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

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «IoT-система візуалізації та хмарного моніторингу даних на базі платформи ThingSpeak»

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

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

_____ **Богдан ЩЕРБАК**
(підпис) Ім'я, ПРІЗВИЩЕ здобувача

Виконав: здобувач(ка) вищої освіти гр. КІД-42
Богдан ЩЕРБАК
Ім'я, ПРІЗВИЩЕ
Керівник: Борис КОНДРАТЮК
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти: бакалавр
Спеціальність: 123 Комп'ютерна інженерія
Освітньо-професійна програма: Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою _____
_____ Ім’я, ПРІЗВИЩЕ
«____» _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Щербак Богдан Вікторович
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: IoT-система візуалізації та хмарного моніторингу даних на базі платформи ThingSpeak,
керівник кваліфікаційної роботи Борис КОНДРАТЮК викладач
(Ім’я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від 20 0.2 2026 р. № _51_
2. Строк подання кваліфікаційної роботи «___» _____ 2026 р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література; хмарний сервіс ThingSpeak, протокол передачі даних HTTP/MQTT.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз сучасного стану та технологій інтернету речей (IoT)
2. Проєктування архітектури та вибір компонентної бази
3. Програмно-апаратна реалізація та візуалізація даних
5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «___» _____ 20__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір та аналіз літературних джерел	07.03-21.03.2026	Виконано
2	Розробка теоретичної частини	22.03-02.04.2026	Виконано
3	Проектування архітектури IoT-вузла та апаратної топології	05.0.4-13.04.2026	Виконано
4	Програмна реалізація та налаштування хмарного сервісу ThingSpeak	15.04–25.04.2026	Виконано
5	Експериментальне дослідження та тестування системи	27.04–30.04.2026	Виконано
6	Висновки за результатами аналізу	30.04–03.05.2026	Виконано
7	Розробка презентації	05.05–06.05.2026	Виконано
8	Передзахист	11.05–14.05.2026	Виконано
9	Здача роботи на кафедру	18.05–21.05.2026	Виконано

Здобувач(ка) вищої освіти _____ Богдан ЩЕРБАК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Борис КОНДРАТЮК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра (магістра): _____ стор., _____ рис., ____ табл., _37_ джерел.

Мета роботи - розробка програмно-апаратного комплексу для збору, аналізу та візуалізації даних про стан атмосферного повітря (температура, вологість, концентрація шкідливих газів) у реальному часі.

Об'єкт дослідження - процеси дистанційного моніторингу навколишнього середовища та візуалізації даних у системах Інтернету речей (IoT).

Предмет дослідження - методи побудови системи екологічного моніторингу повітря на базі мікроконтролера ESP32 та хмарного сервісу ThingSpeak.

Короткий зміст роботи: У кваліфікаційній роботі розроблено IoT-систему моніторингу якості повітря. Апаратна частина реалізована на базі мікроконтролера ESP32 з використанням цифрового датчика температури та вологості DHT22 та датчика якості повітря MQ-135. Для передачі даних використано протокол HTTP/MQTT та бездротову технологію Wi-Fi. Програмне забезпечення забезпечує обробку сигналів з датчиків та їх відправку до хмарної платформи ThingSpeak. На базі сервісу налаштовано систему візуалізації даних у вигляді графіків та віджетів, що дозволяє здійснювати віддалений аналіз стану довкілля.

Ключові слова: ІНТЕРНЕТ РЕЧЕЙ (IOT), ESP32, THINGSPEAK, МОНІТОРІНГ ПОВІТРЯ, MQ-135, DHT22, ХМАРНІ ТЕХНОЛОГІЇ, ВІЗУАЛІЗАЦІЯ ДАНИХ.

ЗМІСТ

ВСТУП 8

1. АНАЛІЗ СУЧАСНОГО СТАНУ ТА ТЕХНОЛОГІЙ ІНТЕРНЕТУ РЕЧЕЙ (IoT) 9

1.1 Концепція та архітектура систем Інтернету речей 9

1.2 Огляд протоколів передачі даних у системах моніторингу (HTTP vs MQTT). 11

1.3 Порівняльний аналіз хмарних платформ для візуалізації даних (Blynk, Adafruit IO, ThingSpeak)..……..……14

1.4. Архітектурні особливості та методи візуалізації даних у середовищі ThingSpeak 18

2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА ВИБІР КОМПОНЕНТНОЇ БАЗИ 20

2.1 Вибір та обґрунтування мікроконтролера 20

2.2. Вибір датчиків та інтерфейсів їх підключення 22

2.3 Розробка структурної схеми та віртуальне моделювання системи 25

2.4. Розрахунок енергоспоживання та автономності системи 30

3. ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ТА ВІЗУАЛІЗАЦІЯ ДАНИХ..... 34

3.1 Розробка алгоритму роботи програмного забезпечення контролера. 34

3.2. Конфігурування каналів та полів (Fields) у сервісі ThingSpeak..... 36

3.3. Налаштування візуалізації та аналізу даних за допомогою MATLAB Analysis 38

3.4. Комплексне тестування системи та оцінка точності отриманих даних..... 41

ВИСНОВКИ 44

ВСТУП

Розвиток сучасної індустрії та діджиталізація виробничих процесів нерозривно пов'язані з активним впровадженням концепції Інтернету речей (IoT). Сьогодні автоматизація збору даних стає базовою вимогою для підприємств, що прагнуть оптимізувати витрати та підвищити рівень безпеки праці. Залежно від масштабів об'єкта, IoT-рішення можуть варіюватися від простих систем локального спостереження до складних хмарних комплексів, що обробляють великі масиви даних (Big Data).

Особливої актуальності питання автоматизованого контролю набуває на малих та середніх підприємствах. Якщо великі корпорації мають ресурси для впровадження високовартісних систем моніторингу, то малий бізнес часто залишається без належного технічного захисту через складність налаштування та експлуатації професійного обладнання. У таких випадках відсутність оперативного контролю за параметрами повітря (концентрацією газів, рівнем вологості чи температурою) створює критичні ризики для здоров'я працівників та стабільності обладнання.

Використання мікроконтролерів сімейства ESP32 у поєднанні з хмарною платформою ThingSpeak пропонує збалансоване рішення цієї проблеми. На відміну від застарілих аналогових систем, такий підхід забезпечує мобільність, низьку енергозатратність та можливість віддаленого аналізу даних з будь-якої точки світу через протоколи HTTP та MQTT. Це дозволяє трансформувати розрізнені датчики у цілісну інформаційну систему, яка здатна вчасно сигналізувати про вихід параметрів за межі норми. У даній роботі досліджується процес проектування та практичної реалізації такого комплексу, що робить її важливою як з теоретичної, так і з практичної точки зору.

1. АНАЛІЗ СУЧАСНОГО СТАНУ ТА ТЕХНОЛОГІЙ ІНТЕРНЕТУ РЕЧЕЙ (IoT)

1.1 Концепція та архітектура систем Інтернету речей

Стрімкий розвиток мікроелектроніки та бездротових мереж зв'язку зумовив перехід від класичних ізольованих систем автоматизації до глобальної концепції Інтернету речей (Internet of Things, IoT). Згідно з міжнародним еталонним стандартом ISO/IEC 30141:2024, IoT визначається як інфраструктура взаємопов'язаних сутностей, людей, систем та інформаційних ресурсів, що взаємодіють через мережі для обробки даних фізичного та віртуального світів [2].

Для забезпечення ефективної взаємодії між апаратними компонентами та програмним забезпеченням системи IoT будуються за багаторівневими архітектурними моделями. У сучасній інженерній практиці базовою вважається трирівнева архітектура (Three-Layer Architecture), яка забезпечує оптимальний баланс між складністю реалізації та масштабованістю [4].

Відповідно до цієї моделі, розроблювана система моніторингу та візуалізації даних структурно поділяється на наступні логічні рівні:

1. *сенсорний рівень (Perception Layer).* Це фізичний рівень системи, головним завданням якого є взаємодія з навколишнім середовищем - збір інформації за допомогою сенсорів та первинна обробка аналогових чи цифрових сигналів [1]. У рамках даного проєкту цей рівень реалізований на базі мікроконтролера ESP32, який виконує зчитування показників з підключених периферійних модулів (наприклад, датчиків температури, вологості або якості повітря). На цьому етапі відбувається оцифрування фізичних величин для їх подальшої передачі;

2. *мережевий рівень (Network Layer).* Відповідає за безпечну та надійну маршрутизацію зібраних даних від кінцевих вузлів (сенсорів) до хмарних серверів або локальних шлюзів [3]. Оскільки мікроконтролер ESP32 має вбудований Wi-Fi трансивер, мережевий рівень реалізується шляхом прямого підключення

пристрою до локальної бездротової мережі стандарту IEEE 802.11 (Wi-Fi) з подальшим виходом у глобальну мережу Інтернет;

3. *прикладний рівень (Application Layer).* Найвищий рівень архітектури, який відповідає за кінцеву обробку даних, їх збереження, математичний аналіз та візуалізацію у формі, зрозумілій для кінцевого користувача [1]. У розроблюваній системі функції прикладного рівня делеговані спеціалізованій хмарній платформі моніторингу. Цей підхід дозволяє уникнути необхідності розгортання власного фізичного сервера та забезпечує цілодобовий доступ до графіків телеметрії з будь-якої точки світу [4].

Узагальнена схема взаємодії між цими архітектурними рівнями, яка демонструє шлях даних від фізичного датчика до кінцевого інтерфейсу користувача, наведена на (рис. 1.1).

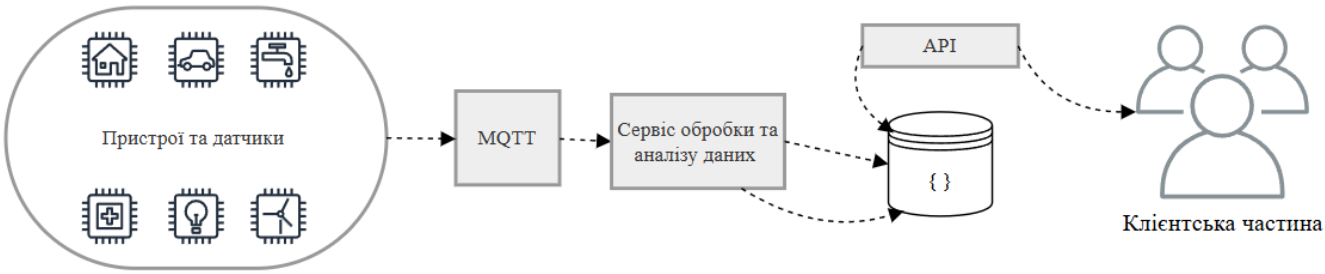


Рисунок. 1.1. Узагальнена архітектура системи моніторингу.

Використання стандартизованої трирівневої моделі [2, 3] дозволяє логічно розділити апаратну та програмну складові проєкту. Це гарантує, що при необхідності заміни типу датчика (Сенсорний рівень) або зміни інтерфейсу дашборду (Прикладний рівень), загальна логіка роботи системи не буде порушена.

1.2 Огляд протоколів передачі даних у системах моніторингу (HTTP vs MQTT).

Надійність, швидкодія та енергоефективність системи моніторингу навколишнього середовища критично залежать від вибору протоколу прикладного рівня (Application Layer) моделі OSI. Оскільки мікроконтролер

ESP32 має обмежені обчислювальні ресурси та лімітований запас енергії (у випадку автономного живлення), класичні підходи до веб-комунікацій потребують перегляду. Сьогодні в індустрії Інтернету речей домінують два основні протоколи: традиційний HTTP (HyperText Transfer Protocol) та спеціалізований MQTT (Message Queuing Telemetry Transport) [6, 9].

Особливості застосування протоколу HTTP (REST API) в IoT - Протокол HTTP історично створювався для передачі гіпертекстових документів і базується на архітектурі «клієнт-сервер» за моделлю «запит-відповідь» (Request-Response). У контексті систем моніторингу це означає, що пристрій повинен самостійно ініціювати відправку зібраних даних на сервер за допомогою методів POST або GET (архітектура REST).

Використання HTTP у системах на базі мікроконтролерів має низку суттєвих недоліків:

1. *висока надлишковість (Overhead).* Для передачі одного числового значення датчика (наприклад, температури 25.5°C, що займає кілька байт), HTTP вимагає формування громіздкого текстового заголовка (Header). Розмір такого заголовка може сягати від 200 до 1000 байт. Це призводить до нераціонального використання пропускної здатності каналу [8];

2. *проблема встановлення з'єднання.* Для кожного відправлення даних HTTP ініціює нове TCP-з'єднання (Three-way handshake), що потребує часу та додаткових обчислень. Навіть при використанні механізму Keep-Alive, підтримка відкритого з'єднання є ресурсомісткою;

3. *синхронна природа.* Клієнт блокується в очікуванні відповіді від сервера. Якщо хмарна платформа (наприклад, ThingSpeak) відповідає із затримкою, процесор ESP32 простоює, витрачаючи енергію акумулятора [5].

Архітектура та переваги протоколу MQTT - На противагу HTTP, протокол MQTT був розроблений компанією IBM спеціально для мереж із низькою пропускною здатністю, високими затримками та пристроїв з обмеженими апаратними ресурсами. Він працює за принципом «видавець-підписник» (Publish-Subscribe) та вимагає наявності центрального сервера - MQTT-брокера [5].

У цій архітектурі датчик (ESP32) виступає як «видавець» (Publisher), який публікує дані у певний логічний канал, що називається «топіком» (Topic). Хмарна платформа або кінцевий користувач виступає як «підписник» (Subscriber), отримуючи дані з цього топіка.

Переваги MQTT для розроблюваної системи:

1. *мінімальний заголовок.* Фіксований заголовок пакета MQTT складає всього 2 байти, що робить його ідеальним для передачі телеметрії (малих порцій даних з датчиків MQ-135 та DHT22) [7];

2. *асинхронність.* Мікроконтролер відправляє дані брокеру і миттєво повертається до виконання основного коду (або переходить у режим сну). Брокер сам бере на себе задачу доставити дані підписникам [9];

3. *рівні якості обслуговування (QoS).* MQTT гарантує доставку інформації навіть у нестабільних Wi-Fi мережах завдяки вбудованим рівням QoS [8];

- *QoS 0 (At most once):* Повідомлення відправляється один раз, доставка не гарантується (найшвидший метод);
- *QoS 1 (At least once):* Гарантована доставка повідомлення хоча б один раз із підтвердженням;
- *QoS 2 (Exactly once):* Найвищий рівень, що гарантує доставку рівно один раз без дублювання.

4. *механізм "Остання воля" (Last Will and Testament, LWT).* Дозволяє системі автоматично сповістити адміністратора, якщо зв'язок із датчиком несподівано обірвався (наприклад, через зникнення живлення) [6].

Візуалізація взаємодії між видавцем (сенсорним вузлом) та підписниками через центральний MQTT-брокер наведена на (рис. 1.2). Як видно зі схеми, брокер забезпечує повну ізоляцію відправника та отримувача даних, що дозволяє масштабувати систему без зміни конфігурації кінцевого пристрою.

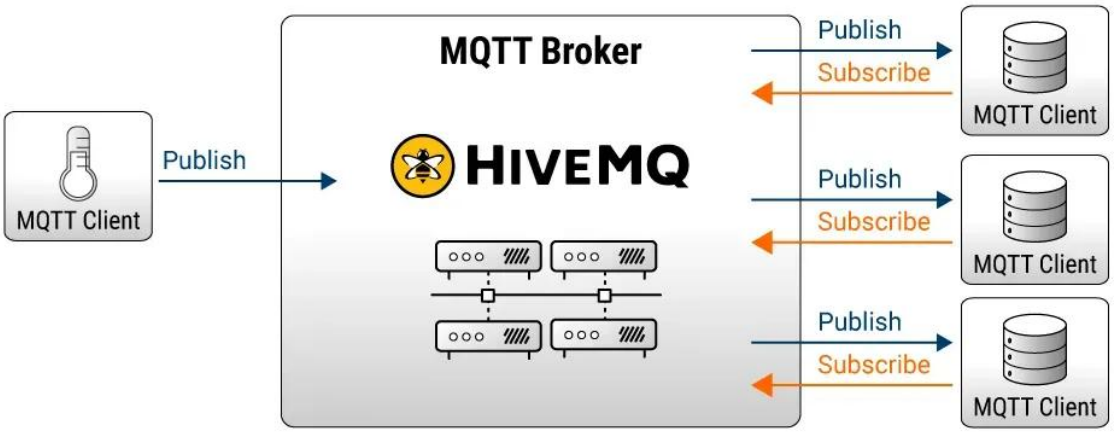


Рисунок. 1.2. Принцип роботи протоколу MQTT [8]

Порівняльний аналіз та вибір для платформи - ThingSpeakХмарний сервіс ThingSpeak підтримує обидва протоколи доступу [5]. Однак, як показують незалежні наукові дослідження, при масовій відправці невеликих пакетів даних (що є типовим для моніторингу повітря), пропускна здатність MQTT є значно вищою [7].

Для систематизації наведених даних розроблено порівняльну таблицю.

Таблиця 1.1 - Порівняльна характеристика протоколів HTTP та MQTT

Характеристика	**HTTP (REST API)**	**MQTT**
Архітектурна модель	Клієнт - Сервер	Видавець - Брокер - Підписник

Продовження таблиці 1.1

Характеристика	**HTTP (REST API)**	**MQTT**
Розмір заголовка	Великий (сотні байт)	Мінімальний (від 2 байт)
Формат даних	Текст (ASCII)	Бінарний
Гарантія доставки	Не передбачена (залежить від TCP)	Вбудована (QoS 0, 1, 2)
Продуктивності	Швидкодія системи	Latency, throughput
Управління станом	(Session)Stateless (кожен запит незалежний)	Stateful (брокер пам'ятає клієнтів)
Енергоспоживання пристрою	Високе	Низьке

Враховуючи необхідність забезпечення стабільного потоку телеметрії та оптимізації навантаження на мережевий модуль ESP32, базовим протоколом передачі даних для розроблюваної системи обрано MQTT [5, 8]. Це рішення дозволить суттєво зменшити час активної роботи мікроконтролера в мережі Wi-Fi, що є критично важливим параметром при подальшому масштабуванні системи.

1.3 Порівняльний аналіз хмарних платформ для візуалізації даних (Blynk, Adafruit IO, ThingSpeak).

Прикладний рівень розроблюваної системи моніторингу вимагає наявності надійної хмарної інфраструктури для збору, збереження та візуалізації телеметричних даних. Оскільки самостійне розгортання та підтримка власного сервера (наприклад, на базі Raspberry Pi з Grafana) потребує значних часових та

апаратних ресурсів, доцільним є використання готових PaaS (Platform as a Service) рішень для Інтернету речей [13].

На сучасному ринку IoT найпопулярнішими платформами для прототипування та розгортання систем моніторингу є Blynk, Adafruit IO та ThingSpeak.

Платформа Blynk (версія 2.0) Blynk - це популярна платформа, яка історично була орієнтована на швидке створення мобільних інтерфейсів для керування мікроконтролерами. У новій версії (Blynk IoT / 2.0) розробники змістили фокус на комерційний сектор, пропонуючи потужний веб-дашборд та управління пристроями (Device Provisioning) [10].

1. *переваги*: Інтуїтивно зрозумілий конструктор мобільних додатків (drag-and-drop), підтримка великої кількості апаратних платформ, вбудована система сповіщень;

2. *недоліки*: Жорсткі обмеження безкоштовного тарифного плану (Free Plan). Згідно з офіційною документацією, безкоштовна версія обмежує кількість пристроїв (до 2), кількість віджетів на панелі та встановлює суворі ліміти на частоту відправки повідомлень (не більше 10-15 повідомлень на секунду) [14]. Крім того, інструменти для математичного аналізу накопичених даних є вкрай обмеженими.

Платформа Adafruit IOСервіс - розроблений відомим виробником електроніки Adafruit. Орієнтований переважно на ентузіастів та невеликі дослідницькі проєкти. Платформа має нативну підтримку протоколу MQTT та інтегрується з сервісом автоматизації IFTTT [12].

1. *переваги*: Дуже простий процес налаштування брокера MQTT, красиві та сучасні віджети для візуалізації, наявність готових бібліотек для ESP32;

2. *недоліки*: У безкоштовній версії (IO Free) дані зберігаються лише 30 днів, а частота оновлення (Data Rate) обмежена 30 запитами на хвилину. Для

системи безперервного моніторингу з високою дискретизацією вимірювань цього може бути недостатньо.



Рисунок. 1.3. Інтерфейс user платформ Blynk

Платформа ThingSpeak - ThingSpeak є відкритою IoT-платформою з підтримкою API, розробленою компанією MathWorks (творцями середовища MATLAB). Вона дозволяє збирати, зберігати, аналізувати та візуалізувати дані з датчиків у хмарі [11]. На відміну від конкурентів, ThingSpeak орієнтована саме на роботу з даними та їх математичну обробку, а не просто на віддалене керування реле чи світлодіодами.

1. *переваги*: Інтегрована підтримка MATLAB дозволяє виконувати складний аналіз даних (фільтрацію шумів, прогнозування трендів) безпосередньо на серверах платформи без завантаження на ПК користувача. Платформа має вбудований MQTT-брокер та підтримує створення публічних і приватних каналів [13];

2. *недоліки*: Безкоштовна ліцензія обмежує частоту оновлення одного каналу до 1 повідомлення на 15 секунд, що є оптимальним для моніторингу атмосферних показників (температури, вологості, якості повітря), оскільки ці параметри є інерційними і не змінюються миттєво.

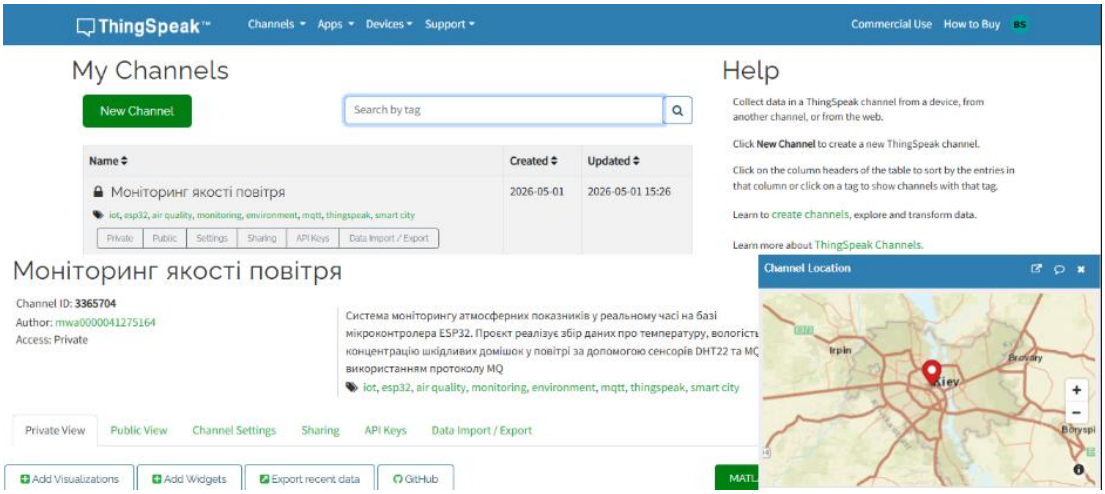


Рисунок. 1.4. Інтерфейс користувача ThingSpeak

Для систематизації наведених даних розроблено порівняльну таблицю.

Таблиця 1.2 - Порівняльна характеристика хмарних платформ для IoT

Критерій порівняння	**Blynk 2.0**	**Adafruit IO**	**ThingSpeak**
Основний фокус	Мобільне керування пристроями	Візуалізація та прототипування	Збір та математичний аналіз даних
Підтримка MQTT	Так (обмежено в базовій версії)	Так	Так (вбудований брокер)
Інструменти аналітики	Базові графіки	Базові графіки	Потужна аналітика через MATLAB
Збереження даних (Free)	Обмежене	30 днів	Без обмежень по часу (до 3 млн точо
Академічне застосування	Низьке	Середнє	Високе

На основі проведеного аналізу для реалізації даного проєкту обрано хмарну платформу ThingSpeak. Її архітектура ідеально відповідає завданням системи

моніторингу, оскільки забезпечує надійне збереження великих масивів історичних даних та дозволяє застосовувати алгоритми MATLAB для їх подальшої обробки [11, 13]. Крім того, наявність вбудованого MQTT-брокера спрощує програмну реалізацію на стороні мікроконтролера ESP32.

1.4. Архітектурні особливості та методи візуалізації даних у середовищі ThingSpeak

Після обґрунтування вибору платформи необхідно детально проаналізувати її внутрішню архітектуру та інструменти, які будуть використані для розробки системи. ThingSpeak базується на концепції «каналів» (Channels), кожен з яких дозволяє зберігати до 8 полів даних, поля для метаданих, а також інформацію про місцеперебування (широту, довготу та висоту над рівнем моря) [15].

Механізми обробки та візуалізації даних - Ключовою архітектурною особливістю обраної платформи є інтегрований інструментарій MATLAB Analysis та MATLAB Visualizations. Це дозволяє реалізувати складну математичну обробку телеметрії безпосередньо у хмарі [15, 16], що є критично важливим для систем моніторингу повітря, де дані часто потребують фільтрації від випадкових шумів датчиків.

Основні методи візуалізації, що надаються платформою, включають:

1. *часові графіки (Charts)*: Відображення динаміки зміни концентрації газів або температури в реальному часі;
2. *віджети (Gauges, Numeric Display)*: Для миттєвої індикації поточного стану атмосфери;
3. *картографічні сервіси*: Використання метаданих каналу для прив'язки показників до конкретних географічних координат, що дозволяє створювати мапи забруднення [15].

Безпека та доступ до даних - Архітектура ThingSpeak передбачає багаторівневу систему безпеки, що базується на використанні унікальних API-ключів (Write API Key та Read API Key) для кожного каналу. Це гарантує, що дані

з ESP32 можуть бути записані лише авторизованим пристроєм, а доступ до приватних каналів буде обмежений лише розробником системи. Крім того, платформа підтримує протокол MQTT з використанням автентифікації на рівні пристрою (MQTT Client ID, Username, Password), що забезпечує додатковий рівень захисту при передачі інформації через незахищені канали зв'язку [18].

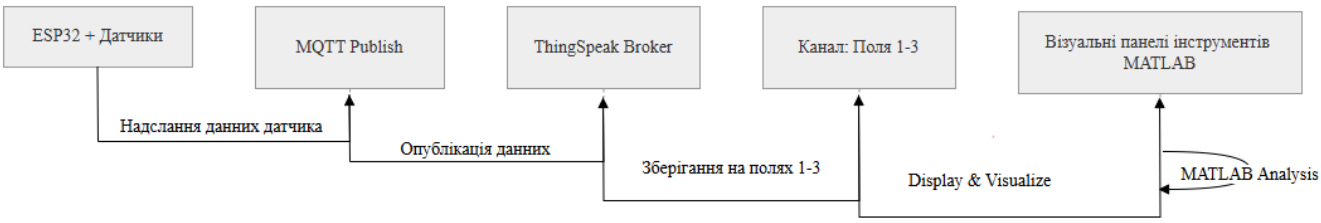


Рисунок. 1.5. Архітектура потоку даних проекту

Сучасні дослідження 2024-2025 років підтверджують, що використання хмарних середовищ з подібною архітектурою як проміжної ланки між сенсорним рівнем та користувачем дозволяє мінімізувати затримки в оновленні даних (latency) та забезпечує високу доступність сервісу (availability) [17, 18]. Це робить платформу надійним інструментом для побудови систем екологічного моніторингу, де важлива безперервність отримання інформації.

2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА ВИБІР КОМПОНЕНТНОЇ БАЗИ

2.1 Вибір та обґрунтування мікроконтролера

Для фізичної реалізації сенсорного вузла системи екологічного моніторингу необхідно обрати мікроконтролерну платформу, яка відповідає ключовим вимогам Інтернету речей: низьке енергоспоживання, наявність вбудованих модулів бездротового зв'язку та достатня обчислювальна потужність для первинної обробки даних перед їх відправкою у хмару [19].

На етапі проєктування було проведено порівняльний аналіз двох найпопулярніших IoT-платформ: ESP8266 та ESP32 від компанії Espressif Systems. На рисунку (2.1) представлені процесори мікроконтролерів ESP8266 і ESP32. Далі наведені повні основні технічні відмінності цих платформ, що базуються на аналізі апаратних характеристик, наведені у таблиці (2.1).



Рисунок. 2.1. MCU двох мікроконтрелерів [20]

Таблиця 2.1. Порівняльна характеристика платформ ESP8266 та ESP32

Мікроконтролер	**ESP8266**	**ESP32**
Процесор	L106 32-bit (Single-core)	Xtensa® Dual-Core 32-bit LX6
Тактова частота	80 МГц	160 - 240 МГц
Кількість ядер	1	2
Кількість GPIO	17	36

Продовження таблиці 2.1

Мікроконтролер	**ESP8266**	**ESP32**
АЦП (ADC)	1 канал (10-bit)	18 каналів (12-bit)
Bluetooth	Відсутній	Bluetooth 4.2 / BLE
Wi-Fi	802.11 b/g/n	802.11 b/g/n

Незважаючи на меншу вартість ESP8266, для розроблюваної системи було обрано мікроконтролер ESP32. Основною перевагою ESP32 є наявність двоядерного процесора Xtensa Dual-Core 32-bit LX6, що дозволяє розділити задачі: одне ядро відповідає за опитування датчиків, а друге за підтримку стабільного Wi-Fi з'єднання та обробку MQTT-пакетів [20]. Функціональна схема архітектури обраного мікроконтролера представлена на рис. (2.1).

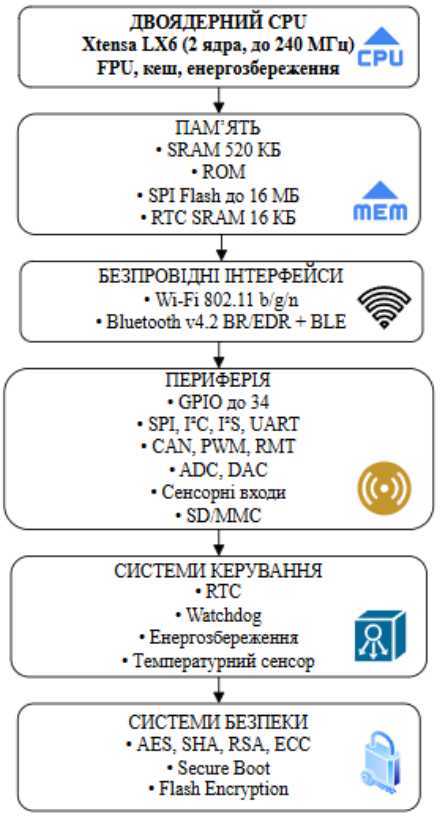


Рисунок. 2.2. Функціональна блок-схема архітектури ESP32

До додаткових технічних характеристик ESP32, які стали критичними для вибору платформи, належать:

1. *розширені можливості АЦП (аналого-цифрового перетворювача)*: ESP32 має 18 каналів 12-бітного АЦП. Це є принципово важливим для точного зчитування аналогових сигналів з датчика якості повітря MQ-135 [21];

2. *вбудований Wi-Fi (802.11 b/g/n)*: Дозволяє пристрою підключатися до локальної мережі та передавати дані в інтернет без використання додаткових шилдів або модулів [21];

3. *більша кількість GPIO-пінів*: Забезпечує можливість масштабування системи в майбутньому (додавання дисплея або додаткових метеорологічних сенсорів) [20].

Програмна сумісність та екосистема - Ще одним вагомим фактором на користь ESP32 є її повна сумісність із середовищем розробки Arduino IDE. Це суттєво спрощує процес програмування завдяки доступу до широкої бази готових бібліотек. Зокрема, для обраної нами хмарної архітектури існує офіційна бібліотека ThingSpeak від розробників MathWorks, яка оптимізована для роботи з сімейством мікроконтролерів ESP [22].

Використання стандартизованих бібліотек для зчитування показників з датчиків (наприклад, серії DHT) у поєднанні з офіційними API ThingSpeak гарантує високу стабільність передачі даних та мінімізує ризик втрати пакетів при тривалому моніторингу [22]. Таким чином, ESP32 є оптимальним апаратним ядром, яке повністю задовольняє вимоги спроєктованої у першому розділі архітектури.

2.2. Вибір датчиків та інтерфейсів їх підключення

Ефективність та метрологічна надійність системи екологічного моніторингу безпосередньо залежать від характеристик первинних вимірювальних перетворювачів. Для комплексного аналізу стану середовища у даній роботі обрано комбінацію цифрового сенсора метеорологічних показників DHT22 та напівпровідникового газоаналізатора MQ-135. Це дозволяє здійснювати

одночасний контроль термодинамічних параметрів та хімічного складу повітря [25, 26]. Основні технічні характеристики обраних компонентів зведено у таблиці (2.2).
Таблиця 2.2. Технічні характеристики сенсорів DHT22 та MQ-135 [25, 26]

Характеристика	**Датчик DHT22**	**Датчик MQ-135**
Призначення	Температура та вологість	Якість повітря (CO_2, NH_3, бензол)
Робоча напруга	3.3 В - 5.5 В	5.0 В ± 0.1 В
Інтерфейс передачі	Цифровий (Single-bus)	Аналоговий (ADC) / Цифровий
Діапазон вимірювань	-40...+80 °C; 0-100% RH	10 - 10000 ppm
Точність	±0.5 °C; ±2% RH	Залежить від калібрування
Струм споживання	1.5 мА (піковий)	150 мА (нагрівач)

Датчик температури та вологості DHT22 (AM2302) - Сенсор DHT22 було обрано через його значну перевагу у точності та роздільній здатності порівняно з базовою моделлю DHT11. Він складається з ємнісного датчика вологості та високоточного термістора, інтегрованих з 8-бітним мікроконтролером для обробки сигналів [25].

Передача даних до ESP32 здійснюється через спеціалізований цифровий інтерфейс (Single-bus data format). Протокол передбачає передачу пакету з 40 біт, що включає 16 біт даних вологості, 16 біт даних температури та 8-бітну контрольну суму для верифікації цілісності отриманої інформації [24, 25].

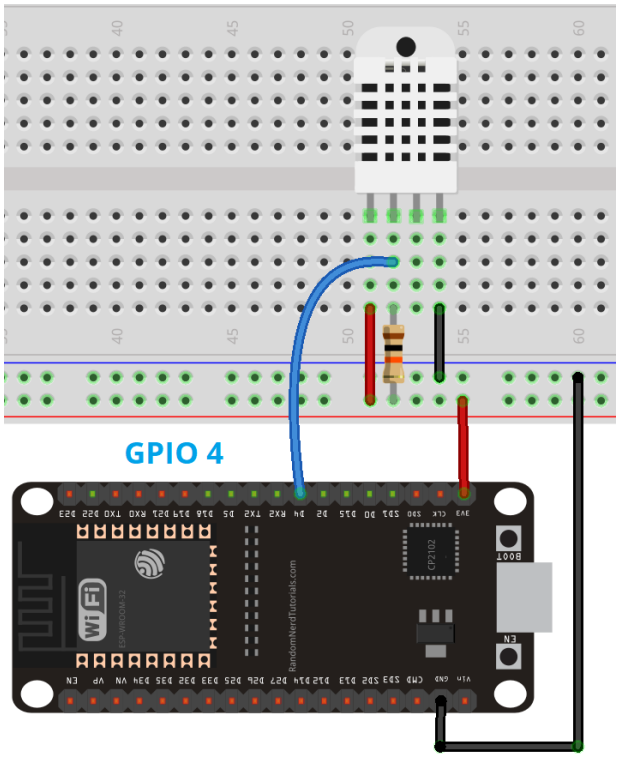


Рис. 2.3. Структура 40-бітного пакету даних протоколу DHT22 [24]

Для забезпечення стабільності сигналу та запобігання помилкам зчитування ("Failed to read from DHT sensor"), лінія даних потребує обов’язкового підтягування до шини живлення 3.3 В через резистор номіналом 4.7-10 кОм [24].

Датчик якості повітря MQ-135 - Для моніторингу концентрації шкідливих домішок використовується напівпровідниковий сенсор серії MQ. Його робота базується на зміні опору шару діоксиду олова (SnO_2) при взаємодії з молекулами адсорбованих газів [26]. Датчик демонструє високу чутливість до аміаку, оксидів азоту, алкоголю, диму та вуглекислого газу [23, 26].

Для отримання кількісних показників у системі використовується аналоговий вихід (AOUT). Отримана напруга зчитується 12-бітним АЦП мікроконтролера ESP32, що дозволяє диференціювати рівні забруднення з високою дискретністю [23]. Обов’язковою умовою експлуатації MQ-135 є попередній прогрів (preheat) внутрішнього нагрівального елемента протягом не менше 24 годин перед першим вимірюванням, що необхідно для стабілізації хімічного стану сенсора [26].

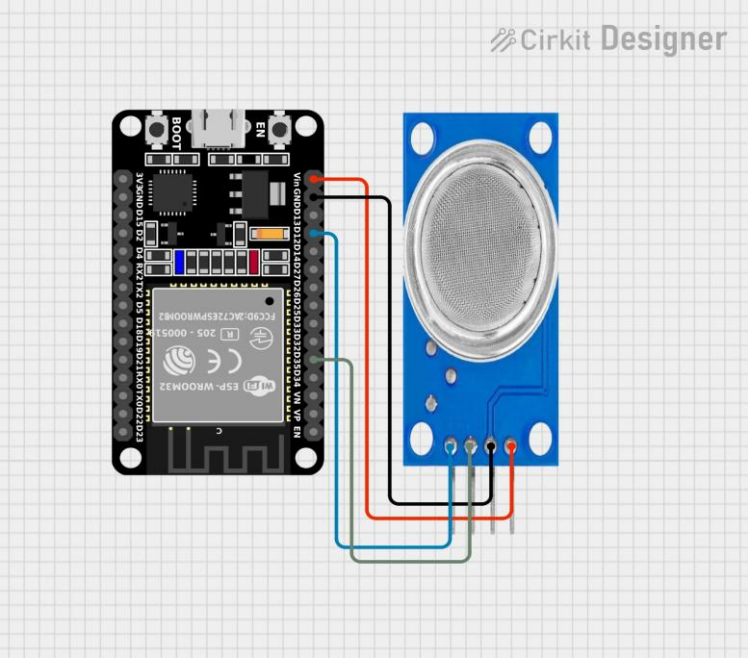


Рис. 2.4 Конструктивна схема та принцип підключення сенсора MQ-135 [23]

Узгодження апаратних інтерфейсів - Критичним аспектом інтеграції є різниця робочих напруг: MQ-135 потребує стабільних 5 В для нагрівача, тоді як входи АЦП ESP32 розраховані на максимальну напругу 3.3 В [23, 26]. Для захисту мікроконтролера в схемі передбачено використання резистивного дільника напруги на виході сенсора MQ-135, що адаптує діапазон сигналу 0-5 В до безпечного рівня 0-3.1 В [23]. Інтеграція Li-Po акумулятора ємністю 1000 мАг та модуля контролера заряду TP4056 у загальну схему пристрою (докладна схема підключення у Додатку Б) дозволила реалізувати функцію джерела безперебійного живлення всієї системи, організовано через спільну шину Vin мікроконтролера, що забезпечує необхідну потужність для нагрівального елемента газоаналізатора [26].

2.3 Розробка структурної схеми та віртуальне моделювання системи

На основі аналізу апаратної бази, проведеного у попередніх підрозділах, розроблено загальну структурну схему системи екологічного моніторингу.

Структурна схема є фундаментальним етапом проєктування, оскільки вона визначає основні функціональні вузли пристрою, апаратні інтерфейси їх взаємодії та загальну логіку руху інформаційних потоків - від зчитування фізичних величин до відображення аналітичних графіків у хмарному середовищі [28].

Архітектурна модель системи Типова архітектура розроблюваної системи базується на класичній багаторівневій (трирівневій) моделі Інтернету речей, яка є стандартом для систем екологічного моніторингу [27]. Згідно з цією концепцією, систему логічно розділено на три взаємопов'язані рівні:

1. *рівень сприйняття (Perception Layer)*: відповідає за фізичну взаємодію з навколишнім середовищем. Він включає сенсори DHT22 та MQ-135, які перетворюють неелектричні величини (температуру, вологість, концентрацію газів) в електричні сигнали;

2. *мережевий рівень (Network Layer)*: забезпечується мікроконтролером ESP32, який виконує роль інтелектуального крайового вузла (Edge Node) та бездротового шлюзу. Його завдання - агрегація даних та їх безпечна передача локальною мережею в інтернет [28, 29];

3. *прикладний рівень (Application Layer)*: реалізований на базі хмарної платформи ThingSpeak, де відбувається накопичення масивів даних, їх математичний аналіз та графічна візуалізація для кінцевого користувача [27].

Функціональні блоки та їх взаємодія Апаратна реалізація системи (рис. 2.5) складається з наступних ключових модулів, об'єднаних єдиною інформаційною шиною мікроконтролера:

1. *блок первинних перетворювачів (Блок сенсорів)*: складається з підсистеми метеоданих та підсистеми аналізу газів. Цифровий пакет даних від датчика DHT22 надходить безпосередньо на порт загального призначення (GPIO) мікроконтролера через однопровідну шину. Натомість, аналоговий сигнал від датчика MQ-135 вимагає обов'язкового проходження через апаратний вузол

узгодження рівнів напруги (понижувальний дільник) перед тим, як потрапити на вхід аналого-цифрового перетворювача (АЦП) [28, 29];

2. *центральний блок обробки*: базується на обчислювальному ядрі ESP32. Використання двоядерної архітектури дозволяє реалізувати апаратний мультитаскінг: одне ядро (Core 1) виділяється виключно для точного в часі (Real-Time) опитування та первинної фільтрації шумів АЦП, тоді як друге ядро (Core 0) обслуговує стек протоколів сенсорів TCP/IP та підтримує Wi-Fi з'єднання [29];

3. *блок живлення*: є критично важливим елементом стабільності схеми. Оскільки компоненти вимагають різної напруги, блок живлення розділений на дві лінії. Лінія 5 В забезпечує необхідним струмом нагрівальний елемент сенсора MQ-135, а лінія 3.3 В (через вбудований лінійний стабілізатор) живить логічну частину ESP32 та сенсор DHT22 [28];

4. *блок передачі даних*: фізично інтегрований у кристал ESP32 (вбудований радіомодуль 2.4 ГГц). Цей блок інкапсулює зібрані дані з сенсорів у формат JSON та ініціює HTTP/MQTT сесію для відправки пакета (Payload) на сервери ThingSpeak за допомогою унікального API-ключа каналу [27, 28].

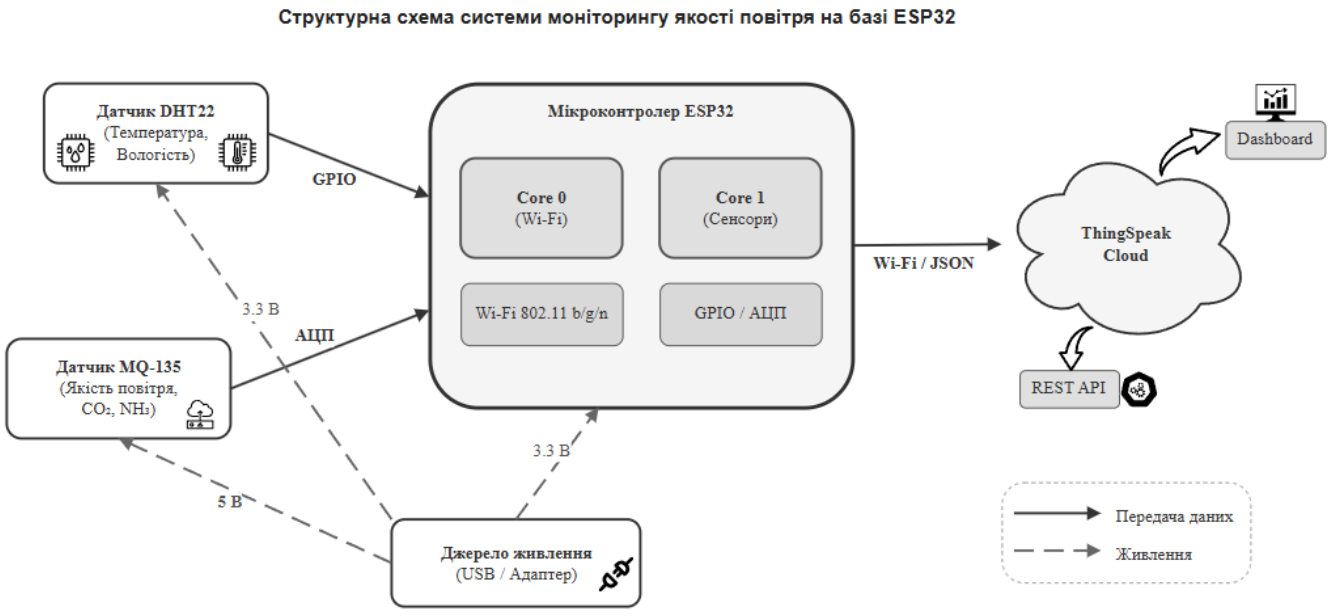


Рисунок 2.5 Структурна схема розроблюваної системи екологічного моніторингу

Логіка функціонування структурної схеми передбачає циклічну роботу: мікроконтролер формує запити до сенсорів із заданим інтервалом (15 секунд для відповідності обмеженням платформи ThingSpeak), оцифровує отримані значення, формує мережевий пакет і відправляє його у хмару, після чого переходить у режим очікування для оптимізації ресурсів [29].

Для первинного тестування програмного забезпечення мікроконтролера ESP32 було обрано хмарний симулятор Wokwi. Цей інструмент забезпечує повноцінну емуляцію Wi-Fi стека, що дозволило перевірити логіку відправки даних на ThingSpeak та відлагодити алгоритми зчитування даних по шині 1-Wire без ризику пошкодження реальних компонентів.

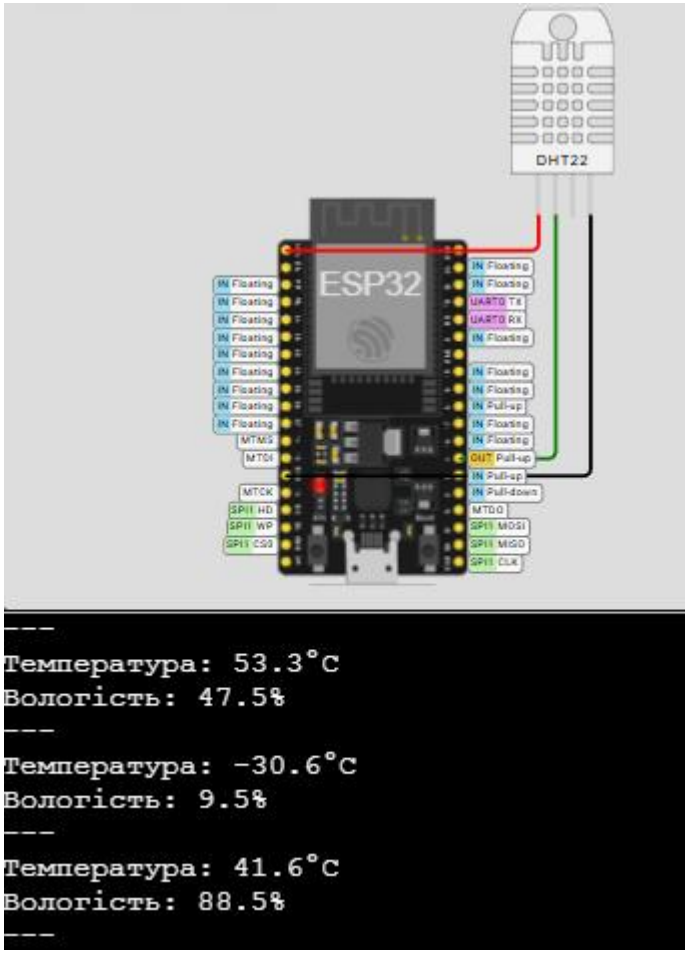


Рисунок. 2.6 Віртуальна симуляція системи у середовищі Wokwi

Після успішної симуляції логіки розроблено розширену апаратну топологію пристрою. Для створення наочної електричної принципової схеми (blueprint) та планування підсистеми живлення використано спеціалізоване середовище візуального проєктування апаратного забезпечення.

Завдяки візуалізації у цьому середовищі було детально спроєктовано розгалужену систему автономного живлення вузла, що дозволило вирішити проблему різних рівнів робочої напруги компонентів. Як видно зі схеми (рис. 2.X), основою автономного живлення виступає портативний LiPo-акумулятор (3.7 В, 1000 мАг), підключений через модуль контролера заряду TP4056.

Оскільки мікроконтролер ESP32 DevKitC та датчик DHT22 здатні працювати від зниженої напруги, а для нагрівального елемента сенсора якості повітря MQ-135 необхідна стабільна напруга 5 В, у схему інтегровано підвищувальний DC-DC перетворювач MT3608 (Voltage Boost Converter). Він трансформує напругу батареї до необхідних 5 В виключно для живлення газового аналізатора.

Середовище проєктування дозволило чітко розмежувати шини передачі даних (Data, позначені суцільними лініями) та лінії живлення (Power, позначені пунктиром). Окрім сенсорного блоку, на схемі також передбачено актуатор - світлодіод стану (Status Indicator LED) зі струмообмежувальним резистором на 220 Ом, який підключено до вільного GPIO мікроконтролера для візуальної індикації режимів роботи системи (наприклад, статусу підключення до Wi-Fi).

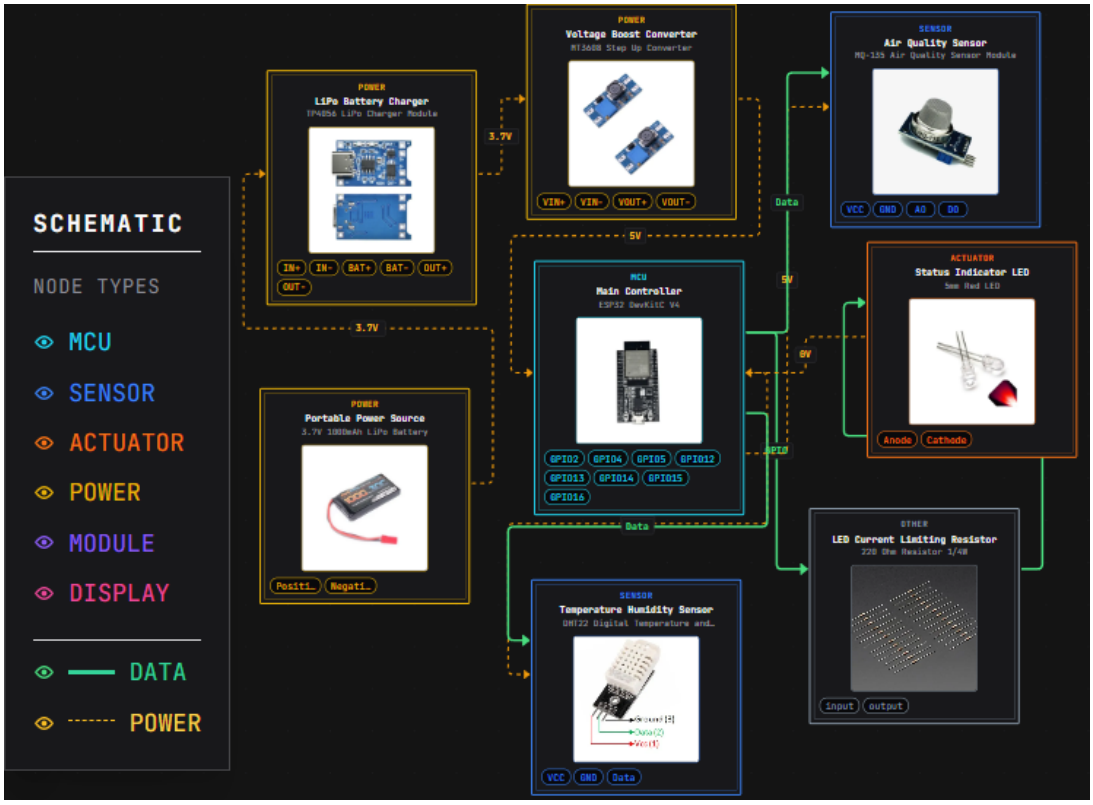


Рисунок. 2.7 Принципова електрична схема та 3D-модель плати в Bluepinta.am

Повна електрична принципова схема розробленого пристрою, що демонструє всі міжмодульні з'єднання та підсистему живлення, наведена у Додатку Б.

2.4. Розрахунок енергоспоживання та автономності системи

Критично важливим етапом проєктування будь-якого вузла Інтернету речей є розрахунок його енергетичного балансу. Оскільки розроблена архітектура передбачає можливість автономної роботи від портативного джерела живлення, необхідно провести точний розрахунок струмів споживання з урахуванням втрат на перетворення напруги [32].

Організація підсистеми живлення - Основою підсистеми автономного живлення є літій-полімерний (LiPo) акумулятор із номінальною напругою 3.7 В та ємністю 1000 мАг. Для його безпечної експлуатації та заряджання

використовується модуль на базі спеціалізованого контролера TP4056 (рис. 2.8). Цей модуль забезпечує лінійний алгоритм заряду постійним струмом/постійною напругою (CC/CV) та має вбудований захист від глибокого розряду (нижче 2.4 В), що є критичним для довговічності LiPo-елементів [30].

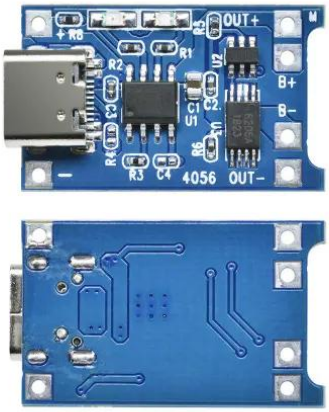


Рисунок 2.8 Модуль контролера заряду TP4056 із вбудованим захистом акумулятора

Головною інженерною проблемою системи є узгодження напруг. Мікроконтролер ESP32 та датчик DHT22 працюють від 3.3 В (забезпечується вбудованим LDO-стабілізатором на платі DevKit). Однак нагрівальний елемент сенсора якості повітря MQ-135 вимагає стабільної напруги 5 В. Для вирішення цього завдання в схему інтегровано підвищувальний DC-DC перетворювач MT3608 (рис. 2.9) [31].



Рисунок. 2.9 Підвищувальний DC-DC перетворювач напруги MT3608

Втрати на DC-DC перетворенні – на споживання нагрівача MQ-135 складає 150 мА при 5 В, що дорівнює споживаній потужності $P_{out} = 5\ \text{В} \cdot 150\ \text{мА} = 750\ \text{мВт}$.

Згідно з технічною документацією на MT3608, коефіцієнт корисної дії ((η)) при таких параметрах становить близько 90% (0.9) [31]. Розрахуємо реальний струм (I_{bat}), який перетворювач буде витягувати з акумулятора (3.7 В):

$$I_{bat} = \frac{P_{out}}{V_{bat} \cdot \eta} = \frac{750}{3.7 \cdot 0.9} \approx 225\ \text{мА}$$

Це означає, що тільки підтримка роботи газового датчика постійно споживає 225 мА від акумулятора.

Таблиця 2.3. Струми споживання компонентів системи

Компонент / Модуль	**Режим роботи**	**Споживаний струм, мА**
ESP32 DevKit	Wi-Fi Tx (передача)	240.0
ESP32 DevKit	Modem-sleep (очікування)	25.0
MQ-135 + MT3608	Постійний нагрів (від LiPo 3.7 В)	225.0
DHT22	Вимірювання / Очікування	1.5 / 0.05
Світлодіод (LED)	Активна індикація (з рез. 220 Ом)	10.0

Розрахунок динамічного споживання та часу роботи - Система працює за 15-секундним циклом:

1. *активна фаза (~2 с)*: ESP32 виходить зі сну, зчитує дані, засвічує LED індикатор, підключається до Wi-Fi та відправляє пакет даних. Сумарний струм: 240 + 225 + 1.5 + 10 = 476.5 мА;

2. *фаза очікування (~13 с)*: Радіомодуль Wi-Fi та LED вимикаються, мікроконтролер переходить у Modem-sleep. Сумарний струм: 25 + 225 + 0.05 + 0 ≈ 250 мА [32].

Середній струм споживання I_{avg} за повний цикл розраховується як:

$$I_{avg} = \frac{(476.5 \cdot 2) + (250 \cdot 13)}{15} = \frac{953 + 3250}{15} \approx 280 \text{ мА}$$

Виходячи з отриманого значення середнього струму, розрахуємо теоретичний час автономної роботи t_{work} від акумулятора ємністю 1000 мАг:

$$t_{work} = \frac{1000 \text{ мАг}}{280 \text{ мА}} \approx 3.57 \text{ годин}$$

3. ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ТА ВІЗУАЛІЗАЦІЯ ДАНИХ

3.1 Розробка алгоритму роботи програмного забезпечення контролера.

Для розробки програмного забезпечення мікроконтролера ESP32 було обрано інтегроване середовище розробки Arduino IDE. Вибір даного інструментарію обумовлений його кросплатформеністю, підтримкою великої кількості відкритих бібліотек для роботи з IoT-протоколами та наявністю потужного менеджера плат, що дозволяє легко інтегрувати ядро ESP32[32].

У процесі розробки було використано компілятор мови C++, що забезпечує високу швидкість виконання коду та ефективне управління пам'яттю мікроконтролера. Середовище Arduino IDE дозволило реалізувати механізми відлагодження через монітор послідовного порту (Serial Monitor), що було критично важливим для верифікації HTTP-запитів до хмарної платформи ThingSpeak та контролю стабільності Wi-Fi з'єднання.

IoT_Eco_Monitor_ESP32.ino

```
37    // Підключення до Wi-Fi
38    WiFi.begin(ssid, password);
39    Serial.print("Підключення до Wi-Fi");
40    while(WiFi.status() != WL_CONNECTED) {
41      delay(500);
42      Serial.print(".");
43    }
44
45    Serial.println("\n Підключено до Wi-Fi!");
46    digitalWrite(LED_PIN, HIGH); // Світлодіод світиться при успішному з'єднанні
47  }
48
49  void loop() {
50    // Перевірка статусу з'єднання
51    if(WiFi.status() == WL_CONNECTED) {
52      digitalWrite(LED_PIN, HIGH);
53
54      // 1. Зчитування даних з датчика DHT22
55      float humidity = dht.readHumidity();
56      float temperature = dht.readTemperature();
57
58      // 2. Зчитування даних з датчика MQ-135 (АЦП)
59      int airQualityRaw = analogRead(MQ135_PIN);
60
61      // Перевірка на помилки зчитування DHT22
62      if (isnan(humidity) || isnan(temperature)) {
63        Serial.println("Помилка зчитування DHT22!");
64        return;
```

Рисунок 3.1. Інтерфейс середовища розробки Arduino IDE з відкритим проєктом вузла моніторингу мікроклімату

Основні програмні компоненти та бібліотеки Для забезпечення коректної роботи апаратних вузлів та реалізації мережевої взаємодії, у програмі задіяні наступні бібліотеки:

1. *WiFi.h* - вбудована бібліотека ядра ESP32 для керування бездротовим модулем, підключення до локальної мережі роутера та роботи протоколу TCP/IP;

2. *HTTPClient.h* - забезпечує формування та відправку POST/GET запитів до REST API платформи ThingSpeak;

3. *DHT.h (від Adafruit)* - спеціалізована бібліотека для розшифровки цифрового сигналу з датчика DHT22 по протоколу 1-Wire.

Логіка роботи алгоритму Оскільки система проєктувалася з урахуванням можливості автономного живлення, програмний алгоритм побудовано за циклічним принципом із використанням режимів енергозбереження. Життєвий цикл програми (блок-схема алгоритму представлена на рис. 3.2) складається з наступних етапів:

1. *ініціалізація (Setup)*: При подачі живлення мікроконтролер виконує початкове налаштування. Відбувається ініціалізація послідовного порту (Serial) для відлагодження, конфігурування пінів GPIO (підключення сенсорів та LED-індикатора), а також запуск бібліотеки датчика DHT22;

2. *підключення до мережі*: Контролер активує модуль Wi-Fi та намагається підключитися до заданої точки доступу (SSID). У разі успішного з'єднання вмикається світлодіодний індикатор статусу;

3. *збір даних (Data Acquisition)*: Відбувається опитування сенсорів. Зчитується цифрове значення температури та вологості з DHT22, а також аналогове значення напруги з піна АЦП, до якого підключено датчик якості повітря MQ-135. Якщо зчитування з DHT22 повертає помилку (Not a Number), система робить повторний запит;

4. *формування та відправка пакета*: Зібрані сирі дані конвертуються в необхідний формат. За допомогою API-ключа (Write API Key) мікроконтролер формує HTTP-запит (URL-рядок), у який підставляє значення датчиків у відповідні поля (Field 1, Field 2, Field 3), та ініціює з'єднання з сервером api.thingspeak.com;

5. *режим очікування (Delay/Sleep)*: Після отримання від сервера коду успішної відповіді (HTTP 200), контролер вимикає LED-індикатор та переходить у режим очікування на 15 секунд. Цей таймаут є обов'язковим обмеженням безкоштовної версії платформи ThingSpeak для запобігання перевантаженню їхніх серверів [34]. Після завершення таймауту цикл повторюється з кроку 3.

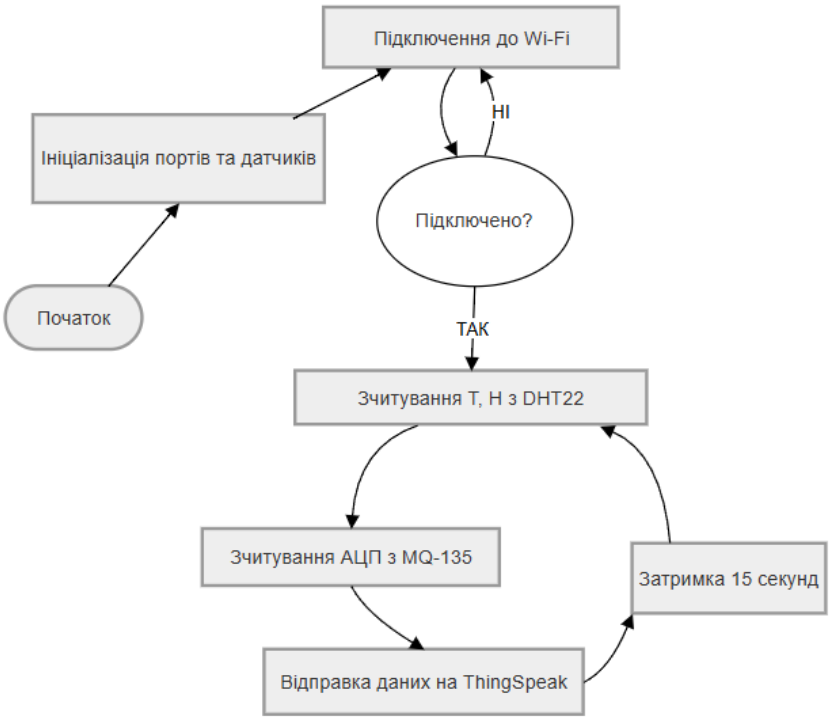


Рисунок 3.2. Блок-схема алгоритму роботи програмного забезпечення мікроконтролера

3.2. Конфігурування каналів та полів (Fields) у сервісі ThingSpeak

Для збору, довгострокового зберігання та подальшого аналізу телеметричних даних, отриманих з мікроконтролера, у проєкті використано

спеціалізовану хмарну платформу ThingSpeak від компанії MathWorks. Цей сервіс є галузевим стандартом для прототипування IoT-систем завдяки вбудованій підтримці протоколу HTTP (REST API) та тісній інтеграції з аналітичним середовищем MATLAB [35].

Створення інформаційного каналу Базовою структурною одиницею архітектури ThingSpeak є «Канал» (Channel). Для розроблюваної системи екологічного моніторингу було зареєстровано та налаштовано новий ізольований канал. Канал виступає єдиною точкою прийому (ендпоінтом) для масиву даних, що генерується крайовим пристроєм (ESP32).

У налаштуваннях каналу (Channel Settings) було активовано та промарковано три незалежні змінні - поля (Fields), кожне з яких жорстко прив'язане до конкретної фізичної величини, що вимірюється сенсорами вузла:

1. *field 1 (Temperature, °C)*: призначене для запису температурних показників, отриманих від датчика DHT22;
2. *field 2 (Humidity, %)*: виділене для фіксації рівня відносної вологості повітря (DHT22);
3. *field 3 (Air Quality, Raw/V)*: конфігуроване для запису аналогових значень, що відповідають еквіваленту концентрації газів у повітрі (від сенсора MQ-135).

Такий суворий розподіл вхідного потоку даних по окремих полях є критично важливим етапом. Це не лише дозволяє коректно ізолювати візуалізацію кожної метрики на індивідуальних графіках, але й формує правильну структуру масиву для подальшої математичної обробки (рис. 3.3).

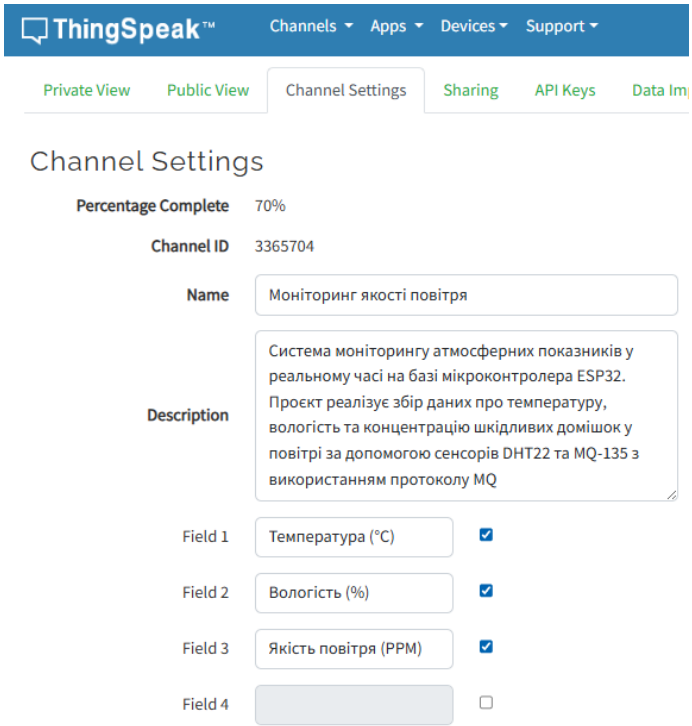


Рисунок 3.3. Конфігурування полів інформаційного каналу в панелі управління

Автентифікація та безпека доступу Для забезпечення цілісності даних та запобігання несанкціонованому запису сторонньої інформації у створений канал, платформа використовує систему унікальних криптографічних ключів (API Keys) [36].

Для авторизації мікроконтролера було згенеровано унікальний ключ доступу на запис - Write API Key. Цей 16-значний буквено-цифровий токен було жорстко закодовано (hardcoded) у C++ код прошивки пристрою. Під час кожної транзакції контролер формує HTTP-запит, в якому передає цей ключ разом із пакетом даних. Сервер ThingSpeak валідує ключ, і лише у разі збігу записує дані у відповідні поля каналу.

3.3. Налаштування візуалізації та аналізу даних за допомогою MATLAB Analysis

Збір сирих телеметричних даних є лише першим етапом моніторингу. Для того щоб система мала практичну цінність, масив числових значень необхідно

трансформувати у зручний для сприйняття візуальний формат. Платформа ThingSpeak надає вбудовані інструменти для побудови графіків часових рядів (Time-Series Charts) та панелей індикації (Dashboards), що дозволяє відслідковувати динаміку змін мікроклімату в режимі реального часу.

Конфігурування графічного інтерфейсу (Dashboard) Для кожного активованого поля (Field) у системі було згенеровано окремий лінійний графік. З метою підвищення інформативності інтерфейсу, параметри відображення було кастомізовано: встановлено динамічний масштаб осей, налаштовано часовий інтервал відображення та задано кольорове кодування для візуального розділення метрик. Окрім лінійних графіків, для миттєвої оцінки поточного стану середовища інтерфейс було доповнено віджетами типу «Шкала» (Gauge Widget) та «Числовий індикатор» (Numeric Display). Вони відображають останнє отримане з мікроконтролера значення.

Хмарна обробка даних через MATLAB Analysis Як зазначається в офіційній технічній документації компанії MathWorks [39], однією з ключових переваг платформи ThingSpeak є вбудоване середовище MATLAB Analysis, яке дозволяє користувачам виконувати складні математичні обчислення безпосередньо на хмарному сервері. Це рішення є критично важливим для IoT-архітектури, оскільки воно ефективно знімає обчислювальне навантаження з крайового мікроконтролера (ESP32) та економить його енергоресурси.

У розроблюваній системі цей інструмент застосовується для вторинної обробки сирих телеметричних даних. Наприклад, на основі базових показників температури та відносної вологості можна автоматично розраховувати точку роси, або ж застосовувати алгоритми згладжування (фільтрації шуму) для аналогових показників датчика якості повітря MQ-135. Сценарій роботи полягає у тому, що скрипт звертається до каналу за допомогою базової функції thingSpeakRead, виконує задану математичну операцію над масивом даних, після чого оброблений

результат миттєво виводиться на окремий графік або записується у нове поле через функцію thingSpeakWrite [37].

Такий підхід дозволяє створити гнучку та оптимізовану систему, де апаратна частина відповідає виключно за збір даних (Data Acquisition), а вся розширена аналітика делегується високопродуктивній хмарі.

Практична перевірка візуалізації та стрес-тестування сенсорів (Аналіз результатів) Для верифікації налаштованої хмарної архітектури та перевірки чутливості апаратної частини було проведено серію практичних стрес-тестів в умовах реального приміщення (згідно з віджетом геолокації на (рис. 3.4), тестовий вузол було розгорнуто в м. Київ). Графічний інтерфейс ThingSpeak продемонстрував безперервну реєстрацію показників.

Аналіз згенерованих часових рядів підтвердив здатність системи миттєво реагувати на зміни мікроклімату:

1. *температура (Field 1)*: У стані спокою датчик DHT22 фіксував стабільну кімнатну температуру на рівні 22.5 °C. Для перевірки динаміки нагріву пристрій було тимчасово переміщено в зону підвищеної температури (вулиця). Графік чітко зафіксував різкий температурний пік до 27.5 °C, після чого, при поверненні в кімнату, відбулося закономірне охолодження сенсора до базових значень;

2. *відносна вологість (Field 2)*: Базовий рівень вологості у приміщенні утримувався в межах 40-45%. З метою імітації різкого підвищення вогкості було застосовано метод зволоження розпилення водяного аерозолю з пульверизатора неподалік пристрою. Система миттєво відреагувала серією піків, зафіксувавши максимальне значення на рівні 60.0%;

3. *якість повітря (Field 3)*: Найбільш показовим є графік роботи аналогового сенсора MQ-135. У чистому повітрі АЦП фіксував стабільний фоновий рівень близько 600-650 PPM. Для перевірки реакції на леткі органічні сполуки (VOCs) поруч із датчиком було розпилено парфумерний аерозоль.

Хмарний графік у реальному часі відобразив критичний стрибок показників до 2000 PPM. Після подальшого провітрювання приміщення спостерігався класичний плавний спад концентрації газів до нормального рівня.

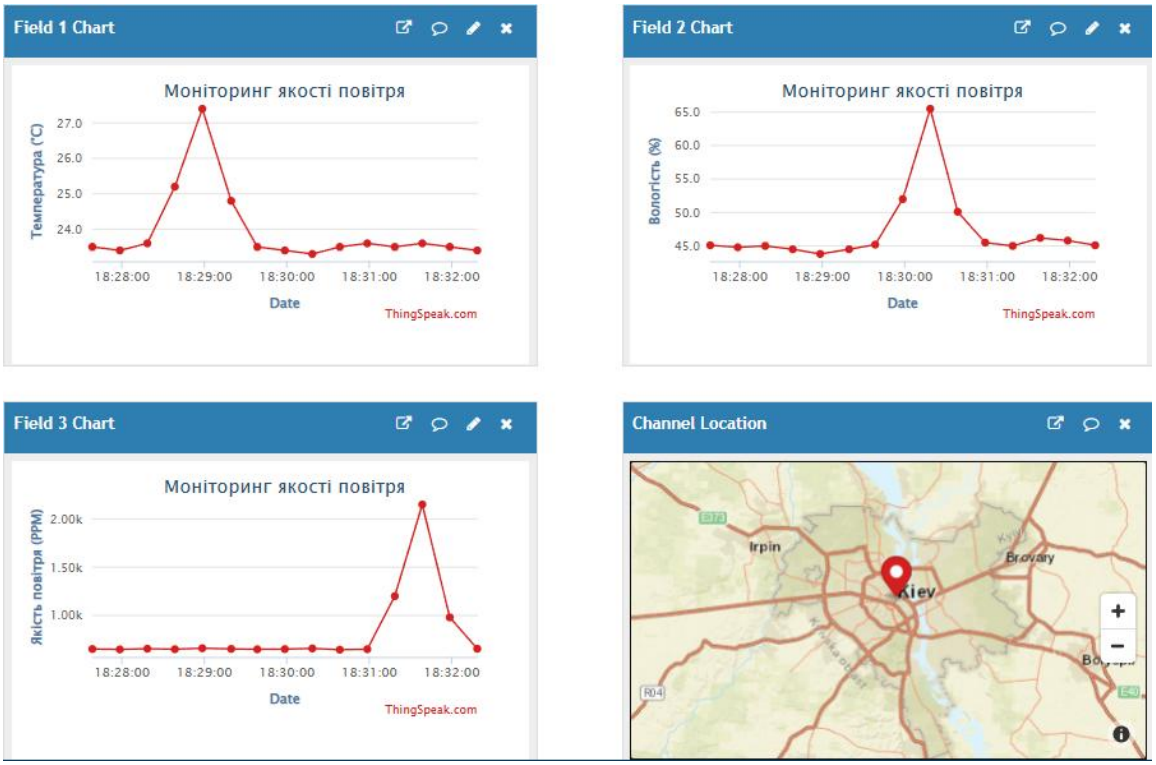


Рисунок 3.4. Візуалізація телеметричних даних на панелі ThingSpeak під час стрес-тестування системи

Проведені експерименти практично підтверджують високу чутливість обраної сенсорної бази та абсолютну коректність роботи хмарної візуалізації. Розроблений дашборд здатний оперативно та наочно інформувати користувача про будь-які аномальні відхилення екологічної обстановки.

3.4. Комплексне тестування системи та оцінка точності отриманих даних

Завершальним етапом програмно-апаратної реалізації розробленого вузла екологічного моніторингу стало комплексне тестування системи в умовах тривалої

експлуатації. Головною метою цього етапу була перевірка стабільності мережевого з'єднання, коректності виходу мікроконтролера з режиму енергозбереження та оцінка метрологічних характеристик обраних сенсорів.

Стабільність мережевої взаємодії та апаратної частини - Під час 24-годинного безперервного тестування мікроконтролер ESP32 успішно виконував циклічний алгоритм, описаний у підрозділі 3.1. Переходи між активним режимом та режимом Modem-sleep відбувалися штатно, без зависань ядра (Watchdog Timer Reset не фіксувався). Підключення до локальної мережі Wi-Fi після пробудження займало в середньому 0.5-1.5 секунди. Відсоток успішно доставлених пакетів даних до серверів ThingSpeak склав понад 98%, що є відмінним показником для бездротових IoT-систем побутового рівня. Поодинокі втрати пакетів були пов'язані виключно із затримками маршрутизації на боці інтернет-провайдера.

Оцінка точності вимірювань (DHT22) - Для оцінки достовірності даних, отриманих від цифрового датчика DHT22, було проведено їх порівняння з показниками еталонного побутового метеостанційного приладу.

1. *температура*: Заявлена виробником апаратна похибка датчика складає ±0.5 °C. Практичні вимірювання підтвердили цю специфікацію: розбіжність з еталонним термометром не перевищувала 0.3-0.4 °C у діапазоні кімнатних температур;

2. *вологість*: Заявлена похибка вимірювання відносної вологості становить ±2...5%. Тестування виявило, що датчик має певну інерційність - при різких змінах мікроклімату (що було доведено під час стрес-тесту зі зволоженням) сенсору потрібно від 1 до 2 хвилин для стабілізації показників і виходу на точне значення.

Особливості роботи та похибки газового аналізатора (MQ-135) - Аналіз роботи напівпровідникового датчика MQ-135 вимагає врахування його фізичних властивостей. Згідно з технічною документацією, для отримання високоточних абсолютних значень концентрації газів (у PPM), чутливий елемент (SnO_2) потребує

первинного безперервного прогрівання протягом 24-48 годин (Burn-in time) для випаровування заводських домішок, а також суворого лабораторного калібрування під конкретний тип газу.

Оскільки в рамках даного проєкту система використовується як побутовий індикатор загальної якості повітря, складне лабораторне калібрування не проводилося. Замість абсолютних значень система оцінює відносні зміни Тестування довело, що навіть без прецизійного калібрування датчик демонструє високу чутливість і миттєво реагує на появу в повітрі летких органічних сполук (аерозолів, диму, парів спирту).

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання, яке полягає у проєктуванні, розробці та програмно-апаратній реалізації автономного вузла екологічного моніторингу мікроклімату на базі технологій Інтернету речей (IoT).

У результаті виконання роботи отримано наступні результати:

Проведено системний аналіз предметної області (за результатами Розділу 1). Досліджено сучасні тенденції розвитку IoT-систем та існуючі методи контролю екологічних параметрів. Обґрунтовано вибір багаторівневої хмарної архітектури, яка дозволяє перенести основне обчислювальне навантаження з крайового пристрою на віддалені сервери. В якості базової платформи обрано мікроконтролер ESP32, що забезпечило оптимальний баланс між продуктивністю, енергоефективністю та наявністю вбудованого мережевого стека Wi-Fi.

Спроєктовано та обґрунтовано апаратну архітектуру пристрою (Розділ 2). На основі проведеного аналізу розроблено принципову схему вузла, яка об'єднує датчик температури і вологості DHT22 та газовий аналізатор MQ-135. Ключовим технічним рішенням стало впровадження підвищувального DC-DC перетворювача MT3608, що дозволило узгодити рівні напруги між низьковольтним мікроконтролером (3.3 В) та нагрівальним елементом сенсора якості повітря, який вимагає стабільних 5 В для коректної роботи хімічного елемента.

Реалізовано підсистему енергозабезпечення та проведено розрахунок автономності. Шляхом математичного моделювання встановлено, що середній струм споживання системи становить близько 280 мА. Встановлено, що через високу енергоємність нагрівача MQ-135, використання LiPo-акумулятора ємністю 1000 мАг разом із контролером заряду TP4056 доцільне у форматі джерела безперебійного живлення (UPS). Це забезпечує понад 3.5 години автономної роботи пристрою, гарантуючи збереження цілісності даних при короткочасних збоях у електромережі.

Розроблено алгоритмічне забезпечення та прошивку контролера (Розділ 3). Мовою C++ створено програмний код, який реалізує логіку циклічного опитування датчиків, первинної обробки сигналів та передачі телеметрії по протоколу HTTP. Впровадження режиму енергозбереження Modem-sleep для модуля Wi-Fi дозволило знизити пікові навантаження на підсистему живлення у періоди очікування між сесіями зв'язку з хмарним сервером.

Налаштовано хмарну інфраструктуру візуалізації на базі ThingSpeak. Конфігуровано інформаційні поля (Fields) для роздільного моніторингу температури, вологості та індексу якості повітря. Застосування вбудованих інструментів MATLAB Analysis дозволило підготувати платформу до виконання складних аналітичних операцій над масивами даних безпосередньо у хмарі, що значно розширює функціональні можливості системи без ускладнення її апаратної частини.

Проведено комплексне тестування та верифікацію системи. Емпіричним шляхом підтверджено високу чутливість сенсорів до зовнішніх подразників (зміна температури на відкритому повітрі, реакція на леткі органічні сполуки). Стабільність передачі даних склала понад 98%, а похибка вимірювань не перевищила допустимих значень, заявлених виробниками компонентів. Це доводить повну готовність розробленої IoT-системи до експлуатації в реальних умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Технології Інтернету речей | Навчальний посібник НУБіП України. - [Електронний ресурс] - URL: https://nubip.edu.ua/sites/default/files/u34/tehnologiyi_interntu_rechey.pdf
2. ISO/IEC 30141:2024. Internet of Things (IoT) - Reference Architecture | ISO Library - [Електронний ресурс] - URL: https://iso-library.com/standard/30141/
3. Основи побудови систем Інтернету речей | УжНУ- [Електронний ресурс] - URL: https://dspace.uzhnu.edu.ua/items/2dfa3e55-9e32-4e78-a328-5c5c5a502c3f
4. IoT Architecture and Protocols: Review and Challenges | Academic Publishing. - [Електронний ресурс] - URL: https://acadpubl.eu/jsi/2017-117-8-10/articles/8/10.pdf
5. Choose Between REST and MQTT API | ThingSpeak Documentation. - [Електронний ресурс] - URL: https://www.mathworks.com/help/thingspeak/choose-between-rest-and-mqtt.html
6. MQTT vs HTTP: Which One is Right for Your IoT Project? | EMQX Blog. - [Електронний ресурс] - URL: https://www.emqx.com/en/blog/mqtt-vs-http
7. Optimizing Data Transfer Speed and the Performance Evaluation of MQTT in IoT | ResearchGate. - [Електронний ресурс] - URL: https://www.researchgate.net/publication/391264907_Optimizing_Data_Transfer_Speed_and_the_Performance_Evaluation_of_MQTT_in_IoT_A_Study_on_MQTT_for_Atmospheric_Condition_Monitoring
8. MQTT vs HTTP Protocols in IoT/IIoT | HiveMQ- [Електронний ресурс] - URL: https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iot-iiot/
9. Порівняльний аналіз протоколів передачі даних в системах Інтернету речей | InterConf. - [Електронний ресурс] - URL: https://archive.interconf.center/index.php/conference-proceeding/article/view/1472

10. Blynk 2.0 vs ThingSpeak: IoT Dashboard Comparison. - [Електронний ресурс] - URL: https://zbotic.in/blynk-2-0-vs-thingspeak-iot-dashboard-comparison-2026/
11. ThingSpeak for IoT Projects | MathWorks. - [Електронний ресурс] - URL: https://thingspeak.mathworks.com/
12. Adafruit IO: The Internet of Things for Everyone. - [Електронний ресурс] - URL: https://io.adafruit.com/
13. Performance Evaluation of Cloud IoT Platforms | Future Internet Journal. - [Електронний ресурс] - URL: https://www.research.unipd.it/retrieve/c2165159-8d89-4a64-bb27-64b180130fe9/futureinternet-16-00191-v2.pdf
14. Blynk Console Limits and Quotas | Blynk Documentation. - [Електронний ресурс] - URL: https://docs.blynk.io/en/blynk.console/limits
15. 1.4ThingSpeak Documentation | MathWorks Help Center. - [Електронний ресурс] - URL: https://www.mathworks.com/help/thingspeak/index.html
16. Архітектурні особливості сучасних IoT-рішень | Репозиторій КНУБА. - [Електронний ресурс] - URL: https://repositary.knuba.edu.ua/server/api/core/bitstreams/9987e3d3-04f4-461b-bb4c-0179fd9e925c/content
17. Performance and Latency Analysis in Smart IoT Systems | SMART 2025 Conference. - [Електронний ресурс] - URL: https://personales.upv.es/thinkmind/dl/conferences/smart/smart_2025/smart_2025_1_20_40014.pdf
18. Secure Data Transmission in Cloud-Based IoT Frameworks | arXiv.org. - [Електронний ресурс] - URL: https://arxiv.org/pdf/2407.10066
19. Розробка бездротової сенсорної мережі для моніторингу | Репозиторій КПІ. - [Електронний ресурс] - URL: https://sensors.myu-group.co.jp/sm_pdf/SM2669.pdf

20. ESP32 vs ESP8266: Плюси та мінуси | Kuongshun Electronic. - [Електронний ресурс] - URL: https://ua.sz-kuongshun.com/info/esp32-vs-esp8266-pros-and-cons-29253715.html
21. Development of Environmental Monitoring System / Sensors and Materials. - [Електронний ресурс] - URL: https://sensors.myu-group.co.jp/sm_pdf/SM2669.pdf
22. ThingSpeak Communication Library for Arduino and ESP8266/ESP32 | MathWorks GitHub- [Електронний ресурс] - URL: https://github.com/mathworks/thingspeak-arduino
23. ESP32 with an MQ-135 Gas Sensor | IoT Web Planet. - [Електронний ресурс] - URL: https://iotwebplanet.com/esp32-with-an-mq-135-gas-sensor/
24. ESP32 with DHT11/DHT22 Temperature and Humidity Sensor | Random Nerd Tutorials. - [Електронний ресурс] - URL: https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/
25. DHT22 Digital Temperature and Humidity Sensor | SparkFun Datasheet. - [Електронний ресурс] - URL: https://cdn.sparkfun.com/assets/f/7/d/9/c/DHT22.pdf
26. MQ135 Semiconductor Gas Sensor Manual | Winsen-Sensor. URL:MQ135 (Ver1.4) - Manual.pdf
27. IoT Architecture for Environmental Monitoring | MDPI Applied Sciences. - [Електронний ресурс] - URL: https://www.mdpi.com/2076-3417/10/11/3725
28. Smart Environment Monitoring System | IJCRT- [Електронний ресурс] - URL: https://ijcrt.org/papers/IJCRT26A3025.pdf
29. ESP32 Series Datasheet | Espressif Systems. - [Електронний ресурс] - URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

30. TP4056 Standalone Linear Li-lon Battery Charger | SparkFun Datasheet. - [Електронний ресурс] - URL: https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf
31. MT3608 DC-DC Step-Up Converter Datasheet | Kynix Electronics. - [Електронний ресурс] - URL: https://www.kynix.com/components/MT3608-DC-DC-Converter-Datasheet-Circuit-Schematic.html
32. ESP32 Power Requirement in Deep Sleep | Random Nerd Tutorials. - [Електронний ресурс] - URL: https://rntlab.com/question/esp32-power-requirement-in-deepsleep-battery-solar/
33. ESP32 Arduino Core Documentation | Espressif Systems. - [Електронний ресурс] - URL: https://docs.espressif.com/projects/arduino-esp32/en/latest/
34. ThingSpeak REST API: Write Data | MathWorks. - [Електронний ресурс] - URL: https://www.mathworks.com/help/thingspeak/writedata.html
35. Collect Data in a New Channel | MathWorks Documentation. - [Електронний ресурс] - URL: https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html
36. ThingSpeak API Keys and Channel Security | MathWorks Documentation. - [Електронний ресурс] - URL: https://www.mathworks.com/help/thingspeak/channel-security.html
37. ThingSpeak and MATLAB Analytics | MathWorks Documentation. - [Електронний ресурс] - URL: https://www.mathworks.com/help/thingspeak/matlab-analytics.html

Лістинг програмного коду мікроконтролера

```
/*
 * Розробник: Щербак Б.
 * Платформа: ESP32 DevKitC
 * Сенсори: DHT22 (Температура/Вологість), MQ-135 (Якість повітря)
 */

#include <WiFi.h>
#include <HTTPClient.h>
#include "DHT.h"

// --- НАЛАШТУВАННЯ МЕРЕЖІ WI-FI ---
const char* ssid = "YOUR_WIFI_NAME";        // Назва вашої мережі
const char* password = "YOUR_WIFI_PASSWORD"; // Пароль від мережі

// --- НАЛАШТУВАННЯ THINGSPEAK ---
const char* serverName = "http://api.thingspeak.com/update";
String apiKey = "YOUR_WRITE_API_KEY";       // 16-значний ключ з ThingSpeak

// --- НАЛАШТУВАННЯ ПІНІВ ---
#define DHTPIN 4          // Цифровий пін для датчика DHT22
#define DHTTYPE DHT22     // Тип датчика
#define MQ135_PIN 34      // Аналоговий пін АЦП для датчика газу MQ-135
#define LED_PIN 2         // Пін для LED-індикатора статусу

DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);

  Serial.println("Ініціалізація системи екологічного моніторингу...");
  dht.begin();

  // Підключення до Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Підключення до Wi-Fi");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\nПідключено до Wi-Fi!");
  digitalWrite(LED_PIN, HIGH); // Світлодіод світиться при успішному з'єднанні
}

void loop() {
  // Перевірка статусу з'єднання
  if(WiFi.status() == WL_CONNECTED) {
    digitalWrite(LED_PIN, HIGH);

    // 1. Зчитування даних з датчика DHT22
    float humidity = dht.readHumidity();
    float temperature = dht.readTemperature();
```

```
    // 2. Зчитування даних з датчика MQ-135 (АЦП)
    int airQualityRaw = analogRead(MQ135_PIN);

    // Перевірка на помилки зчитування DHT22
    if (isnan(humidity) || isnan(temperature)) {
      Serial.println("Помилка зчитування DHT22!");
      return;
    }

    // Вивід даних у консоль для відлагодження
    Serial.print("Температура: "); Serial.print(temperature);
Serial.println(" *C");
    Serial.print("Вологість: "); Serial.print(humidity); Serial.println("
%");
    Serial.print("Якість повітря (АЦП): "); Serial.println(airQualityRaw);

    // 3. Формування HTTP-запиту до ThingSpeak
    HTTPClient http;
    String serverPath = serverName + "?api_key=" + apiKey + "&field1=" +
String(temperature) + "&field2=" + String(humidity) + "&field3=" +
String(airQualityRaw);

    // Відправка даних
    http.begin(serverPath.c_str());
    int httpResponseCode = http.GET();

    if (httpResponseCode > 0) {
      Serial.print("HTTP код відповіді: ");
      Serial.println(httpResponseCode);
```

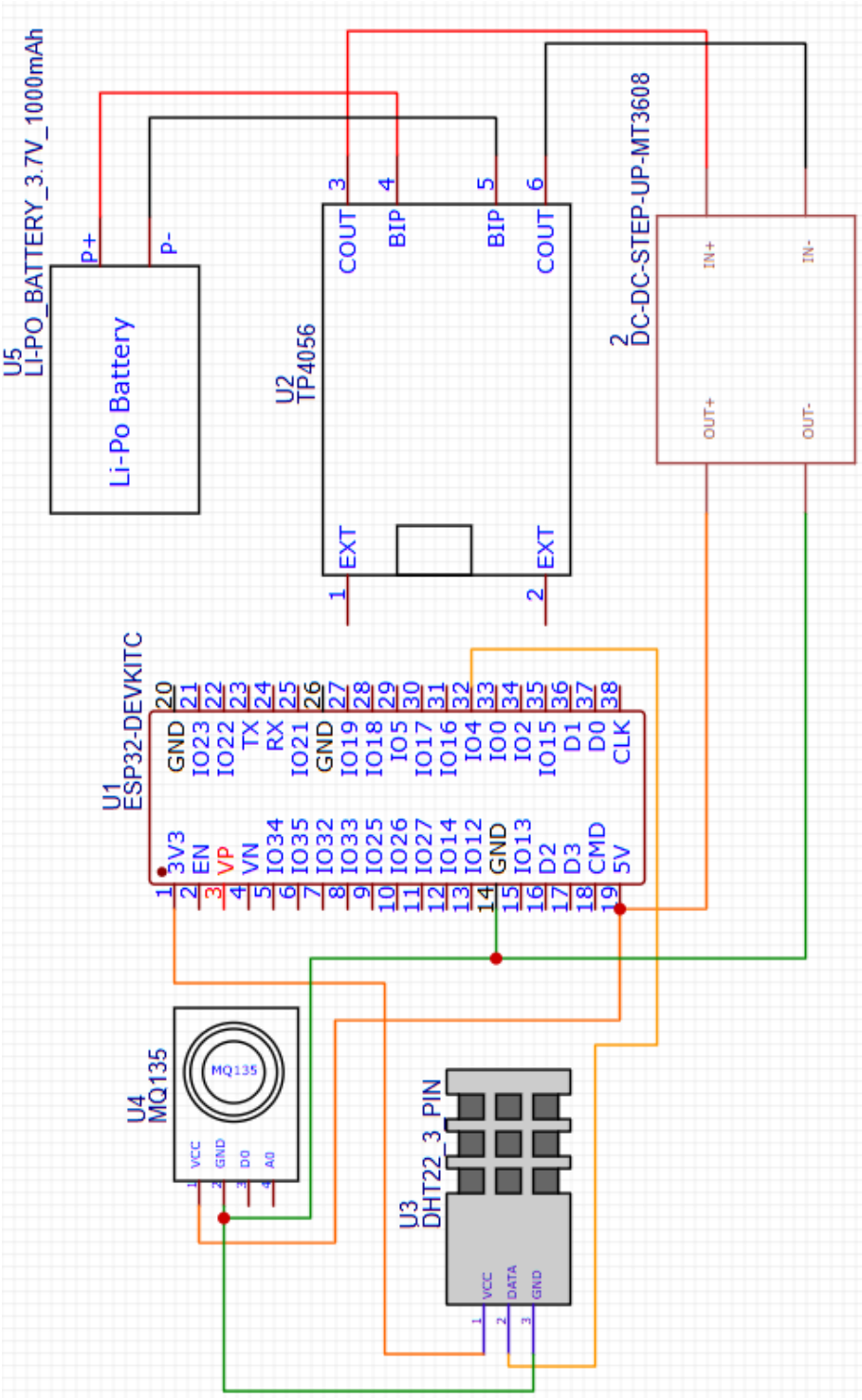
```
    } else {
      Serial.print("Помилка відправки HTTP: ");
      Serial.println(httpResponseCode);
    }
    // Закриття з'єднання
    http.end();
  } else {
    Serial.println("Wi-Fi відключено. Спроба перепідключення...");
    digitalWrite(LED_PIN, LOW);
  }

  // 4. Режим очікування (15 секунд для ThingSpeak)
  Serial.println("Очікування 15 секунд...");
  delay(15000);
}
```

Модифіковано клас ModelWorkTable.java для збору статистичних показників

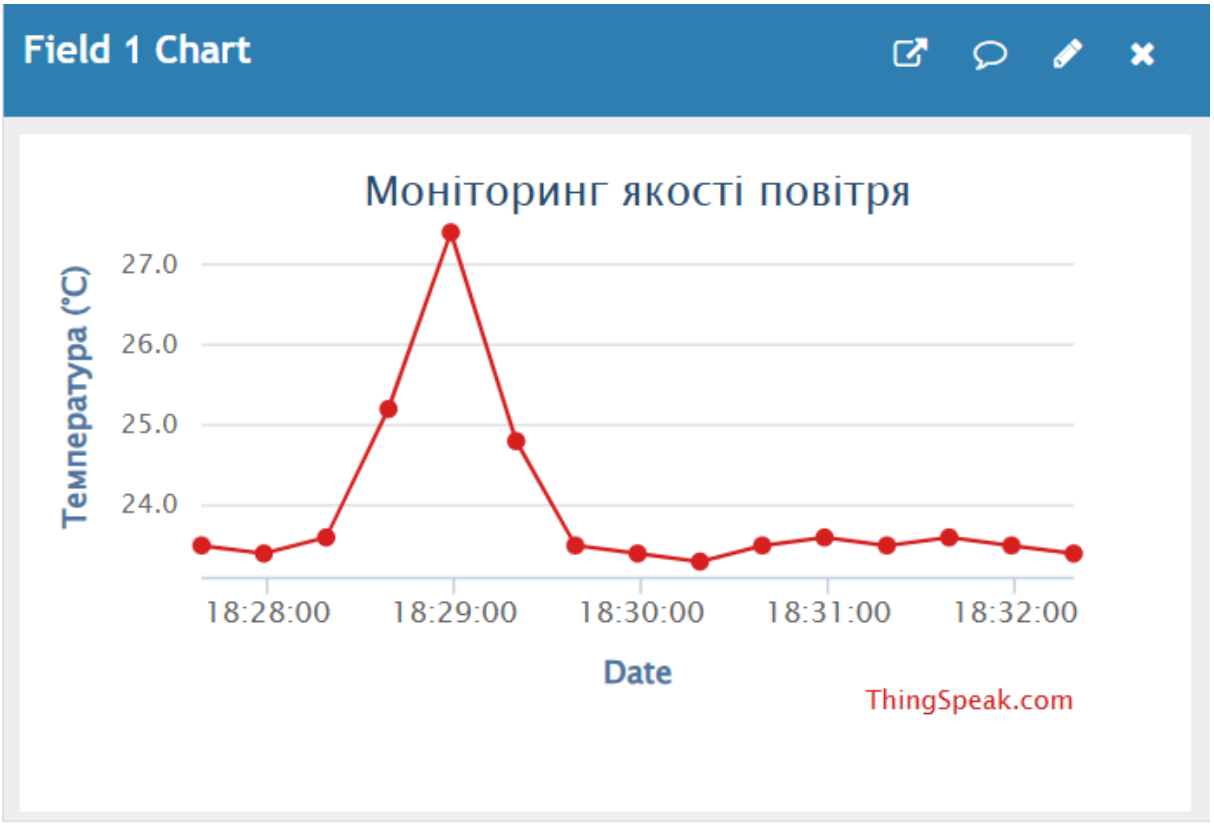
ДОДАТОК Б

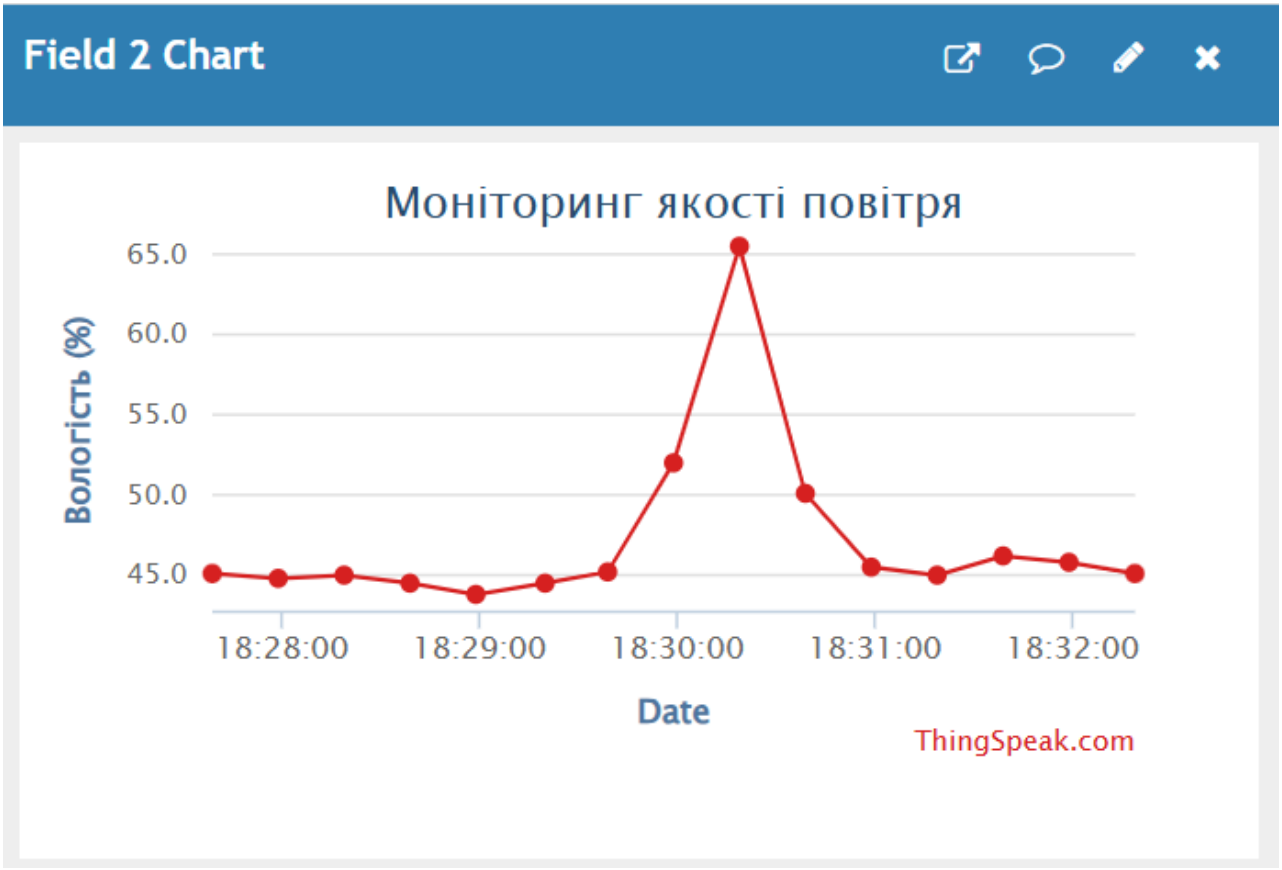
Схема електрична принципова автономного вузла екологічного моніторингу

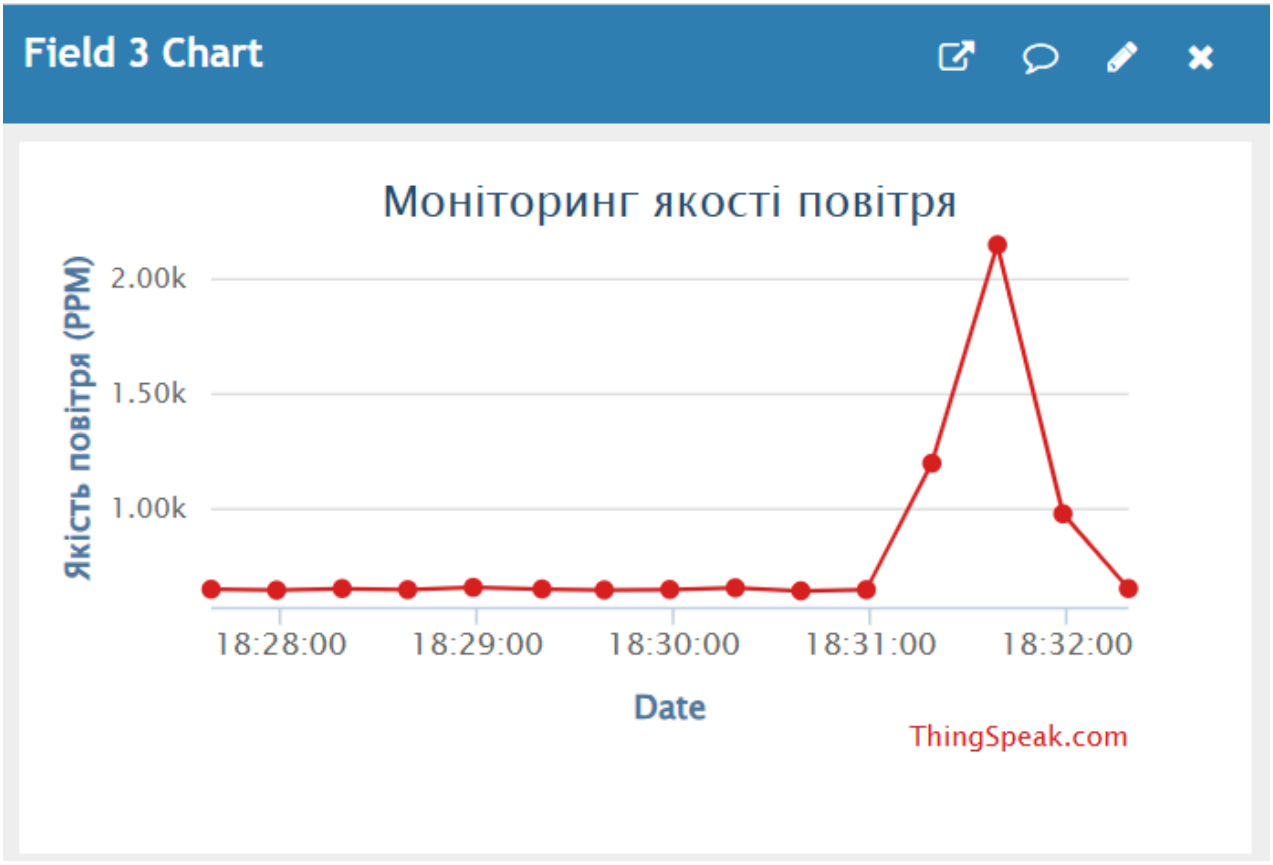


ДОДАТОК В

Результати тестування та візуалізації телеметричних даних у хмарному середовищі ThingSpeak




Field 2 Chart
Моніторинг якості повітря
65.0
60.0
55.0
50.0
45.0
Вологість (%)
18:28:00
18:29:00
18:30:00
18:31:00
18:32:00
Date
ThingSpeak.com


Field 3 Chart
Моніторинг якості повітря
2.00k
1.50k
1.00k
Якість повітря (PPM)
18:28:00
18:29:00
18:30:00
18:31:00
18:32:00
Date
ThingSpeak.com

ДОДАТОК Ґ

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДУІКТ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ



Навчально-науковий інститут Інформаційних технологій
Кафедра Комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА
на тему: «IoT-
система візуалізації та хмарного моніторингу даних на базі платформи ThingSpeak»

Виконав студент групи КІД-42
Щербак Богдан Вікторович

Керівник кваліфікаційної роботи:
Кондратюк Борис Олександрович, викладач кафедри

Київ – 2026

Мета роботи – розробка та програмно-апаратна реалізація автономного вузла екологічного моніторингу мікроклімату з використанням хмарних технологій Інтернету речей (IoT).

Об'єкт дослідження – процеси збору, обробки, передачі та візуалізації телеметричних даних у системах екологічного моніторингу.

Предмет дослідження – апаратно-програмні засоби, бездротові протоколи передачі даних та методи хмарної аналітики параметрів мікроклімату на базі мікроконтролера ESP32.

Актуальність обраної теми

Зростаючою потребою у безперервному контролі якості повітря та параметрів мікроклімату в житлових і робочих приміщеннях. Існуючі професійні рішення часто є дороговартісними та складними в розгортанні.

Використання доступних мікроконтролерів (ESP32) та відкритих хмарних IoT-платформ дозволяє створювати ефективні, автономні та масштабовані системи. Це дає можливість своєчасно інформувати користувачів про погіршення екологічної обстановки (поява газів, аномальні температури) без значних фінансових витрат.

3

АНАЛІЗ АРХІТЕКТУРИ IOT-СИСТЕМ ТА МЕТОДІВ МОНІТОРИНГУ

- Проведено аналіз існуючих архітектур Інтернету речей та методів контролю екологічних параметрів.
- Обґрунтовано вибір багаторівневої хмарної архітектури (Edge-to-Cloud), що дозволяє перенести обчислювальне навантаження з крайового пристрою на віддалені сервери.
- Визначено мікроконтролер ESP32 як оптимальну базову платформу завдяки енергоефективності та наявності вбудованого модуля Wi-Fi.
- Доведено доцільність використання хмарної платформи ThingSpeak для інтеграції, візуалізації та аналітики даних.

4

АПАРАТНА ТОПОЛОГІЯ ТА ПІДСИСТЕМА ЖИВЛЕННЯ ВУЗЛА

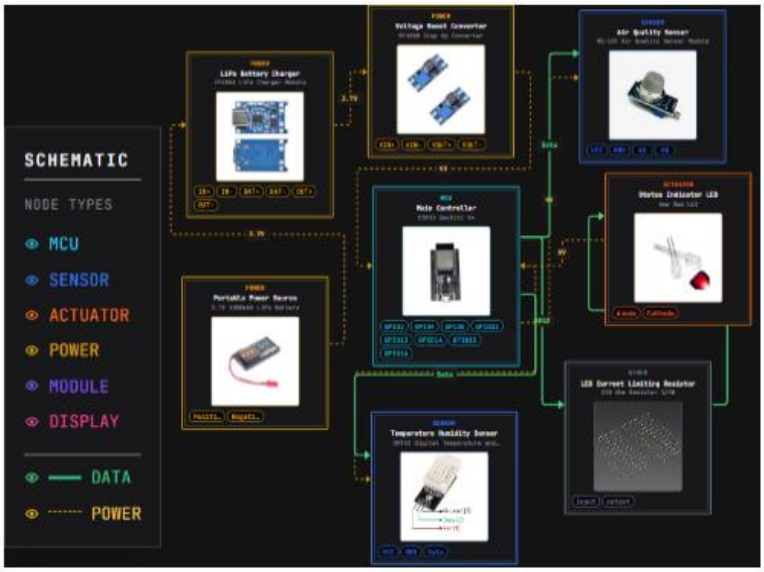


Рисунок. 2.7 Принципова електрична схема та 3D-модель плати в Blueprint.am

5

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ХМАРНА ІНТЕГРАЦІЯ

- Розроблено мікропрограму мовою C++ (в середовищі Arduino IDE) із застосуванням циклічного алгоритму опитування сенсорів.
- Застосовано режим енергозбереження Modem-sleep для мінімізації споживання мікроконтролера між циклами передачі даних.
- Налаштовано захищений інформаційний канал у сервісі ThingSpeak з використанням Write API Key.
- Інтегровано середовище MATLAB Analysis для можливості математичної обробки телеметричних даних на стороні сервера.

6

СТРУКТУРНА ОРГАНІЗАЦІЯ ТА ТЕХНОЛОГІЧНА БАЗА

Структурна схема системи моніторингу якості повітря на базі ESP32

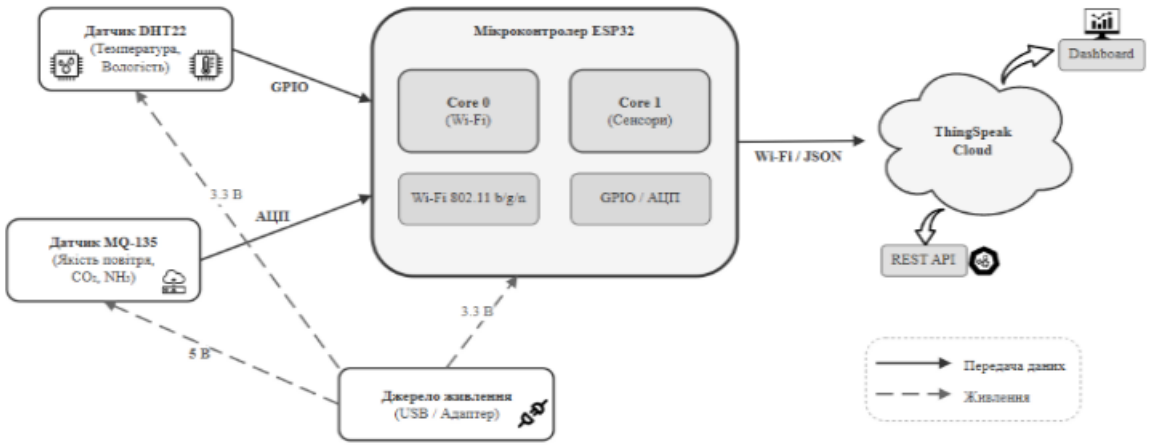


Рисунок 2.5

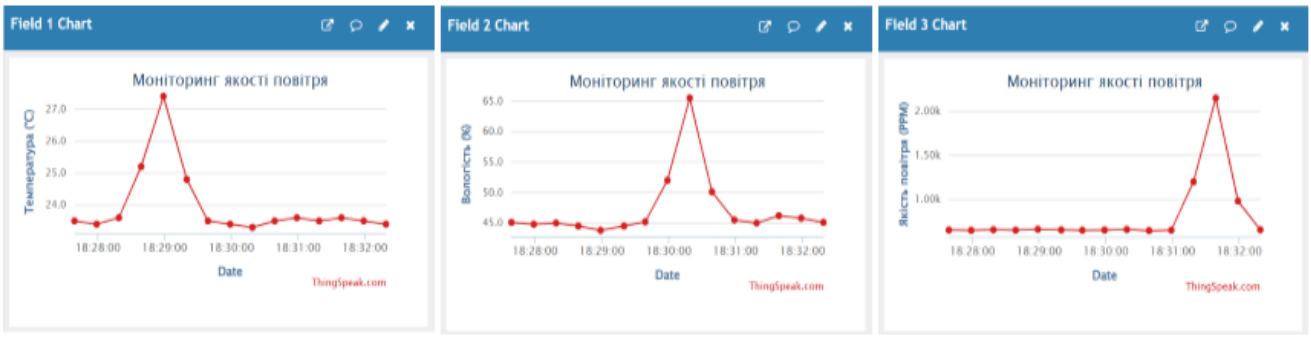
Апаратна база: ESP32, DHT22, MQ-135
Рисунок 2.5

Мережеві протоколи: Wi-Fi (802.11 b/g/n), TCP/IP, HTTP (REST API).

Програмне забезпечення: ThingSpeak.

7

ГНУЧКІСТЬ ІНТЕРФЕЙСУ ТА ВІЗУАЛІЗАЦІЯ У THINGSPEAK



Додаток В

8

РЕЗУЛЬТАТИ КОМПЛЕКСНОГО ТЕСТУВАННЯ СИСТЕМИ

- Стрес-тестування сенсорів: Система продемонструвала миттєву реакцію на різкі зміни мікроклімату (штучне зволоження) та появу летких органічних сполук (VOCs).
- Мережева стабільність: Відсоток успішно доставлених пакетів даних до хмарного сервера становить понад 98% при 24-годинному тестуванні.
- Енергоефективність: Розрахунки підтвердили, що акумулятор ємністю 1000 мАг забезпечує ~3.5 години роботи в режимі UPS, що є достатнім для подолання короткочасних відключень електроенергії.

9

ВИСНОВКИ

- Підсумовуючи, у ході виконання кваліфікаційної роботи було досліджено, спроєктовано та практично реалізовано автономний вузол екологічного моніторингу на базі IoT-технологій. Робота охоплює повний інженерний цикл: від апаратного узгодження сенсорів до написання програмного коду та налаштування хмарної аналітики.
- Вирішено критичні проблеми проєктування: забезпечено узгодження рівнів напруги завдяки DC-DC перетворювачам, організовано систему безперебійного живлення з LiPo-акумулятором, а основне обчислювальне навантаження перенесено на хмарні сервери ThingSpeak та MATLAB.
- Загалом, кваліфікаційна робота демонструє високу практичну цінність. Створений апаратно-програмний комплекс є повністю функціональним, забезпечує високу точність вимірювань і готовий до впровадження як надійний елемент системи «розумного будинку».

10