Міністерство освіти і науки україни
Національний університет “Львівська політехніка”
Інститут комп'ютерних наук та інформаційних технологій
Кафедра систем автоматизованого проектування

Курсова робота

з дисципліни
«Проектування та розробка інформаційних систем»
на тему:
“Розробка інформаційної системи моніторингу погодних умов у реальному часі”

Виконав:
ст. гр. ПП-34
Кущик А.С.

Прийняв:
Колесник К.К.

ЛЬВІВ – 2025

ЗМІСТ

Мета роботи:

Розробити робочий прототип персональної метеостанції, орієнтованої на потреби агрономів, яка вимірює основні метеорологічні параметри (температуру, вологість, атмосферний тиск тощо) та передає дані на сервер для збереження, обробки й візуалізації у веб-інтерфейсі. Забезпечити функціонал попередження агрономів про критичні зміни погодних умов у реальному часі.

Хід роботи
ПЛАНУВАННЯ

1. Тип інформаційної системи

Розроблено інформаційно-вимірювальну систему, що інтегрує вбудовану систему на базі мікроконтролера ESP32 з веб-додатком для збору, обробки, зберігання та візуалізації метеорологічних даних у реальному часі, з акцентом на потреби фермерів.

1.1 Бізнес-потреба

Агрономи та інші користувачі, чия діяльність залежить від погодних умов, потребують точних, локалізованих і оперативних метеоданих для прийняття своєчасних рішень. Неточна чи запізніла інформація може спричинити фінансові втрати або зниження ефективності. Система забезпечує моніторинг температури, вологості, атмосферного тиску, опадів і якості повітря в конкретній місцевості, дозволяючи встановлювати частоту постачання даних на сервер та граничні значення параметрів із сповіщеннями у разі їх перевищення. Доступ до даних надається через зручний веб-інтерфейс із графіками та історією вимірювань, що підтримує ефективне планування та управління.

2. Компоненти інформаційної системи

Інформаційна система персональної метеостанції, орієнтованої на потреби фермерів, складається з кількох взаємопов’язаних компонентів, які забезпечують повний цикл збору, обробки, зберігання та візуалізації метеорологічних даних. Ці компоненти можна класифікувати за такими категоріями: апаратне забезпечення, програмне забезпечення, дані, людські ресурси та організаційні процеси. Кожен компонент відіграє ключову роль у забезпеченні функціональності системи, її енергоефективності, надійності та зручності для кінцевих користувачів. Нижче детально розглянуто кожну категорію.

2.1 Апаратне забезпечення

Апаратне забезпечення системи є основою для збору метеорологічних даних і їх передачі на сервер. Воно розроблено з урахуванням компактності, енергоефективності та можливості автономної роботи в польових умовах, що особливо важливо для фермерів, які працюють у віддалених місцевостях.

1) Метеостанція:

- Використовується мікроконтролер esp32 як основний обчислювальний вузол для збору даних із сенсорів і передачі їх через Wi-Fi. ESP32 обрано завдяки його вбудованому модулю Wi-Fi, низькому енергоспоживанню та підтримці багатозадачності. Мікроконтролер забезпечує стабільну роботу навіть за умов обмеженого живлення, що робить його ідеальним для автономних метеостанцій.
- Сенсори:
  - Температури, вологості повітря та тиску (BMP280): Високоточний цифровий датчик, який вимірює температуру в діапазоні від -40°C до +80°C, відносну вологість від 0% до 100% та тиск у діапазоні від 300 до 1100 гПа. Частота зчитування даних налаштовується користувачем (наприклад, кожні 10 хвилин).
  - Датчик якості повітря (ENS160): Датчик, що CO2, AQI, TVOC.
  - HALL-сенсор для відслідковування кількості опадів.
- Модуль зв’язку: Вбудований Wi-Fi-модуль ESP32 забезпечує передачу даних на локальний сервер через HTTP або MQTT-протоколи. Для віддалених місцевостей із відсутністю Wi-Fi можливе підключення модуля LoRa для передачі даних на великі відстані.

2) Локальний сервер:

Використовується для обробки та зберігання даних, отриманих від метеостанції. Сервер може бути реалізований на базі компактного одноплатного комп’ютера, наприклад, Raspberry Pi 4, який підтримує стабільну роботу операційної системи та бази даних. Сервер забезпечує локальну обробку даних і передачу їх до хмарного сховища за наявності підключення до Інтернету.

3) Мережеве обладнання:

Wi-Fi-роутер або точка доступу для забезпечення зв’язку між метеостанцією та сервером. У разі використання LoRa-модуля необхідний LoRa-шлюз для прийому даних і їх передачі до сервера через Інтернет. Для забезпечення стабільності зв’язку в сільській місцевості можуть використовуватися підсилювачі сигналу або 4G-модеми.

4) Блок живлення:

Метеостанція підтримує живлення через USB (5 В) для стаціонарного використання або акумуляторну батарею (наприклад, Li-Ion 18650) для автономної роботи. Додатково може бути встановлена сонячна панель потужністю 5–10 Вт для зарядки батареї в польових умовах, що підвищує енергоефективність системи.

2.2 Програмне забезпечення

Програмне забезпечення системи забезпечує обробку даних, їх зберігання, аналіз та зручну візуалізацію для користувачів. Воно поділяється на три основні компоненти: прошивку метеостанції, серверну частину та клієнтську частину.

1) Прошивка метеостанції:

Реалізована на мові C++ у середовищі ESP-IDF. Прошивка відповідає за:

- Зчитування даних із сенсорів (температура, вологість, тиск, опади, якість повітря).
- Формування JSON-об’єктів для передачі даних.
- Надсилання даних на сервер через HTTP POST-запити або MQTT-протокол.
- Прошивка підтримує енергоощадний режим (deep sleep), що дозволяє економити заряд батареї, активуючи метеостанцію лише для зчитування та передачі даних.

2) Серверна частина (back-end):

Розроблена на базі фреймворку Flask (Python) для обробки запитів від метеостанції та клієнтського веб-додатку. Основні функції:

- Прийом і валідація даних від метеостанції.
- Зберігання даних у базі MongoDB.
- Аналіз даних для виявлення перевищення порогових значень (наприклад, температури вище 30°C або вологості нижче 20%).
- Надсилання сповіщень користувачам.
- Сервер підтримує API (RESTful) для взаємодії з клієнтською частиною та метеостанцією.

3) База даних:

- Використовується MongoDB для зберігання метеорологічних даних у форматі JSON. Переваги MongoDB включають гнучкість структури даних і швидкий доступ до великих обсягів інформації.
- Для локального керування базою даних застосовується MongoDB Compass, що дозволяє адміністраторам переглядати, редагувати та аналізувати дані.

4) Клієнтська частина (front-end):

Основні функції:

- Візуалізація даних у вигляді графіків (температура, вологість, тиск, опади) за вибраний період (день, тиждень, місяць).
- Налаштування порогових значень для сповіщень (наприклад, температура <10°C або >30°C).
- Відображення історії вимірювань у табличному вигляді.
- Адаптивний дизайн, що підтримує використання на смартфонах, планшетах і комп'ютерах.
- Для побудови графіків використовується бібліотека Chart.js, яка забезпечує інтерактивні та візуально привабливі діаграми.

5) Система контролю версій та деплоймент:

- Код прошивки, серверної та клієнтської частин зберігається в репозиторії на GitHub.
- Автоматизація збірки та розгортання забезпечується через GitHub Actions, що дозволяє швидко оновлювати серверну та клієнтську частини.
- Для керування залежностями використовується pip (для Python).

2.3 Дані

Дані є ключовим елементом системи, оскільки вони формують основу для аналізу погодних умов і прийняття рішень фермерами. Усі дані зберігаються та передаються у форматі JSON для забезпечення сумісності між компонентами системи.

1) Дані про погоду:

- Параметр даних: Параметр погодних даних (наприклад, CO2)
- Значення параметра: Числові значення вимірювань (температура в °C, вологість у %, тиск у гПа, опади в мм).
- Час заміру: Мітка часу у форматі ISO 8601 (наприклад, 2025-05-28T19:07:00Z), що дозволяє точно відстежувати зміни параметрів.

2) Дані про налаштування:

- Час оновлення даних: Частота, з якою метеостанція зчитує та передає дані (наприклад, кожні 10 хвилин, 1 година).
- Порогові значення: Граничні значення для температури, вологості, тиску тощо, які налаштовуються користувачем через веб-інтерфейс. Наприклад, сповіщення надсилається, якщо температура перевищує 30°C або вологість падає нижче 40%.

3) Дані про історію вимірювань:

- Зберігаються в MongoDB для подальшого аналізу. Дозволяють користувачам переглядати погодні тренди за день, тиждень або місяць.

2.4 Людські ресурси

Людські ресурси відіграють важливу роль у розробці, підтримці та просуванні системи. Вони включають різні категорії фахівців і користувачів, кожен із яких виконує свої завдання.

1) Користувачі:

Власники метеостанцій (агрономи): Основна цільова аудиторія, яка використовує веб-інтерфейс для моніторингу погодних умов, налаштування порогових значень і отримання сповіщень. Фермери можуть планувати сільськогосподарські роботи (полив, внесення добрив, захист від заморозків) на основі отриманих даних.

2) Розробники:

- Front-end розробник: Створює та підтримує веб-інтерфейс. Відповідає за інтерактивність, адаптивність і зручність інтерфейсу для користувачів.
- Back-end розробник: Розробляє серверну логіку на Flask, інтегрує базу даних MongoDB і забезпечує стабільну обробку запитів від метеостанції та клієнтської частини.
- Embedded розробник: Програмує мікроконтролер ESP32, забезпечує коректну роботу сенсорів і передачу даних. Відповідає за енергоефективність і надійність прошивки.

3) Адміністратори:

- Маркетологи: Розробляють маркетингові стратегії для просування метеостанції серед фермерів, проводять аналіз ринку та конкурентів, створюють рекламні кампанії.

- Менеджери продажів: Консультують клієнтів щодо можливостей системи, обробляють замовлення, вирішують питання щодо гарантійного обслуговування та повернення.

4) Інші:

- Тестувальники: Проводять тестування апаратного та програмного забезпечення, перевіряють стабільність роботи метеостанції, сервера та веб-інтерфейсу. Виявляють і документують помилки, забезпечуючи високу якість продукту.
- Технічна підтримка: Надає консультації користувачам, допомагає з налаштуванням метеостанції та вирішенням технічних проблем.

2.5 Організаційні процеси

Для ефективної роботи системи необхідна чітка організація процесів, які включають:

- Розробка та тестування: Використання методології Agile для швидкої ітеративної розробки, що дозволяє оперативно вносити зміни на основі зворотного зв’язку від користувачів.
- Технічна підтримка: Організація служби підтримки через email, чат або телефон для оперативного вирішення проблем користувачів.
- Оновлення системи: Регулярне оновлення прошивки метеостанції та програмного забезпечення сервера через OTA та GitHub Actions.
- Маркетинг і продажі: Проведення рекламних кампаній, демонстрація системи на сільськогосподарських виставках, співпраця з аграрними кооперативами для розширення аудиторії.

ПРОЄКТУВАННЯ ТА РОЗРОБКА

3. Використання системи

3.1 Діаграма випадків використання

Діаграма випадків використання ілюструє взаємодію між користувачами та СПМ з ключовими дійовими особами: Кінцеві користувачі (власники метеостанцій) та адміністратори.

Актори

Кінцевий користувач

- Переглядати погодні дані в реальному часі з персональної метеостанції.
- Отримувати доступ до архівних даних про погоду, записаних їхньою станцією.
- Налаштовувати станцію.

Адміністратор

- Надання кінцевим користувачам оновлення системи.

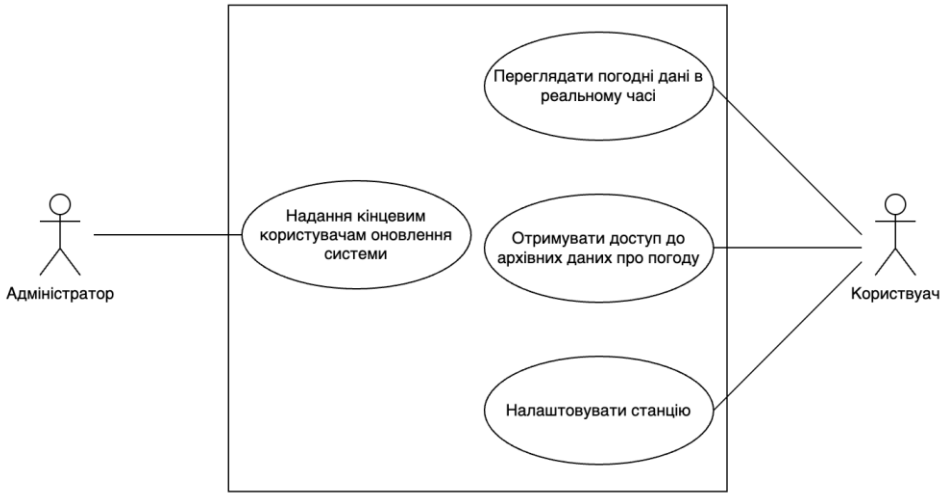


Рис. 1. Діаграма випадків використання

3.2 Опис випадків використання системи (Use Case Description)

Перегляд погодних даних у реальному часі

Актор: Кінцевий користувач

Опис: Користувач переглядає актуальні метеорологічні параметри (температуру, вологість, тиск, якість повітря тощо), що надходять безпосередньо з метеостанції. Дані відображаються у веб-інтерфейсі у вигляді текстових значень або графіків.

Результат: Користувач отримує оперативну інформацію про погодні умови у своїй локації.

Отримання доступу до архівних погодних даних

Актор: Кінцевий користувач

Опис: Користувач має змогу переглядати історичні дані за попередні дні або періоди. Система дозволяє вибрати конкретну дату або діапазон дат, і відображає відповідні погодні параметри у вигляді таблиць або графіків.

Результат: Користувач може аналізувати погодні тренди у своїй місцевості.

Налаштування граничних значень та отримання сповіщень

Актор: Кінцевий користувач

Опис: Користувач встановлює критичні значення температури, вологості тощо. У разі перевищення цих меж система автоматично надсилає сповіщення.

Результат: Користувач оперативно реагує на зміни погодних умов, що можуть зашкодити його врожаю або діяльності.

Налаштування метеостанції

Актор: Кінцевий користувач

Опис: Користувач може змінювати конфігурацію пристрою (інтервали вимірювань, типи датчиків, параметри підключення до мережі, мова інтерфейсу тощо) через веб-інтерфейс або мобільний додаток.

Результат: Метеостанція працює відповідно до індивідуальних потреб користувача.

Оновлення системи

Актор: Адміністратор

Опис: Адміністратор відповідає за оновлення прошивки мікроконтролера, серверного ПЗ та веб-інтерфейсу.

Результат: Система підтримується в актуальному стані, що забезпечує її стабільну та безпечну роботу.

3.3 Відображення випадків використання:

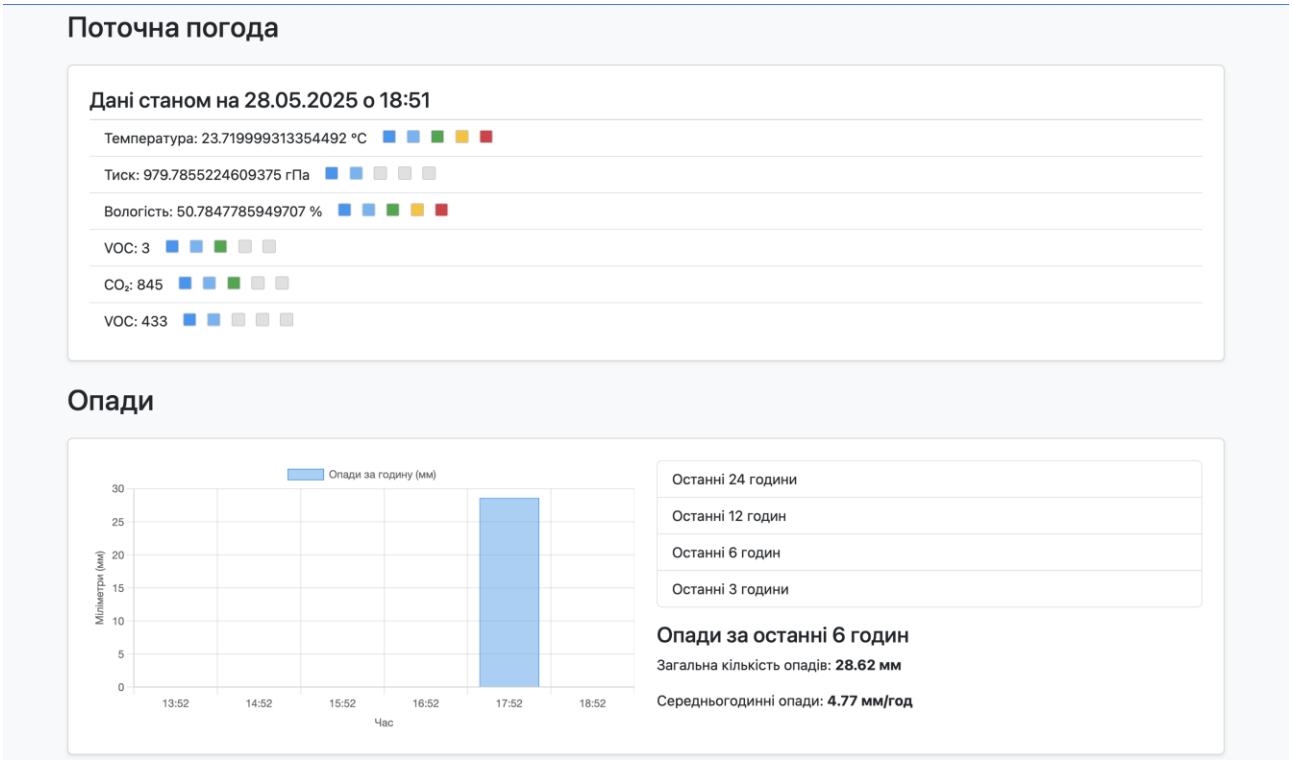


Рис. 2. Перегляд погодних даних у реальному часі

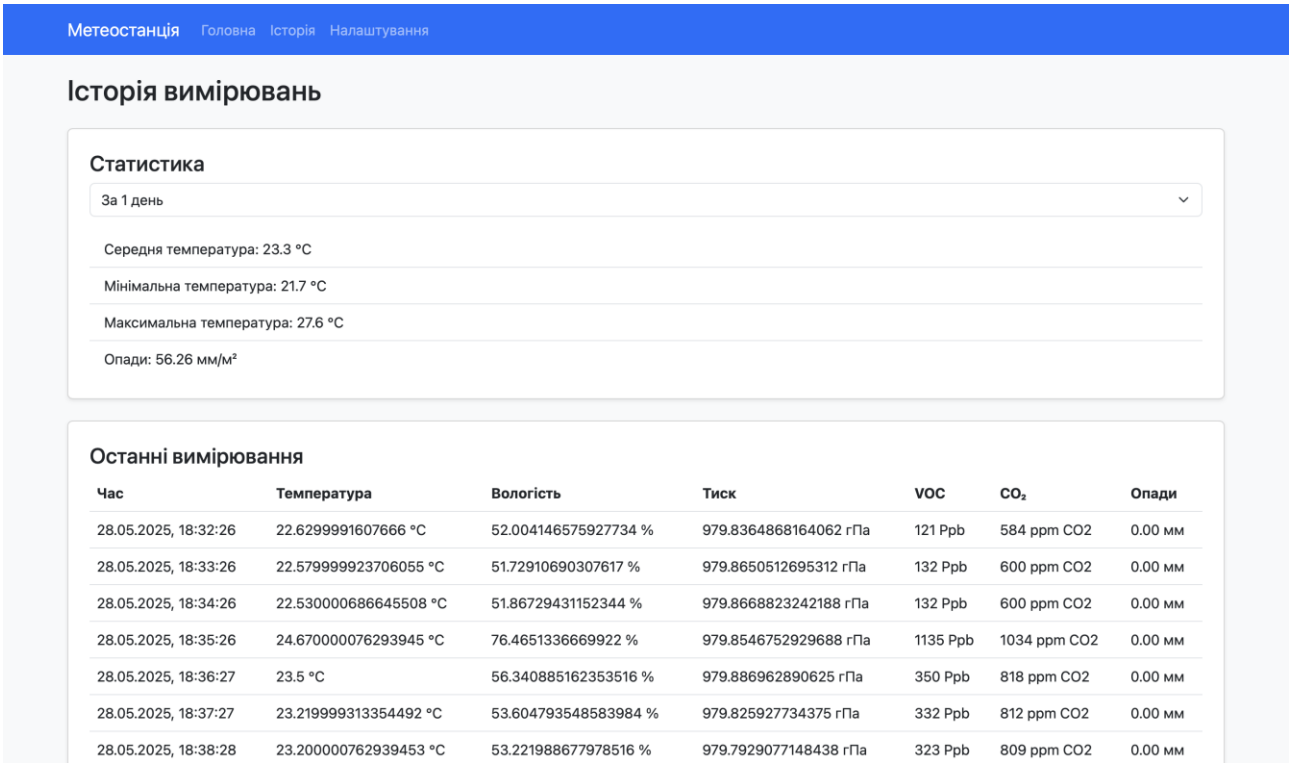


Рис. 3. Перегляд статистики та архівних погодних даних

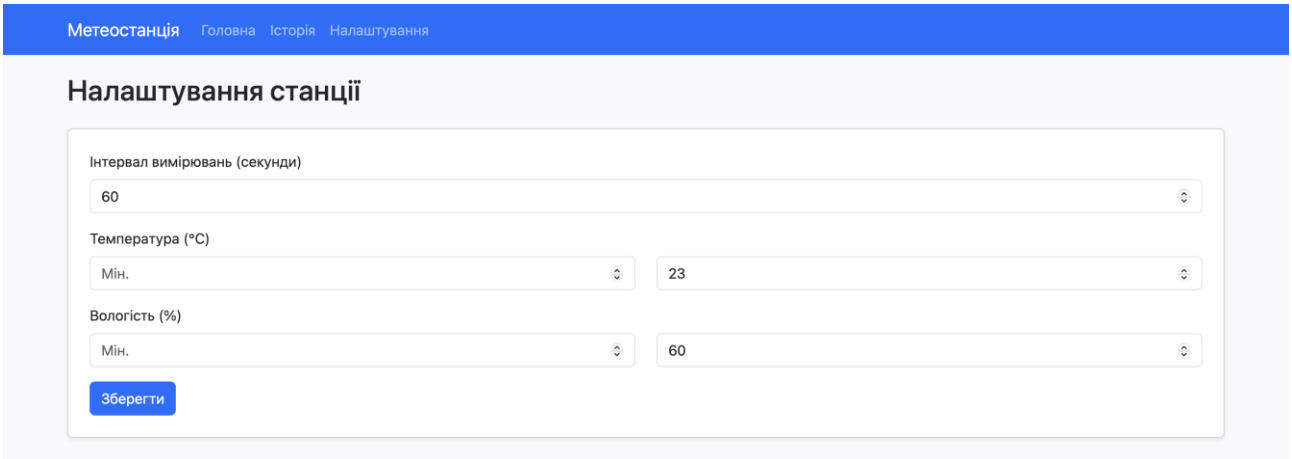


Рис. 4. Налаштування метеостанції

4. Блок-схема наскрізного процесу

Дана блок-схема ілюструє повний цикл взаємодії користувача з інформаційною системою метеостанції — від налаштування параметрів до отримання повідомлень та перегляду виміряних даних.

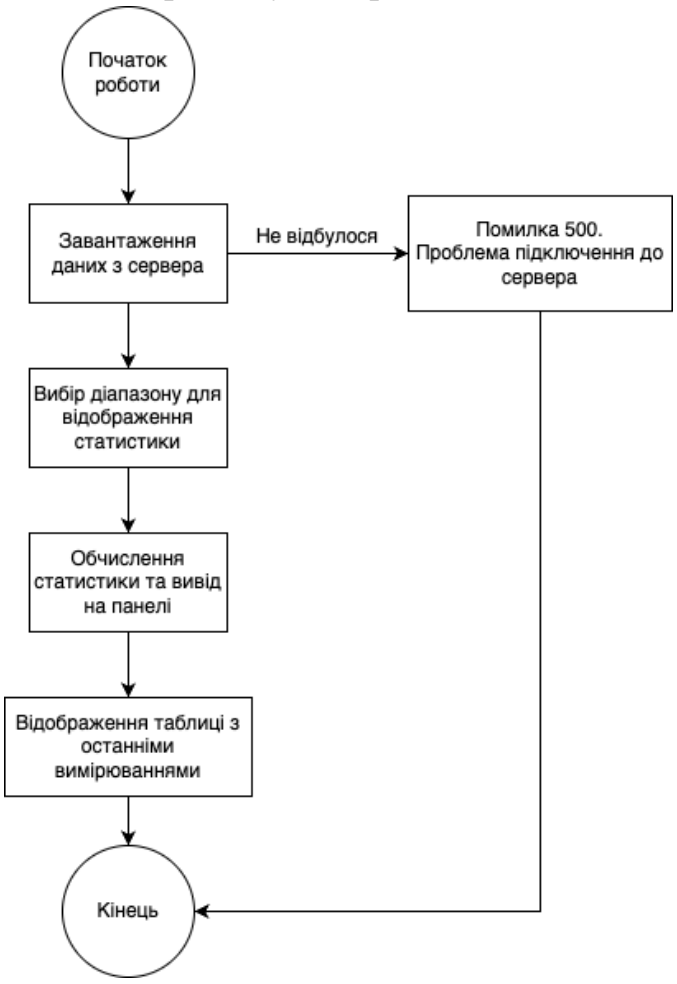


Рис. 5. Взаємодія із Головним вікном

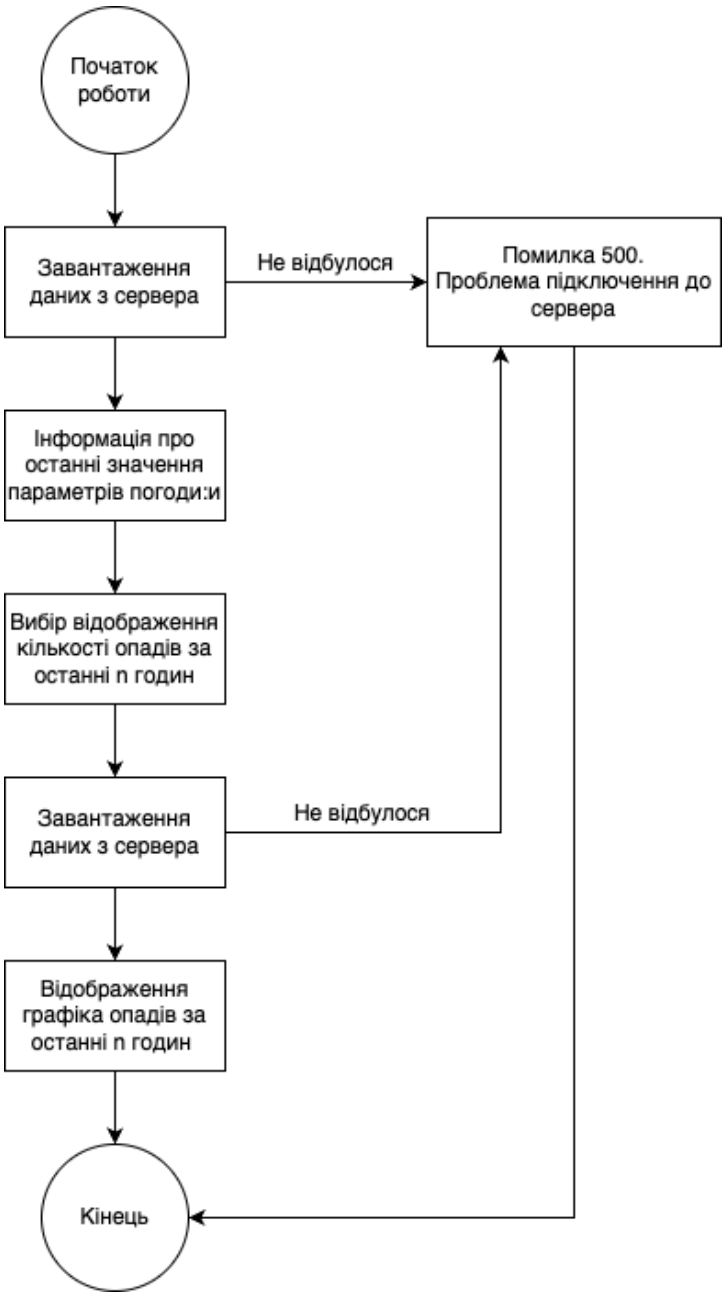


Рис. 6. Взаємодія з вікном Історія

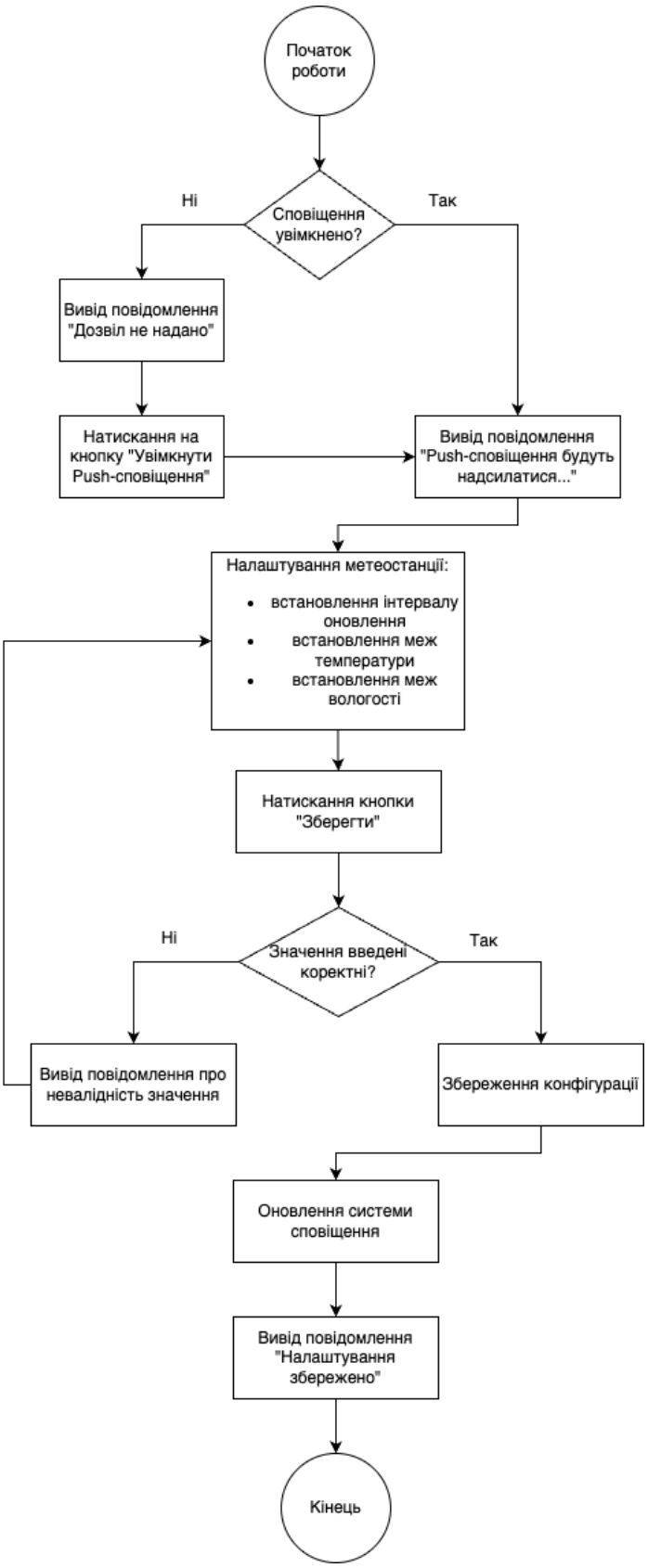


Рис. 7. Взаємодія з вікном Налаштування

5. Опис вимоги

5.1 Функціональні вимоги

Збір даних про погоду

ID	Вимога	Функціональність	Детальний опис
FR-001	Збір даних про погоду	Збір даних з датчиків	Система отримує дані в режимі реального часу (температура, вологість, якість повітря, опади) з датчиків персональної метеостанції користувача.
FR-002	Відображення погодних даних	Інформаційна панель користувача	Погодні дані відображаються на веб-інформаційній панелі з простими візуалізаціями (наприклад, цифрові дисплеї, графіки трендів), що оновлюються кожні 30 секунд.
FR-003	Сповіщення про погоду	Система сповіщень	Користувачі отримують сповіщення про значні погодні зміни (наприклад, дощ) в їхньому місцезнаходженні через push-сповіщення.

Управління даними

ID	Вимога	Функціональність	Детальний опис
FR-004	Зберігання даних про погоду	Реєстрація даних	Система зберігає погодні дані з кожної станції користувача з позначками часу для архівного доступу.
FR-005	Отримати архівні дані	Запит даних	Користувачі можуть отримати минулі погодні дані зі своєї станції за вказаний проміжок часу.

Управління метеостанцією

ID	Вимога	Функціональність	Детальний опис
FR-006	Конфігурація станції	Керування налаштуваннями станції	Користувачі можуть встановлювати інтервали збору даних та оновлювати інформацію про станцію за допомогою програми.

5.2 Нефункціональні вимоги

Зручність використання

- Мінімалістичний інтерфейс: Інтерфейс простий і цілеспрямований, з пріоритетом на відображення погодних даних для нетехнічних користувачів.
- Швидке введення в експлуатацію: Нові користувачі можуть налаштувати свою станцію та вивчити основні функції протягом 10 хвилин.
- Інтуїтивне управління: Елементи інтерфейсу розроблені для легкого доступу до погодних даних та налаштувань станції.

Продуктивність

- Швидке оновлення даних: Оновлення погодних даних протягом 1 секунди після зчитування даних з датчика через P2P-з’єднання.
- Низька затримка: Середній час відгуку на запити користувачів становить менше 0,5 секунди.

Сумісність

- Підтримка браузерів: Сумісність з Chrome, Firefox, Safari та Edge.

Зручність обслуговування

- Журналювання помилок: Журнали для проблем з P2P-з'єднанням, дій користувачів та системних помилок.
- Час безвідмовної роботи: Гарантований час безвідмовної роботи 99,5%, щомісячний час простою менше 4 годин.
- Резервне копіювання: Щотижневе автоматичне резервне копіювання користувацьких погодних даних та конфігурацій станцій.

5.3 Вимоги до зовнішнього інтерфейсу

Користувацькі інтерфейси

UI-1: Чистий інтерфейс, з акцентом на відображення погодних даних з використанням простої навігації та зрозумілих сповіщень.

Програмні інтерфейси

API-1: REST API з методами GET та POST для інтеграції з зовнішніми погодними API та сервісами сповіщень, використовуючи формат JSON.

Апаратні інтерфейси

HW-1: Підтримка персональних датчиків метеостанції (температура, вологість, дощ) через P2P Wi-Fi з'єднання з точкою доступу сервера за допомогою протокола HTTP.

HW-2: Розгортається на стандартних ноутбуках.

Інтерфейси зв'язку

COMM-1: Використовує HTTPS для безпечного обміну даними з SSL/TLS 1.3; підтримує MQTT через Wi-Fi для передачі даних P2P в режимі реального часу між персональними метеостанціями та точкою доступу сервера.

6. Проектування та розробка бази даних

6.1 Вибір бази даних

У рамках курсової роботи потрібно було розробити систему моніторингу за метеоданими, які будуть поступати через вбудовані системи прямо на сервер, а з нього на базу даних. Через нерівнозначність фінального вигляду даних, було вирішено використовувати нераляційну базу даних. Серед багатьох варіантів ми зійшлися на MongoDB, так як ми мали попередній досвід користування та там був безкоштовний план.

6.2 Структура бази даних

В MongoDB використовують систему колекцій, тобто кожна колекція це є окремий об’єкт, який переважно має свою json структуру. До колекцій належить:

Колекція	Опис
sensorData	Тут зберігається вся інформація яка надходить до нас від embedded системи, тобто від esp32. Серед них дані про навколишнє середовище, якість повітря, дощ і час запису.
config	Ця колекція слугує для налаштування різноманітних об’єктів системи, таких як інтервал оновлення та пороги значень температури і вологості.
time	Використовується для оновлення значення часу, яку esp32 запитує у сервера та повторює запит кожні 2 години.

6.3 Наповнення даними

Для базового функціоналу та тестування, для референсу ми наповнили базу даних такими json файлами:

sensorData:


```
{
  "_id": {
    "$oid": "6837583ebe6d22941bdf9d9e"
  },
```

```
  "timestamp": "2025-05-28 18:38:52",
  "environment": {
    "temperature": 24.139999389648438,
    "pressure": 979.2654418945312,
    "humidity": 49.85170364379883
  },
  "air_quality": {
    "aqi": 1,
    "tvoc": 31,
    "eco2": 416
  },
  "rainfall": {
    "tips": 57
  },
  "system": {
    "update_interval_ms": 60000
  }
}
```


config:


```
{
  "_id": {
    "$oid": "68264528e2ab668ca0f95496"
  },
  "system": {
    "update_interval_ms": 60000
  },
  "wifi": {
    "wifi_password": null,
    "wifi_uid": null
  },
  "humidity": {
    "min": null,
    "max": 70
  },
  "temperature": {
    "min": null,
    "max": 25
  }
```

}

time:

{
"unix_time": 1743350820,
"iso_datetime": "2025-05-28T19:07:00",
"timezone": "UTC"
}

7. Проектування та розробка endpoints

7.1 Вибір середовища REST API

При виборі середовища разробки endpoints ми спиралися на ефетивність та стабільність роботи сервера. В кінці кінців було обрано Flask – легкий та гнучкий мікрофреймворк на Python. Також було використано його розширення – flask_socketio, яке забезпечує підтримку WebSocket-з'єднання. Це дозволило реалізувати миттєву передачу алертів (повідомлень про вихід параметрів за межі) у реальному часі без потреби в періодичному опитуванні клієнтом сервера. Такий підхід значно покращив інформування користувача про важливі події.

7.2 Проектування API

У рамках реалізації програмного інтерфейсу прикладного програмування (API) було спроектовано набір RESTful endpoint для взаємодії клієнта з сервером. Основною метою API є отримання, збереження та аналітична обробка метеорологічних даних, а також керування конфігурацією порогових значень параметрів. API було реалізовано у вигляді Flask-додатку зі зручною маршрутизацією та чітким поділом логіки на модулі. Серед категорій endpoint було:

- /api/data (POST): Прийом даних від метеостанції.
- /api/data (GET): Отримання останніх даних.
- /api/data/stats (GET): Статистика за вказаний період.
- /api/data/rainfall (GET): Дані про опади за інтервал.
- /api/data/hourly_rainfall (GET): Погодинна статистика опадів.
- /api/config (GET/POST): Отримання та оновлення конфігурації.
- /time (GET): Отримання поточного часу сервера.

Статичні сторінки для веб-інтерфейсу доступні через маршрути /web, /web/history і /web/settings, які повертають HTML-шаблони, створені за допомогою Flask (render_template).

8. Проектування та розробка Embedded системи

8.1 Визначення вимог системи

Перш ніж почати розробку embedded частини я спочатку мав визначити вимоги до метеостанції, тобто дослідити існуючі варіанти, що вони пропонують, з чого складаються, яку інформацію надають. Було досліджено багато функціональних і нефункціональних вимог до самої метеостанції, серед яких не було імплементовано лише напрям і швидкість вітру, та деякі радіаційні дані, через обмеження бюджету.

Було зумовлено імплементувати такі функціональні та нефункціональні можливості системи:

- Функціональні:
    - Вимірювання температури, вологості та тиску;
    - Вимірювання основних хімічного складу, яке впливає на якість повітря;
    - Автоматизоване вимірювання кількості опадів за допомогою дощеміра;
    - Датчик WI-FI або блютуз для з’єднання до сервера.

- Нефункціональні вимоги:
    - Низьке енергоспоживання, для можливості роботи від акумулятора чи батарей;
    - Стабільність підключення до зовнішніх пристроїв;
    - Можливість збереження даних у пам’яті пристрою;
    - Швидкість роботи, щоб вчасно оновлювати дані відповідно до вимог користувача;
    - Масштабованість для майбутнього оновлення прошивки.

8.2 Компоненти системи

Після підсумку всіх вимог до системи настав час підійти до вибору компонентів системи, на яких буде вестися розробка. Серед основних компонентів стали:

- **ESP32**
    - Швидкий, перевірений та надійний пристрій;
    - Має вбудований Wifi і Bluetooth модуль;
    - Мав попередній досвід користування;
    - Має власну пам’ять, до якої розробник може зчитувати та записувати дані;

- Має можливість ввімкнути deep sleep режим, який змушує пристрій заснути, що тратить в рази менше електроенергії.

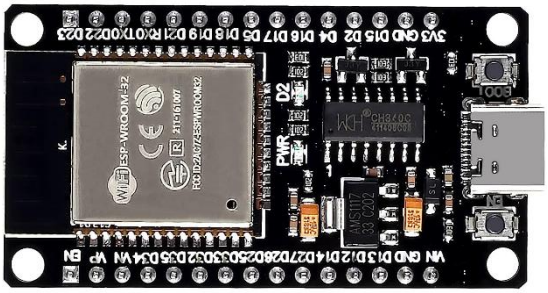


Рис. 8. ESP32 DevKit v1 30 Pins

- **BME280**
  - Складається з датчика BMP280, який відповідає за показ температури та атмосферного тиску;
  - Вміщає датчик DH11 для вимірювання вологості повітря;
  - Дешева ціна та попередній досвід використання.

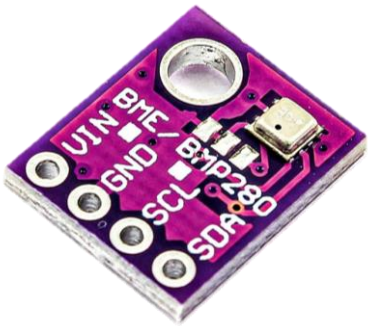


Рис. 9. Датчик BME280

- **ENS160**
  - Надає інформацію про вуглекислий газ CO2;
  - Загальну кількість органічних сполук в повітрі TVOC;
  - Загальну оцінку якості повітря AQI;
  - Відносно невелика ціна;
  - Працює як окремий пристрій, який сам нагрівається та обчислює змінні якості повітря.

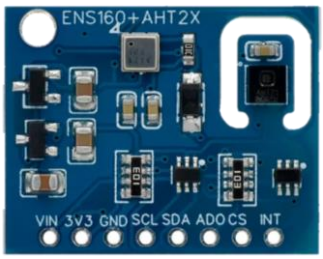


Рис. 10. Датчик ENS160

- **HALL-сенсор**
    - Потрібен для автоматизації дощового вимірювання.

Рис. 11. Сенсор Холла (HALL Sensor)

8.3 Автоматизація дощового вимірювання

Дощоміри поділяються на дві категорії: статичні та автоматизовані. До статичних належить наприклад проста пробірка, з позначками рівня води, які ти можем глянути скільки набралося води після дощу. Автоматизовані системи бувають різні. Дехто використовує рівень води як мірило, тобто вода стікає після досягання певного рівня, хтось використовує ваговий опадомір, а дехто використовує маячних механізм з ковшем, що перекидається при переповнені.

Серед доступних опцій було обрано варіант з ковшем. Сам механізм заключається в концентрації дощу в певній точці, тобто через лійку, яка переповняє так званий ковш, який перекидається через вагу.

Для кращої демонстрації представимо лійку з діаметром близько 9.5 см, або більше. В ту лійку на всю площадь падатиме дощ, який буде стікати всередину лійки. Під лійкою знаходиться простий менханізм, типу гойдалки з невеликим контейнером, який може вміщати певну кількість води. Кожен раз як на одну сторону гойдалки буде надходити достатньо води, він під вагою перекинеться на іншу сторону, цей момент ми назвемо tip, тобто перекид.

Нам потрібно почати з розрахунку кількості опадів, що представляється одним нахилом ковша. Для цього нам потрібно дві речі: радіус вхідного отвору лійки та кількість води, яка викликає переповнення чашки. Перший показник є постійним (r = 47.5 мм), а другий рівний v = 7 мл.

$$P_f = \pi r^2 = 7088.2184 mm^2$$

$$k = \frac{1000000}{P_f} = 141.0791$$

$$A = k * v = 0.9875 \frac{l}{m^2}$$

Де:

$P_f -$ Площа поперечного перерізу вхідного отвору воронки

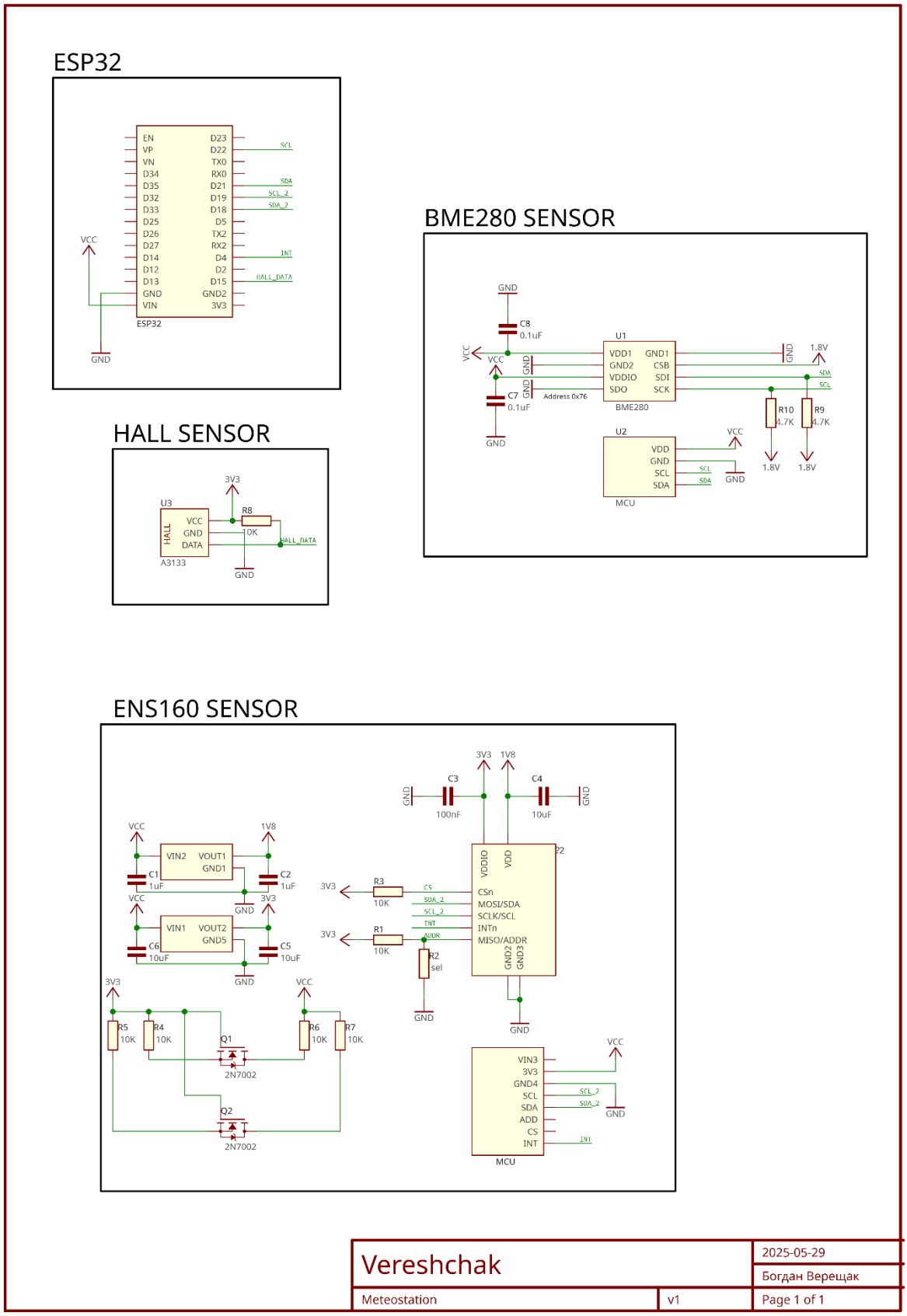
$k -$ відношення квадратних метрів до площі поверхні воронки

$A -$ об′єм одного tip тобто перекиду ковша

В результаті вияснилося що один перекид ковша дорівнює рівно 0.9875 мм на квадрат кубічний, тобто практично 1 літр на м^2.

Рис. 12. Віддрукований на 3D принтерів дощомір

8.4 Схематика



Vereshchak		2025-05-29
		Богдан Верещак
Meteostation	v1	Page 1 of 1

8.5 Підключення та електропроводка

Для нашого проекту я вирішив зробити прототип системи на breadboard, тобто без спаювання компонентів, хоча все одно прийшлось паяти ніжки для сенсорів. Загальне підключення відносно просте:

- BME 280 відповідні піни до живлення 3.3V та землі, та підключення до esp32 через I2C на піни SDA – D21, SCL – D22;
- INS160 відповідні піни до живлення 3.3V та землі, та підключення до esp32 через I2C на піни – SDA – D18, SCL – D19;
- HALL сенсор підключений до живлення і землі відповідно, та до D5 піна на esp32.
- Також тут є toggle схема для вкл/викл на інтегральній схемі timer 555 через транзистори та кнопку. Якщо живлення відбувається через breadboard, то кнопка відповідає за включення та виключення всього електричного кола.

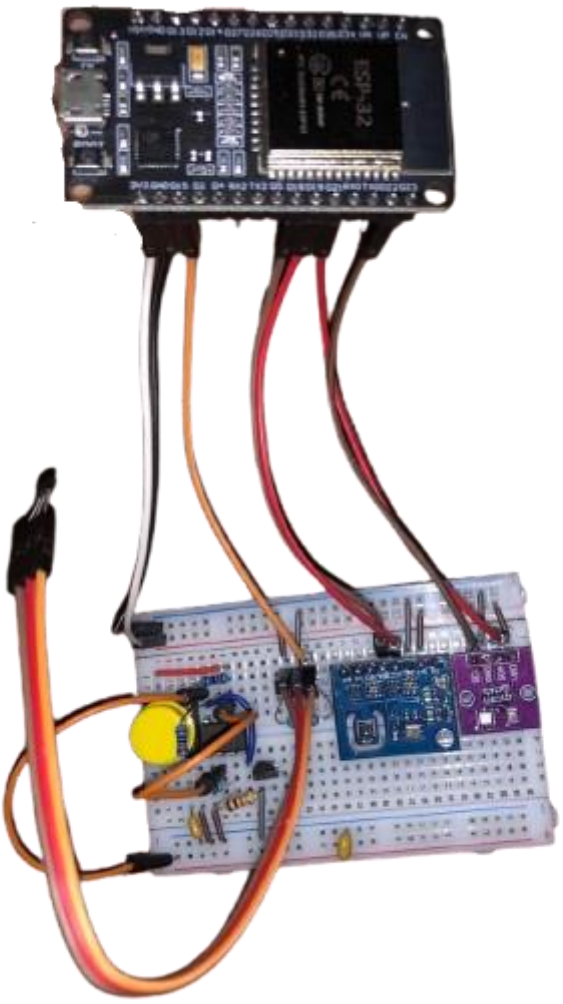


Рис. 13. Прототип зібраної метеостанції на breadboard

8.6 Вибір апаратної платформи

Під апаратної платформи розуміється IDE для написання прошивки на esp32. Серед виборів були Arduino IDE та VSCode. Незважаючи на простоту розробки на Arduino IDE, я все ж таки обрав VSCode через моє особисте побажання, та через великої різноманітності бібліотек та фреймворків, які можна було встановити на нього.

8.7 Вибір фреймворка розробки

Під час вибору фреймворка розробки виникло декілька труднощів. Через те що я відмовився від Arduino IDE, означало обрати більш складніший фреймворк, серед яких я познайомився з PlatformIO та ESP-IDF. Так як у мене не було жодного попереднього досвіду роботи з ними, я поклався на пораду учасника проекту, та обрав ESP-IDF.

Цей фреймворк був відносно хороший. З одної сторони він підтримував паралельний запуску процесів тобто FreeRTOS, мав бібліотеку для Wi-Fi з'єднань та HTTP-клієнта для зв'язку з сервером, cJSON для зручного формування json файлу і так далі. З іншої сторони будь яке підключення до сервісі через нього дуже важке, досить довго не міг налаштувати I2C з'єднання з датчиками. А що саме головніше, це те що не було готової бібліотеки для роботи з датчиками BME280 та ENS160 та HALL сенсором, що означало, що мені власноруч потрібно буде писати з нуля їхні бібліотеки по документації виробників. Також я не зміг вирішити проблему з'єднання по https, тому всі запроси здійснюються через http. В цілому вибір фреймворка непоганий, але я однозначно ускладнив собі життя вибравши його.

8.8 Програмна архітектура

Код програми складається з декількох частин. Наразі загальна структура прошивки виглядає так:

1. Ініціалізація та підтягнення бібліотек;
2. Ініціалізація глобальних змінних та структур даних;
3. Функція для підключення та керування зв'язком з wi-fi, який автоматично перепідключається до wi-fi при втраті зв'язку, або обриванні минулого підключення;
4. Всі функції HTTP запитів до серверу для отримання та надсилання json файлів;
5. Ініціалізація сенсорів та інтерфейсу I2C для з'єднання з датчиками та сенсором Холла;
6. Функції для оновлення часу пристрою через внутрішній таймер, та підтягнення кожні 2 години часу з сервера.

7. Відповідна функція створення json файлу з даними з дачиків для відправки на сервер;
8. Функції процесів;
9. Головна функція, в якій ми:
    a. Ініціалізовуємо сховище NVS, через яке отримуємо основні змінні, такі як теперішній час, інтервал оновлення;
    b. Виконуємо функції ініціалізації всього;
    c. Створюємо черги для датчиків та переривання, щоб реагувати на кожен перекид ковша через сенсор Холла;
    d. Створюємо наші процеси/потоки.

8.9 Наявні змінні системи

Система налічує такі глобальні змінні програми:

Основні змінні:

- WIFI_SSID, WIFI_PASS – назва та пароль мережі, яку роздає хост-пристрій.
- SERVER_URL – посилання на сервер.

Конфігураційні змінні:

- DEFAULT_UPDATE_INTERVAL_MS – інтервал відсилання даних на базу даних та оновлення значень датчиків в мілісекундах.
- RAIN_GAUGE_TIPPING_VOLUME_ML – скільки мм потрібно щоб перехилити ковш.
- FUNNEL_RADIUS_MM – радіус лійки.
- FUNNEL_AREA_MM2 – розрахована через попередній параметр радіусу площа лійки.
- MM_PER_TIP – об’єм одного tick перекидного ковша.

GPIO змінні:

- I2C_SCL_GPIO, I2C_SDA_GPIO – піни для підключення BMP280
- I2C_SCL_GPIO_2, I2C_SDA_GPIO_2 – піни для підключення ENS160
- HALL_SENSOR_GPIO – пін для сенсора Холла.

Глобальні змінні:

- hall_event_queue, sensor_data_queue – черги для передавання даних між потоками.
- tip_count, last_tip_time, rain_detected – змінні для дебаунсу та загального відслідковування параметрів дощу.

Пріоритети задач (FreeRTOS):

- WIFI_TASK_PRIORITY = 2 – задача підключення до Wi-Fi має середній пріоритет.
- SENSOR_TASK_PRIORITY = 4 – задача збору даних із сенсорів має високий пріоритет для зменшення затримок.
- TIME_TASK_PRIORITY = 3 – задача синхронізації часу має середній пріоритет.
- RAIN_TASK_PRIORITY = 5 – найвищий пріоритет, оскільки події з дощу можуть бути короткочасними і не мають втрачатися.

8.10 Основні структури даних

- time_tracker_t – структура даних, яка містить записаний unix час з сервера, один розрахований esp32, сформований дата-час для відправок json файлів та мютекс (для блокування доступу до записаного часу, при підтягування з сервера або оновлення з esp32).
- sensor_data_t – структура даних, які беруться з датчиків, та згодом формуються в json для відправки на сервер, а з нього на базу даних.

8.11 Процеси (потоки)

- wifi_task – ініціалізує та підключається до wifi мережі (в нашому випадку це мережа хоста), кожні 10 хвилин перевіряє підключення до мережі та оновлює в разі відстуності.
- time_updater_task – кожної секунди оновлює часові значення та для безпеки кожної години підтягує часові значення з сервера.
- sensor_task – ініціалізує сенсори bmp280 та ens160 та з заданим інтервалом зчитує дані з датчиків та із загальних параметрів дощу та відправляє чергою sensor_data_queue на потік data_task.
- data_task – спрацьовує в разі отримання даних від датчиків, формуючи json файл, який згодом відправляє його на сервер.
- rain_task – ініціалізовує HALL сенсор, та реагування на переривання на ньому, пізніше спрацьовує лише в разі в разі переривання (взагалом не дуже потрібний, але знадобиться при нарощуванні системи, наприклад для sleep mode).

8.11 Відправка та отримання даних з серверу

Однією з ключових функцій метеостанції є передача даних на сервер для подальшого аналізу та візуалізації. Це працює приблизно так:

1. Відправка даних:

Коли всі датчики зчитали інформацію (температура, тиск, вологість, якість повітря, кількість дощу), система формує JSON-пакет – структурований текстовий формат, який легко обробляти на сервері.

Що входить у JSON?

- Час (timestamp) – точний час вимірювання, синхронізований з сервером.
- Дані довкілля – температура, тиск, вологість.
- Якість повітря – AQI, TVOC, eCO2.
- Опади – кількість перекидань ковша (tips).

Потім цей JSON відправляється HTTP POST-запитом на сервер (наприклад, http://192.168.0.101:3000/api/data). Якщо Wi-Fi підключений, дані йдуть відразу. Якщо ні – можна було б зберігати їх у пам'яті і відправити пізніше (але в нашому випадку ESP32 просто чекає поки з'явиться мережа).

2. Отримання конфігурації з сервера

Метеостанція не лише відправляє дані, але й може отримувати налаштування:

- Інтервал оновлення (update_interval_ms) – як часто відправляти дані (наприклад, кожні 30 секунд або 5 хвилин).
- Параметри Wi-Fi – якщо змінився пароль або назва мережі, станція може оновитися без перепрошивки.

Це відбувається через HTTP GET-запит на /api/config. Сервер повертає JSON з новими налаштуваннями, і ESP32 їх застосовує.

3. Синхронізація часу

Щоб час у даних був точним, метеостанція періодично (кожні 2 години) запитує поточний час у сервера через /time. Отримує Unix-час (кількість секунд з 1970 року) і встановлює його в системі.

Коли сервер не відповідає, ESP:

- Використовує внутрішній таймер (на основі esp_timer_get_time()).
- Пробує знову через 5 секунд (але не більше 3 спроб).

8.12 Обробка даних

У даному розділі розглянемо процеси обробки даних, отриманих від різних сенсорів метеостанції, та їх подальшу підготовку до передачі на сервер.

Робота з BMP280 (температура, тиск, вологість)

Цей датчик – практично найважливіший, який постійно міряє навколишнє середовище. Робота з ним розроблена через мною створену бібліотеку bmp280.h Ось як це відбувається крок за кроком:

1. Ініціалізація:

- По I2C шині ми відправляємо спеціальні команди, які "будять" датчик і налаштовують його роботу.
- Встановлюємо режим високої точності, щоб отримувати максимально точні показники.

2. Зчитування даних:

- Кожні 30 секунд (або інший заданий інтервал) мікроконтролер запитує у BMP280 сирі дані.
- Датчик відповідає послідовністю байтів, які виглядають як набір чисел без явного сенсу.

3. Калібрування:

- Використовуючи спеціальні формули з технічної документації, ми перетворюємо ці байти:
- Температура: обчислюється в градусах Цельсія з точністю до десятих.
- Тиск: переводиться в гектопаскалі (гПа) – саме в цих одиницях дають прогнози погоди.
- Для вологості використовується додатковий датчик DHT11, який працює за схожим принципом.

Робота з ENS160 (якість повітря)

Сенсор ENS160 потребує особливої уваги через свою специфіку. Робота з ним розроблена через мною створену бібліотеку ens160.h і працює так:

1. Процедура ініціалізації:

- Обов'язковий період прогріву (1 хв при щоденній роботі, 1 год при першому вмиканні)
- Калібрування базової лінії (baseline) при першому запуску

2. Інтерпретація показників (більш детально в наступному пункті):

- AQI (1-5) - визначається за вбудованою таблицею відповідності
- TVOC (ppb) - розраховується на основі 5 газових компонентів
- eCO2 (ppm) - обчислюється за алгоритмом Sensirion

3. Перевірка достовірності:

- Аналіз статусного регістра сенсора
- Виявлення помилок калібрування

Робота з дощеміром (HALL-сенсор)

Система вимірювання опадів включає:

1. Апаратна частина:

- Механічний ковш з магнітом (об'єм 6 мл)
- HALL-сенсор з підтягом до живлення

2. Програмна обробка:

- Реалізація дебаунсу (ігнорування подій <100 мс)
- Підрахунок кількості перекидань (tip_count)

3. Збереження даних:

- Автоматичне збереження tip_count у NVS
- Відновлення після перезавантаження

Підготовка даних до передачі

Перед відправкою наших даних з датчиків на сервер, програма передбачає такі обов’язкові процедури:

1. Форматування даних:

- Створення структури sensor_data_t
- Додавання часових міток
- Перетворення у JSON-формат

2. Контроль якості:

- Перевірка меж допустимих значень
- Виявлення відсутніх даних
- Логування помилок

3. Тимчасове зберігання:

- Буферизація при відсутності зв'язку
- Пріоритезація даних для передачі
- Результатом обробки є структуровані дані, готові для аналізу та візуалізації на серверній частині системи.

8.13 Інтерпретування даних якості повітря

Датчик ENS160 дає три основні показники: AQI, TVOC і eCO2. Але потрібно розібрати що вони означають на практиці:

1. AQI (Індекс якості повітря)

Шкала від 1 (відмінно) до 5 (небезпечно):

#	Рейтинг	Гігієнічна оцінка	Рекомендація	Граничний рівень впливу
5	Unhealthy	Situation not acceptable	Use only if unavoidable. Intensified ventilation.	hours
4	Poor	Major objections	Intensified ventilation. Search for sources.	<1 month
3	Moderate	Some objections	Increased ventilation. Search for sources.	<12 months
2	Good	No relevant objections	Sufficient ventilation recommended	no limit
1	Excellent	No objections	Target	no limit

Табл. 2. Інтерпретація змінної AQI

2. eCO2 (еквівалент CO_2)

Показує концентрацію вуглекислого газу (в ppm):

eCO_2 / CO_2 (ppm)	Рейтинг	Коментар / Рекомендація
>1500	Bad	Heavily contaminated indoor air / Ventilation required
1000 - 1500	Poor	Contaminated indoor air / Ventilation recommended
800 - 1000	Fair	Optional ventilation
600 - 800	Good	Average
400 - 600	Excellent	Target

Табл. 3. Інтерпретація змінної CO2

3. TVOC (леткі органічні сполуки)

Загальна кількість легких органічний сполук, яких існує понад 5000 видів. Це всяка хімія в повітрі – фарби, лаки, чистячі засоби, випаровування з меблів. Вимірюється в ppb (частин на мільярд).

• Джерела: людські біоефлюенти (дихання, потовиділення) та будівельні матеріали (меблі, засоби побуту).

• Симптоми надмірного TVOC: подразнення очей, головний біль, сонливість, запаморочення — Sick Building Syndrome (SBS).

TVOC (ppb)	Якість повітря	Коментар
0 – 65	Відмінна	Дуже чисте повітря

65 – 220	Хороша	Нормальна якість повітря
220 – 660	Задовільна	Помірне забруднення, бажано провітрити
660 – 2200	Погана	Забруднене повітря, потрібна вентиляція
2200 – 5500	Дуже погана	Сильне забруднення, негайно провітрювати
>5500	Небезпечно	Можливі негативні наслідки для здоров’я

Табл. 4. Приблизне інтерпретування значення TVOC

ТЕСТУВАННЯ

Метою тестування є перевірка функціональності, продуктивності, безпеки та зручності використання інформаційної системи персональної метеостанції (СПМ). Тестування спрямоване на виявлення та усунення помилок, забезпечення відповідності системи вимогам і гарантування її стабільної роботи в реальних умовах.

Область застосування

Тестування охоплює наступні компоненти системи:

1. Збір даних із сенсорів (температура, вологість, тиск, якість повітря, опади).
2. Обробка та збереження даних у базі даних.
3. Користувацький інтерфейс (веб-додаток).
4. API для інтеграції із зовнішніми сервісами (прогноз погоди, аналітика).
5. Адміністративна панель (керування сенсорами, користувачами).
6. Нефункціональні характеристики (продуктивність, безпека, сумісність).

Посилання на нормативні документи

Тестування проводиться відповідно до:

• Специфікація вимог до СПМ.
• Функціональні вимоги (FR-001 – FR-006).
• Нефункціональні вимоги (usability, performance, security).
• Вимоги до інтерфейсів (UI, API).

9.1 Об’єкти тестування

Програмне забезпечення

- Модуль збору даних: отримання даних із сенсорів (температура, вологість, тиск тощо).
- Модуль обробки даних: фільтрація, нормалізація та збереження даних.
- Користувацький інтерфейс: відображення погодних даних, графіків, прогнозів.
- Адміністративна панель: керування сенсорами, налаштуваннями.

Інтерфейси

- Користувацький інтерфейс (UI): веб-додаток для перегляду даних.
- Програмний інтерфейс (API): REST API для обміну даними із сервісами.

База даних

- Нереляційна БД (MongoDB): збереження погодних даних, логів, налаштувань.

- Перевірка: цілісність даних, швидкість запитів.

Веб-сторінки та інтерфейси
- Головна сторінка: відображення поточних погодних даних.
- Сторінка історії: графіки та звіти за вибраний період.
- Адміністративна панель: керування системою.

9.2 Що тестується
Функціональність
1. Головна сторінка:
   - Перегляд поточних погодних даних.
2. Сторінка історії:
   - Аналіз історичних даних (графіки, звіти).
3. Адміністративна панель:
   - Налаштування сенсорів (калібрування, частота збору даних).

Нефункціональні характеристики
1. Безпека:
   - Шифрування даних (SSL/TLS).
   - Контроль доступу за ролями.
2. Продуктивність:
   - Час завантаження сторінок (< 2 секунди).
3. Сумісність:
   - Підтримка браузерів (Chrome, Firefox, Edge, Safari).

9.3 Що не тестується
- Фізична працездатність сенсорів (тестується на етапі виробництва).
- Аварійне відновлення системи після апаратних збоїв.

9.4 Підхід до тестування
Рівні тестування
1. Юніт-тестування: перевірка модулів збору та обробки даних.
2. Інтеграційне тестування: взаємодія між сенсорами, БД і API.
3. Системне тестування: перевірка системи в цілому.
4. Приймальне тестування: оцінка відповідності бізнес-вимогам.

Методи тестування

- Ручне тестування: перевірка UI/UX, сценаріїв користувача.
- Автоматизоване тестування: API (Postman), UI (Selenium).

Типи тестування

1. Функціональне тестування: збір даних, відображення графіків.
2. Нефункціональне тестування:
   - Навантажувальне тестування (JMeter).
   - Тестування безпеки (OWASP ZAP).
   - Тестування UI (адаптивність, доступність).

9.5 Критерії приймання та припинення тестування

Критерії успішного проходження

- Усі функціональні тести (збір даних, відображення, авторизація) пройдені.
- Критичні помилки усунені та перевірені.
- Час відгуку API < 1 секунди.
- UI відповідає стандартам WCAG.

Критерії провалу

- Неможливість збору даних із сенсорів.
- Час відповіді > 3 секунди.

9.6 Критерії входу та виходу

Критерії входу

- Реалізовані ключові модулі (збір даних, UI, API).
- Налаштоване тестове середовище (MongoDB, сервер).
- Підготовлені тест-кейси та сценарії.
- Відсутність багів у збірці.

Критерії виходу

- Усі функціональні тести пройдені.
- Критичні помилки виправлені.
- Проведено регресійне тестування.
- Система відповідає вимогам продуктивності.
- Користувачі підтвердили готовність системи (UAT).

9.7 Середовище тестування

Операційні системи та браузери

- ОС: Windows 10/11, macOS, Linux (Ubuntu).
- Браузери: Chrome, Firefox, Edge, Safari.

Пристрої

- Ноутбуки, ПК.

Мережа та БД

- Мережа: локальне тестування, 3G/4G.
- БД: MongoDB, Redis (кешування).

Інструменти

- Postman (API).
- Selenium, Cypress (UI).
- JMeter (навантаження).
- TestRail, Jira (документація, баги).

9.8 Ролі та відповідальність

- Менеджер тестування: планування, координація, звіти.
- Тестувальники: створення тест-кейсів, виконання тестів, баги.
- Розробники: виправлення дефектів, юніт-тести.
- Бізнес-аналітики: уточнення вимог.

9.9 Планування ресурсів

Людські ресурси

Роль	Кількість	Відповідальність
Менеджер тестування	1	Планування, координація, аналіз ризиків
Функціональні тестувальники	2	Ручне тестування
Тестувальники автоматизації	1	Автоматизація API і UI тестів

Технічні ресурси

– Інструменти: TestRail, Jira, Postman, JMeter, Cypress.

9.10 Ризики та пом’якшення

Технічні ризики

Ризик	Опис проблеми	Стратегії пом’якшення

Недоступність середовища	Збої серверів	Резервне середовище, моніторинг
Несумісність ПЗ	Конфлікти версій	Контейнеризація (Docker)
Проблеми продуктивності	Повільна робота при навантаженні	Навантажувальне тестування, кешування

Організаційні ризики

Ризик	Опис проблеми	Стратегії пом'якшення
Брак тестувальників	Сповільнення тестування	Автоматизація, планування
Затримки розробки	Вплив на тестування	Гнучкий графік
Нечіткі вимоги	Помилки через неоднозначність	Чітка документація

9.11 Розклад тестування

Розклад тестування визначає ключові етапи, терміни та відповідальних осіб для виконання тестування персональної метеостанції. Тестування включає перевірку функціональних і нефункціональних характеристик системи, таких як точність збору даних, стабільність роботи сенсорів, інтерфейс користувача.

Етап тестування	Опис
Планування тестування	Розробка тест-плану, створення тестових сценаріїв і тест-кейсів
Налаштування тестового середовища	Підготовка апаратного забезпечення (сенсори температури, вологості, тиску), серверів і бази даних
Юніт-тестування	Перевірка окремих модулів: сенсори, API, обробка даних
Інтеграційне тестування	Тестування взаємодії між сенсорами і базою даних
Системне тестування	Перевірка системи в цілому: збір даних, відображення в інтерфейсі, push-повідомлення
Приймальне тестування	Оцінка системи кінцевими користувачами (точність даних, зручність інтерфейсу)
Регресійне тестування	Повторна перевірка після виправлення помилок
Підготовка фінального звіту	Складання звіту про результати тестування, рекомендації

9.12 Артефакти та звітність

Нижче наведено основні артефакти, які створюються під час тестування персональної метеостанції, їхній зміст, відповідальних осіб і терміни створення.

Тест-план (Test Plan)

- Опис: Документ, що визначає стратегію, ресурси, середовище, розклад і ризики тестування.
- Коли створюється: На етапі планування (10.03.2025–15.03.2025).
- Відповідальний: Менеджер тестування.

Тест-кейси (Test Cases)

- Опис: Сценарії для перевірки функціональності, включаючи ID, опис, передумови, очікуваний і фактичний результати, статус.
- Коли створюються: До початку функціонального тестування (15.03.2025).
- Відповідальні: Тестувальники.

Тестові звіти (Test Reports)

- Опис: Щотижневі звіти про хід тестування, включаючи статистику тест-кейсів, дефектів і рекомендації.
- Коли створюються: Після ключових етапів тестування (щотижня з 21.03.2025).
- Відповідальні: Менеджер тестування, Тестувальники.

Звіти про дефекти (Bug Reports)

- Опис: Документація помилок з ID, описом, кроками відтворення, пріоритетом і статусом.
- Коли створюються: Під час тестування, оновлюються при зміні статусу.
- Відповідальні: Тестувальники.

Фінальний звіт про виконання тестування (Final Test Summary Report)

- Опис: Підсумковий документ з аналізом тестування, рівнем виконання, списком дефектів і оцінкою готовності.
- Коли створюється: Після завершення тестування (21.04.2025–23.04.2025).
- Відповідальний: Менеджер тестування.

Висновок

Під час створення проєкту командою було розроблено повноцінну систему персональної метеостанції, яка дозволяє кінцевим користувачам у зручному форматі переглядати погодні умови у режимі реального часу, а також аналізувати архівні дані. Система охоплює не лише вимірювання параметрів, але й збереження, обробку, представлення даних та зручне адміністрування.

Було створено пристрій на базі мікроконтролера ESP32, до якого під'єднано необхідні сенсори (температури, вологості, атмосферного тиску та інші). Програмну частину мікроконтролера реалізовано мовою програмування C++, що забезпечує стабільне зчитування та передачу даних.

Також розроблено веб-інтерфейс, що складається з клієнтської та серверної частин. Серверна частина працює з документоорієнтованою базою даних MongoDB, що дозволило ефективно організувати збереження великих обсягів погодної інформації. Комунікація між клієнтом і сервером здійснюється за допомогою REST API, що забезпечує масштабованість та розширюваність системи.

У процесі роботи над проєктом було визначено та сформульовано технічні, функціональні та нефункціональні вимоги до системи. Вимоги детально описані у відповідних пунктах документації. Також було розроблено матрицю покриття вимог і здійснено тестування основних функцій системи з фіксацією отриманих результатів.

На початковому етапі розробки було побудовано діаграми випадків використання (Use Case Diagram), які відобразили основні ролі користувачів та їхню взаємодію із системою. Це дозволило чітко визначити критичні функціональні компоненти системи та сконцентрувати розробку саме на тих частинах, які мають безпосередню цінність для користувача. Надалі, на основі аналізу цих діаграм, було створено блок-схеми наскрізних процесів, що дозволило детальніше вивчити логіку взаємодії всіх компонентів системи та виявити потенційні місця виникнення помилок.

Система передбачає фіксацію показників з періодичністю (наприклад, кожні 4 години), що дозволяє не лише переглядати поточні погодні умови, але й здійснювати історичний аналіз у вигляді графіків, що автоматично генеруються у веб-інтерфейсі.

Основний функціонал, що був реалізований у межах проєкту:

- **Миттєвий перегляд погодних умов** у реальному часі з будь-якої точки світу;
- **Гнучкий аналіз архівних даних** за різні часові періоди;
- **Налаштування граничних параметрів** (наприклад, температури), після перевищення яких користувач отримує сповіщення;

- **Можливість персоналізації системи** через веб-інтерфейс: зміна параметрів, частоти оновлення, одиниць вимірювання тощо;
- **Оновлення системи та підтримка з боку адміністратора**, що забезпечує довготривалу стабільну роботу пристрою та програмного забезпечення.

В результаті було виконано головну ціль проєкту - задоволення потреб кінцевого користувача в простому, надійному та доступному інструменті для моніторингу погодних умов. Завдяки використанню відкритих технологій та мінімальної кількості компонентів, систему може зібрати і використовувати будь-хто — навіть без спеціальної технічної підготовки. Таким чином, проєкт персональної метеостанції має не лише практичне, але й освітнє значення, дозволяючи людям краще розуміти навколишнє середовище та приймати зважені рішення на основі даних.