовано алгоритми ключових програмних компонентів:**

    - **Скрипт збору даних (read_sensors.py)** реалізує надійний механізм опитування сенсорів з обробкою можливих помилок, що гарантує цілісність потоку даних.
    - **Скрипт резервного копіювання (backup.sh)** повністю автоматизує процес створення "знімків" бази даних, їх ротацію за схемою 7-4-4 та вивантаження на зовнішній FTP-сервер, що забезпечує надійний захист даних.
- **Проведено успішне натурне тестування прототипу** в умовах реальної теплиці протягом 5 діб. Результати тестування повністю підтвердили працездатність та надійність розробленого рішення:
    - Система продемонструвала стабільну роботу в режимі 24/7 без збоїв та зависань.
    - Механізми збору, зберігання, резервного копіювання та захищеного віддаленого доступу функціонували у штатному режимі, повністю відповідаючи поставленим вимогам.

Таким чином, розроблене програмне забезпечення є повністю функціональним, а прототип системи готовий до практичного застосування.

ВИСНОВКИ

У даній дипломній роботі було вирішено актуальну науково-практичну задачу розробки та дослідження мобільної системи збору та резервного зберігання даних мікроклімату на базі одноплатного комп'ютера Raspberry Pi. За результатами виконаної роботи можна зробити наступні висновки.

- Проведено детальний аналіз предметної області, в ході якого було визначено ключові функціональні задачі систем моніторингу в аграрному секторі, розглянуто класифікацію існуючих систем та досліджено особливості сучасних технологій. Обґрунтовано доцільність використання мобільних IoT-платформ, спеціалізованих

баз даних часових рядів (TSDB), надійних схем резервного копіювання (GFS) та сучасних методів організації захищеного доступу (Zero Trust).

- На основі аналізу було розроблено та обґрунтовано архітектуру системи. Сформовано її структурну та функціональну схеми, які описують повний цикл руху даних: від збору сенсорами BME280 та BH1750, обробки на Raspberry Pi за допомогою Python-скриптів, зберігання у базі даних InfluxDB, візуалізації у Grafana до резервування за схемою 7-4-4 та надання захищеного доступу через Cloudflare Tunnel.
- Виконано повний цикл практичної реалізації системи. Було розроблено схему підключення апаратних компонентів, виконано налаштування операційної системи Raspberry Pi OS та розгорнуто весь необхідний програмний стек. Розроблено програмне забезпечення для збору даних та автоматизованого резервного копіювання, а також створено інтерактивний дашборд для візуалізації параметрів мікроклімату.
- Проведено успішне натурне тестування розробленого прототипу в реальних умовах протягом 7 діб. Тестування підтвердило високу стабільність, надійність та повну працездатність системи. Всі компоненти функціонували у штатному режимі, забезпечуючи безперервний збір даних, їх коректне збереження, автоматичне резервне копіювання та стабільний мобільний доступ.
- Наукова новизна роботи полягає у розробці комплексної архітектури мобільної системи, що поєднує локальну обробку даних на периферійному пристрої зберігання у спеціалізованій TSDB, реалізацію гібридної схеми резервного копіювання та застосування технології Zero Trust для організації безпечного доступу, що разом підвищує автономність, надійність та захищеність рішення.
- Практичне значення результатів полягає у створенні готового до впровадження, бюджетного та енергоефективного прототипу системи,

який дозволяє агрономам оперативно аналізувати мікроклімат теплиці та приймати обґрунтовані управлінські рішення, що сприяє цифровизації аграрного сектору та підвищенню ефективності господарства.

Таким чином, мета дипломної роботи досягнута повністю, а всі поставлені задачі — виконані. Розроблена система є завершеним програмно-апаратним комплексом, що вирішує актуальну проблему моніторингу та надійного зберігання аграрних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Al-Fuqaha, A., Guibene, W., Mohammadi, M., Ayyash, M., & Khreishah, A. Internet of Things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 2015. Vol. 17, no. 4. P. 2347–2376.
2. Asay, M. Why time series databases are exploding in popularity. TechRepublic. URL: https://www.techrepublic.com/article/why-time-series-databases-are-exploding-in-popularity/ (дата звернення: 10.11.2025).
3. Bahga, A., & Madisetti, V. Internet of Things: A Hands-On Approach. VPT, 2014. 506 p.
4. Çelik, M. Data Processing Systems: Characteristics and Components. Journal of Data Analysis and Information Processing, 2017. Vol. 5, no. 2. P. 34–45.
5. Elijah, O., Rahman, T. A., Orikumhi, I., Leow, C. Y., & Hindia, M. N. An IoT-based automated greenhouse monitoring system using Raspberry Pi. Journal of Physics: Conference Series, 2019. Vol. 1378, no. 4. P. 042017.
6. Ferrández-Pastor, F. J., García-Chamizo, J. M., Nieto-Hidalgo, M., & Mora-Martínez, J. Precision agriculture design and implementation of a platform for data acquisition and ubiquitous monitoring of greenhouses. Computers and Electronics in Agriculture, 2018. Vol. 154. P. 244–257.
7. Hassan, Q., Khan, J. A., & Bhatti, F. A. IoT based smart greenhouse monitoring and controlling system using Arduino. International Journal of Advanced Computer Science and Applications, 2020. Vol. 11, no. 1.
8. Lutz, M. Learning Python. 5th ed. O'Reilly Media, 2013. 1648 p.
9. Monk, S. Raspberry Pi Cookbook: Software and Hardware Problems and Solutions. 4th ed. O'Reilly Media, 2023. 550 p.

10. Myers, G., Sandler, C., & Badgett, T. The Art of Software Testing. 3rd ed. John Wiley & Sons, 2011. 288 p.
11. Peterson, G. The Art of Data Backup: A Comprehensive Guide to Protecting Your Digital Life. O'Reilly Media, 2021. 200 p.
12. Ray, P. P. A survey on Internet of Things architectures. Journal of King Saud University-Computer and Information Sciences, 2018. Vol. 30, no. 3. P. 291–307.
13. Sadler, G. Cloudflare Tunnel: A Free, Secure Alternative to Port Forwarding. The New Stack. URL: https://thenewstack.io/cloudflare-tunnel-a-free-secure-alternative-to-port-forwarding/ (дата звернення: 11.11.2025).
14. Shamshiri, R. R., Jones, J. W., Thorp, K. R., Ahmad, D., Man, H. C., & Taheri, S. Research and development in greenhouse technologies for sustainable food production. Environment, Development and Sustainability, 2018. Vol. 20. P. 1–28.
15. Silberschatz, A., Korth, H. F., & Sudarshan, S. Database System Concepts. 7th ed. McGraw-Hill, 2019. 1488 p.
16. Adafruit. Adafruit BH1750 - Pi & Linux. Adafruit Learning System. URL: https://learn.adafruit.com/adafruit-bh1750-ambient-light-sensor/python-circuitpython-kernel-i2c-setup (дата звернення: 11.11.2025).
17. Adafruit. Adafruit CircuitPython BME280 - Pi & Linux. Adafruit Learning System. URL: https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout/python-circuitpython-kernel-i2c-setup (дата звернення: 11.11.2025).
18. Bosch Sensortec. BME280 Datasheet. bosch-sensortec.com. URL: https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/ (дата звернення: 11.11.2025).
19. Cloudflare. What is a Zero Trust network?. Cloudflare Learning Center. URL: https://www.cloudflare.com/learning/security/zero-trust/what-is-a-zero-trust-network/ (дата звернення: 11.11.2025).
20. Cron - Wikipedia. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Cron (дата звернення: 12.11.2025).
21. Embedded Linux Wiki. Using i2c-tools in Linux. elinux.org. URL: https://elinux.org/Using_i2c-tools (дата звернення: 11.11.2025).
22. Grafana Labs. Install Grafana on Debian or Ubuntu. Grafana Documentation. URL: https://grafana.com/docs/grafana/latest/installation/debian/ (дата звернення: 11.11.2025).
23. InfluxData. What is a Time Series Database?. InfluxData.com. URL: https://www.influxdata.com/what-is-a-time-series-database/ (дата звернення: 11.11.2025).

24. IBM. What is Edge Computing?. IBM Documentation. URL: https://www.ibm.com/topics/edge-computing (дата звернення: 10.11.2025).
25. Raspberry Pi Foundation. Configuring a Raspberry Pi. Raspberry Pi Documentation. URL: https://www.raspberrypi.com/documentation/computers/configuration.html (дата звернення: 11.11.2025).
26. Rohm Semiconductor. BH1750FVI Datasheet. rohm.com. URL: https://www.rohm.com/products/optical-sensors/ambient-light-sensor-ics/bh1750fvi-e-documents (дата звернення: 11.11.2025).
27. systemd - System and Service Manager. Freedesktop.org. URL: https://www.freedesktop.org/wiki/Software/systemd/ (дата звернення: 12.11.2025).
28. Veeam. Backup Types: Full, Incremental, and Differential Explained. Veeam Software Official Blog. URL: https://www.veeam.com/blog/backup-types.html (дата звернення: 11.11.2025).