Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
Програмне забезпечення для моніторингу виробничих параметрів на основі системи датчиків
(тема)

Виконав:
здобувач 2 року навчання,
групи КТРСм-24-2
Максим МОРОЗ
(власне ім'я, прізвище)

Спеціальність 174 Автоматизаці, комп'ютерно-інтегровані технології та робототехніка
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Комп'теризовані та робототехнічні системи
(повна назва освітньої програми)

Керівник проф. Олександр ЦИМБАЛ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис) (власне ім'я, прізвище)

2025 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики і комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти Другий (магістерський)
Спеціальність 174 Автоматизація та комп'ютерно інтегровані технології, робототехніка
Тип програми Освітньо-професійна
Освітня програма Комп'ютеризовані та робототехнічні системи

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувачеві Морозу Максиму Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмне забезпечення для моніторингу виробничих параметрів на основі системи датчиків
Затверджена наказом по університету від 10.11.2025 № 1018 Ст.
2. Термін подання здобувачем роботи до екзаменаційної комісії 23.12.2025
3. Вхідні дані до роботи

4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.3 Вибір та обґрунтування компонентів системи
4.4 Розробка програмної частини
4.5 Тестування та налагодження системи
4.6 Висновки та перелік джерел посилань

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій
Демонстраційний матеріал, представлений у форматі презентації PowerPoint (*.ppt) – 12 с. формату А4.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз систем моніторингу виробничих параметрів	22.09-5.10.2025	Виконано
2	Проектування системи моніторингу виробництва	6.10-15.10.2025	Виконано
3	Обґрунтування вибору апаратної платформи та датчиків	15.10-22.10.2025	Виконано
4	Обґрунтування вибору програмних технологій	23.10-29.10.2025	Виконано
5	Схема підключення датчиків до контролера	30.10-05.11.2025	Виконано
6	Проєктування бази даних	06.11-12.11.2025	Виконано
7	Розробка архітектури системи моніторингу	13.11-18.11.2025	Виконано
8	Розробка прошивки контролера	19.11-24.11.2025	Виконано
9	Реалізація MQTT-комунікації з брокером	25.11-27.11.2025	Виконано
10	Тестування системи моніторингу	28.11-29.11.2025	Виконано
11	Оформлення пояснювальної записки	30.11-6.12.2025	Виконано

Дата видачі завдання 01.09.2025

Здобувач _____ Максим МОРОЗ
(підпис) (прізвище, ініціали)

Керівник роботи _____ проф. Олександр ЦИМБАЛ
(підпис) (прізвище, ініціали)

Я, Мороз Максим Васильович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу з академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував(ла) штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«22» грудня 2025р.  Мороз М. В.

РЕФЕРАТ

Пояснювальна записка: 95 с., 3 табл., 24 рис., 12 лист., 38 джерел.

СИСТЕМА МОНІТОРИНГУ, MQTT, БАЗА ДАНИХ, WPF, ESP32

Об’єкт дослідження – процес моніторингу виробничих параметрів з використанням мікроконтролерних пристроїв та програмних засобів опрацювання даних.

Предмет дослідження – програмно-апаратна система збору, передавання, збереження та відображення показників датчиків середовища.

Методи дослідження – аналіз технічної літератури, дослідження характеристик сенсорів, проєктування архітектури, розроблення ПЗ на основі мови C++, MQTT-протоколу, середовища C# WPF та СУБД Microsoft SQL Server.

Метою роботи є обґрунтування та реалізація підходу до організації моніторингу виробничих параметрів на об’єктах малого та середнього бізнесу з використанням доступних мікроконтролерних і програмних засобів.

У ході роботи виконано аналіз технологій моніторингу середовища, досліджено апаратні компоненти, розглянуто мережеві протоколи, обґрунтовано вибір ESP32, DHT22 і MQ-2. Спроєктовано структуру бази даних, розроблено архітектуру системи та реалізовано прошивку контролера для передавання даних. Створено десктопний застосунок для приймання телеметрії, її збереження та візуалізації. Проведено тестування у симуляторі Wokwi та перевірено роботу MQTT-брокера і клієнтської частини.

Розроблена система забезпечує отримання та обробку телеметрії у режимі реального часу та може бути адаптована для використання у виробничих процесах різного призначення.

ABSTRACT

Explanatory note: 95 pages, 3 tables, 24 figures, 12 listings, 38 sources.

MONITORING SYSTEM, MQTT, DATABASE, WPF, ESP32

The object of research is the process of monitoring production parameters using microcontroller devices and data processing software.

The subject of research is a software and hardware system for collecting, transmitting, storing, and displaying environmental sensor readings.

Research methods – analysis of technical literature, study of sensor characteristics, architecture design, software development based on the C++ language, MQTT protocol, C# WPF environment, and Microsoft SQL Server DBMS.

The aim of the work is to justify and implement an approach to organizing the monitoring of production parameters at small and medium-sized businesses using available microcontrollers and software tools.

In the course of the work, technologies for environmental monitoring were analyzed, hardware components were examined, and network communication protocols were reviewed. The selection of ESP32, DHT22, and MQ-2 was justified. The database structure was designed, the system architecture was developed, and the controller firmware for data transmission was implemented. A desktop application was created to receive telemetry, store it, and visualize it. Testing was conducted in the Wokwi simulator, and the operation of the MQTT broker and client application was verified.

The developed system provides real-time telemetry acquisition and processing and can be adapted for use in various industrial processes.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

HART – Highway Addressable Remote Transducer

IIC – Inter-Integrated Circuit

IoT – Internet of Things

MQTT – Message Queuing Telemetry Transport

SCADA – Supervisory Control and Data Acquisition

SQL – Structured Query Language

UART – Universal Asynchronous Receiver–Transmitter

USB – Universal Serial Bus

WPF – Windows Presentation Foundation

ВСТУП

Виробничі процеси характеризуються змінами параметрів середовища, що впливають на якість продукції та безпеку технологічних операцій. Відстеження температури, вологості, тиску та інших фізичних величин у режимі реального часу дозволяє виявляти відхилення від норми та запобігати аваріям чи браку. Розробка програмного забезпечення для автоматизованого моніторингу виробничих параметрів на основі системи датчиків забезпечує підприємства засобами контролю технологічних процесів.

Мета роботи є забезпечення можливості впровадження системи моніторингу виробничих параметрів на підприємствах малого та середнього бізнесу шляхом використання доступних апаратних компонентів і програмних засобів без залучення дорогих промислових комплексів.

Предмет дослідження – програмне забезпечення для моніторингу виробничих параметрів на основі системи датчиків, що включає прошивку контролера Arduino на мові C++, організацію передавання даних через MQTT-протокол та десктопний застосунок на платформі C# WPF з інтеграцією бази даних Microsoft SQL Server.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати предметну область систем моніторингу виробничих параметрів та існуючі рішення у цій сфері;

– обґрунтувати вибір апаратної платформи, датчиків та протоколу передавання даних;

– спроєктувати структуру бази даних для зберігання показників датчиків з урахуванням часових міток;

– розробити прошивку контролера для зчитування даних з датчиків та публікації їх до MQTT-брокера;

– реалізувати десктопний застосунок з графічним інтерфейсом для підписки на топіки MQTT, отримання даних, їх збереження у базі даних та візуалізації у вигляді графіків і таблиць;

– впровадити режим симуляції для демонстрації функціонування системи без фізичних датчиків;

– провести тестування розробленого програмного забезпечення з метою перевірки коректності передавання даних, точності збереження показників та стабільності роботи системи.

Кваліфікаційна робота виконана згідно ДСТУ 3008–15 [1], методичних вказівок [2].

1 АНАЛІЗ СИСТЕМ МОНИТОРИНГУ ВИРОБНИЧИХ ПАРАМЕТРІВ

1.1 Аналіз предметної області систем моніторингу виробничих параметрів

Моніторинг виробничих параметрів становить основу забезпечення якості технологічних процесів на промислових підприємствах. Системи моніторингу призначені для спостереження, збирання, обробки, передавання, збереження та аналізу інформації про стан контрольованих об'єктів з метою прогнозування змін та прийняття рішень щодо запобігання негативним відхиленням. Виробничі параметри охоплюють широкий спектр фізичних величин, що характеризують технологічний процес. До них належать температура, вологість, тиск, рівень рідини, концентрація газів, вібрація обладнання та інші показники, які впливають на якість продукції та безпеку виробництва [4].

Процес моніторингу реалізується через послідовність операцій, що включає виділення об'єкта спостереження, збір даних від датчиків, передавання інформації до центральної системи обробки, аналіз отриманих показників та формування керуючих впливів чи сповіщень. Датчики перетворюють фізичні параметри в електричні сигнали, які надалі оцифровуються та передаються засобами провідного або бездротового зв'язку до систем верхнього рівня. Автоматизація технологічних процесів передбачає застосування датчиків для контролю всіх параметрів, що впливають на виробництво, адже своєчасна реакція систем на зміни у контрольованому об'єкті запобігає аваріям та виходу з ладу обладнання [5].

Структура системи моніторингу на основі стану обладнання включає декілька функціональних блоків, що забезпечують збір та обробку інформації від контрольованого об'єкта (рисунок 1.1). Нижній рівень формують вимірювальні датчики, які безпосередньо взаємодіють з обладнанням під час тестування та фіксують його параметри. Контролери процесів здійснюють первинну обробку сигналів та керування виконавчими пристроями. Інтерфейси управління

забезпечують взаємодію оператора з системою [6]. Блок попередньої обробки сигналу виконує фільтрацію, нормалізацію та підготовку даних для передавання до верхнього рівня. На вершині структури розташовуються підсистеми збору даних, аналізу та управління, які формують загальне уявлення про стан обладнання, а також промислові мережі для інтеграції з корпоративними інформаційними системами. Такий підхід дозволяє розподілити функції між компонентами системи та забезпечити масштабованість рішення.

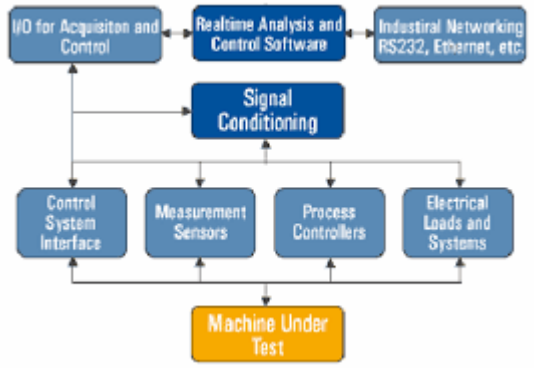

Рисунок 1.1 – Структура системи моніторингу на основі стану обладнання

Передавання даних від датчиків до систем обробки здійснюється через промислові мережі та протоколи, серед яких розповсюдження отримали Modbus, Profibus, HART, а також технології Інтернету речей. Промисловий Інтернет речей охоплює апаратні та програмні засоби моніторингу фізичних пристроїв, зосереджуючись на оцінці продуктивності, збиранні даних та реагуванні в режимі реального часу. Датчики у складі IoT-систем обладнані можливістю передавання інформації через бездротові протоколи, такі як LoRaWAN, Zigbee, NB-IoT, що забезпечує гнучкість розгортання мережі без прокладання кабельних з'єднань. Зібрані дані надходять до хмарних сховищ або локальних серверів, де програмне забезпечення виконує їх аналіз та виявлення аномалій [7].

Принцип роботи типової системи моніторингу базується на циклічному опитуванні датчиків контролером, передаванні зібраних даних до сервера обробки через комунікаційну мережу та відображенні поточного стану параметрів на робочій станції оператора. Частота опитування датчиків

визначається динамікою контрольованого процесу та може варіюватися від декількох разів на секунду для швидкоплинних процесів до одного разу на хвилину для повільно змінюваних величин. Отримані дані зберігаються у базі даних з прив'язкою до часових міток, що дозволяє відтворювати історію зміни параметрів та виконувати ретроспективний аналіз технологічних режимів.

Системи класу SCADA становлять окрему категорію рішень для моніторингу та управління виробничими процесами. SCADA є програмним комплексом, призначеним для збирання, обробки та відображення інформації про стан об'єкта управління в режимі реального часу. До складу SCADA входять модулі обміну даними з контролерами через драйвери, модулі обробки інформації, засоби візуалізації у вигляді мнемосхем та графіків, а також підсистеми архівування та генерації звітів. Оператор взаємодіє з системою через людино-машинний інтерфейс, який відображає технологічний процес у зрозумілій формі та надає можливості для аналізу параметрів з відповідним розподілом прав доступу. Архітектура SCADA-системи передбачає розподіл функцій між серверною частиною, що виконує збір та обробку даних, та клієнтськими робочими місцями, які забезпечують візуалізацію та управління (рисунок 1.2).

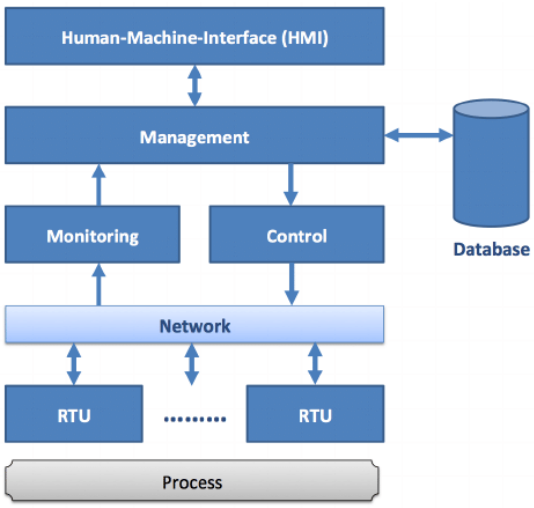

Рисунок 1.2 – Структура SCADA-системи для моніторингу виробництва

Застосування систем моніторингу охоплює різні галузі промисловості. У фармацевтичному виробництві безперервний моніторинг концентрації частинок, температури, вологості та тиску забезпечує дотримання стандартів якості продукції [8]. В енергетиці системи відстежують роботу підстанцій та ліній електропередачі, збираючи дані від розподілених датчиків. На харчових виробництвах моніторинг контрольних точок підтримує вимоги безпеки згідно з міжнародними стандартами. Автомобільна промисловість потребує особливо високої швидкості реагування систем, оскільки затримки у виробничому процесі призводять до значних штрафних санкцій. У хімічній промисловості відстеження параметрів реакцій запобігає виходу процесів за межі безпечних режимів та забезпечує стабільність якості продукції.

Характерною особливістю систем моніторингу є можливість налаштування умов сповіщення про відхилення параметрів від допустимих меж. Коли контрольована величина досягає або перевищує встановлені граничні значення, система активує засоби оповіщення – візуальні сигнали, звукові сповіщення або повідомлення на мобільні пристрої. Сповіщення допомагає обслуговуючому персоналу своєчасно реагувати на проблеми на початковій стадії їх виникнення, що зменшує ризики аварій та простоїв обладнання. Налаштування граничних значень виконується окремо для кожного контрольованого параметра з урахуванням технологічних норм та допусків, встановлених виробником обладнання чи стандартами галузі.

Типовий сценарій використання системи моніторингу на виробництві включає початкову конфігурацію переліку контрольованих параметрів, встановлення датчиків у цільових точках технологічного процесу, налаштування мережевого з'єднання між польовими пристроями та сервером, створення інтерфейсу оператора з мнемосхемами обладнання та запуск процесу збирання даних. У процесі експлуатації оператор спостерігає за поточними значеннями параметрів, реагує на сповіщення про відхилення, переглядає графіки зміни величин у часі та формує звіти про роботу обладнання за визначені періоди. Система забезпечує можливість одночасного моніторингу декількох об'єктів з

єдиного диспетчерського центру, що особливо актуально для територіально розподілених виробництв.

Вибір конкретного рішення для моніторингу визначається низкою факторів, що включають кількість контрольованих точок, вимоги до швидкодії системи, відстані між датчиками та центром обробки, наявність інфраструктури зв'язку, бюджет проекту та потреби у масштабуванні. Для локальних виробничих дільниць з обмеженою кількістю датчиків можуть застосовуватися автономні контролери з вбудованим веб-інтерфейсом. Великі підприємства з розгалуженою структурою потребують розгортання повноцінних SCADA-систем з підтримкою територіально розподіленої архітектури та резервування компонентів.

Сучасні тенденції у розвитку систем моніторингу пов'язані з інтеграцією технологій машинного навчання для прогнозування відмов обладнання на основі аналізу історичних даних, використанням хмарних платформ для централізованого збереження інформації від географічно розподілених об'єктів, а також застосуванням мобільних додатків для віддаленого доступу до даних моніторингу [7-8]. Зростає значення кібербезпеки промислових систем, оскільки підключення до мережі Інтернет створює потенційні загрози несанкціонованого доступу до критичної інфраструктури. Впровадження протоколів шифрування даних, механізмів автентифікації користувачів та сегментації промислових мереж стає обов'язковою складовою проектування систем моніторингу виробничих параметрів.

1.2 Апаратні засоби збору та передавання даних

Збір та передавання даних від виробничих об'єктів до систем обробки вимагає застосування спеціалізованих апаратних компонентів, що включають датчики фізичних величин, мікроконтролерні платформи та засоби комунікації. Вибір конкретних апаратних рішень визначається характером контрольованих параметрів, умовами експлуатації, вимогами до точності вимірювань та способом передавання інформації.

Датчики становлять первинну ланку системи збору даних та виконують перетворення фізичних величин в електричні сигнали. Для моніторингу температури та вологості повітря набули розповсюдження цифрові датчики серії DHT, зокрема DHT11 та DHT22. Датчик DHT11 забезпечує вимірювання температури в діапазоні від 0 до 50 градусів Цельсія з точністю 2 градуси та відносної вологості від 20% до 95% з точністю 5%. Передавання даних здійснюється одним проводом з використанням власного протоколу, а частота опитування становить приблизно один раз на секунду [9]. Датчик DHT22 характеризується вищою точністю – похибка вимірювання температури складає 0,5 градуса, вологості 2%, при цьому діапазон вимірювання вологості розширений до 0-100%. Оновлення даних у DHT22 відбувається раз на дві секунди [9].

Для застосувань, що потребують одночасного контролю температури, вологості та атмосферного тиску, застосовуються інтегровані сенсори типу BME280, які забезпечують високу точність вимірювань та низьке енергоспоживання. Датчики тиску використовуються у системах контролю гідравлічних та пневматичних систем, датчики рівня рідини відстежують наповнення резервуарів, а датчики концентрації газів застосовуються для моніторингу хімічного складу атмосфери у виробничих приміщеннях. Вибір датчика визначається співвідношенням між вимогами до точності, швидкодією, діапазоном вимірювань та вартістю компонента.

Мікроконтролерні платформи виконують функції збору даних від датчиків, первинної обробки інформації та організації передавання до систем верхнього рівня. Платформа Arduino UNO базується на мікроконтролері ATmega328P з тактовою частотою 16 МГц, 32 КБ flash-пам'яті для програм та 2 КБ оперативної пам'яті. Плата забезпечує 14 цифрових входів-виходів, шість з яких підтримують широтно-імпульсну модуляцію, та 6 аналогових входів. Програмування виконується через USB-інтерфейс з використанням середовища Arduino IDE, що спрощує розробку прошивок завдяки наявності готових бібліотек для роботи з датчиками [10]. Перевагою Arduino UNO є простота

освоєння, велика спільнота користувачів та значний обсяг документації, що робить цю платформу придатною для навчальних проектів та прототипування систем моніторингу.

Мікроконтролерна платформа ESP32 представляє розвиток концепції інтернету речей, інтегруючи у одному чіпі двоядерний процесор Xtensa LX6 з тактовою частотою до 240 МГц, 520 КБ оперативної пам'яті, модулі Wi-Fi та Bluetooth. Наявність вбудованих засобів бездротового зв'язку дозволяє створювати автономні пристрої моніторингу, що передають дані до хмарних сервісів або локальних серверів без додаткових комунікаційних модулів. ESP32 підтримує стандарти IEEE 802.11 b/g/n для Wi-Fi та Bluetooth Classic і Bluetooth Low Energy, що забезпечує гнучкість у виборі протоколу передавання даних залежно від вимог до швидкості та енергоспоживання. Програмування ESP32 здійснюється через Arduino IDE або спеціалізоване середовище ESP-IDF, що надає доступ до розширених можливостей мікроконтролера.

Порівняння характеристик Arduino UNO та ESP32 наведено у таблиці 1.1. Вибір між цими платформами визначається специфікою завдання: Arduino UNO придатний для локальних систем з провідним підключенням до комп'ютера, тоді як ESP32 забезпечує автономну роботу з бездротовою передачею даних, що актуально для розподілених систем моніторингу.

Таблиця 1.1 – Порівняння характеристик мікроконтролерних платформ

Характеристика	Arduino UNO	ESP32
Мікроконтролер	ATmega328P	Xtensa LX6 (dual-core)
Тактова частота	16 МГц	до 240 МГц
Flash-пам'ять	32 КБ	4 МБ
Оперативна пам'ять (SRAM)	2 КБ	520 КБ
Цифрові входи-виходи	14	34
Аналогові входи	6 (10-біт АЦП)	18 (12-біт АЦП)
ШІМ виходи	6	16

Продовження таблиця 1.1

Характеристика	Arduino UNO	ESP32
Вбудований Wi-Fi	Ні	Так (802.11 b/g/n)
Вбудований Bluetooth	Ні	Так (Classic + BLE)
Напруга живлення	5 В	3,3 В
Напруга входів-виходів	5 В	3,3 В
Інтерфейси	UART, SPI, I2C, USB	UART, SPI, I2C, USB, CAN
Середовище програмування	Arduino IDE	Arduino IDE, ESP-IDF
Споживання енергії	~50 мА	80-240 мА (активний режим)
Можливість багатозадачності	Обмежена	FreeRTOS (підтримка багатопотоковості)

Передавання даних від мікроконтролера до комп'ютера або іншого пристрою здійснюється через різні інтерфейси зв'язку. Інтерфейс UART забезпечує асинхронну послідовну передачу даних між пристроями і є одним з найпоширеніших способів комунікації в системах збору даних. Arduino UNO використовує UART для зв'язку з персональним комп'ютером через USB-порт, при цьому перетворення сигналів UART в USB здійснює спеціалізована мікросхема-міст. Швидкість передавання даних через UART може варіюватися від 9600 до 115200 біт на секунду залежно від налаштувань програмного забезпечення.

Бездротові способи передавання даних розширюють можливості систем моніторингу, дозволяючи організувати зв'язок з віддаленими або рухомими об'єктами. Модулі Bluetooth, такі як HC-06 або JDY-31, підключаються до мікроконтролера через UART та забезпечують бездротовий зв'язок з комп'ютером або смартфоном на відстані до 10 метрів. Після встановлення з'єднання Bluetooth-модуль емулює віртуальний послідовний порт, що дозволяє використовувати стандартні функції роботи з UART без змін у програмному коді. Напруга живлення Bluetooth-модулів становить 3,3-6 В, споживання струму в режимі передачі досягає 8 мА [11].

Застосування Wi-Fi для передавання даних дозволяє інтегрувати пристрої моніторингу в локальні мережі та забезпечити доступ до них через мережу Інтернет. ESP32 містить вбудований Wi-Fi-модуль, що підтримує роботу як у режимі клієнта, підключаючись до існуючої бездротової мережі, так і в режимі точки доступу, створюючи власну мережу. Для плат Arduino без вбудованого Wi-Fi застосовуються зовнішні модулі ESP8266, які підключаються через UART та керуються AT-командами або працюють як автономні контролери з власною прошивкою. Передавання даних через Wi-Fi відбувається з використанням протоколів TCP/IP, що дозволяє реалізувати різноманітні схеми комунікації – від простого надсилання даних на сервер до організації веб-інтерфейсу для віддаленого доступу.

Вибір інтерфейсу передавання даних залежить від відстані між датчиком та пристроєм збору інформації, наявності інфраструктури зв'язку, вимог до швидкості передавання та енергоспоживання. Провідне підключення через USB або UART забезпечує стабільний зв'язок для стаціонарних установок, розташованих поблизу комп'ютера. Bluetooth придатний для бездротових з'єднань на коротких відстанях, коли потрібна мобільність пристрою або відсутня можливість прокладання кабелів. Wi-Fi застосовується для інтеграції сенсорів у корпоративні мережі та створення систем з віддаленим доступом через Інтернет, що особливо актуально для розподілених виробничих об'єктів або систем диспетчеризації.

Інтеграція датчиків з мікроконтролерами вимагає врахування електричних характеристик компонентів. Більшість цифрових датчиків працюють з логічними рівнями 3,3 В або 5 В, що відповідає напрузі живлення мікроконтролерів Arduino та ESP32. Датчики DHT підключаються до цифрових входів-виходів мікроконтролера трьома проводами – живлення, земля та лінія даних, при цьому між лінією даних та живленням рекомендується встановлювати підтягувальний резистор номіналом 10 кОм для забезпечення стабільності сигналу [9]. Аналогові датчики підключаються до аналогових входів мікроконтролера, які виконують

оцифровування сигналу за допомогою вбудованого АЦП з роздільною здатністю 10 біт для Arduino UNO та 12 біт для ESP32.

Для роботи з датчиками в середовищі Arduino IDE існують готові програмні бібліотеки, що спрощують розробку прошивок. Бібліотека DHT sensor library від Adafruit надає функції для зчитування температури та вологості від датчиків серії DHT, приховуючи деталі протоколу обміну даними. Використання бібліотек дозволяє зосередитися на логіці програми моніторингу, не заглиблюючись у низькорівневі операції з апаратурою. Програмний код завантажується у мікроконтролер через USB-інтерфейс за допомогою бутлоадера, що записаний у flash-пам'ять контролера на етапі виробництва плати.

1.3 Протоколи обміну даними між контролером і ПК

Обмін даними між мікроконтролером та персональним комп’ютером забезпечує можливість передавання інформації, команд керування, результатів вимірювань та інших параметрів системи. Реалізація комунікації здійснюється за допомогою апаратних інтерфейсів і відповідних протоколів передачі даних, які визначають послідовність, формат і синхронізацію сигналів.

Одним із базових методів взаємодії є використання послідовного інтерфейсу UART (Universal Asynchronous Receiver-Transmitter). Цей протокол застосовується для обміну інформацією між контролером і комп’ютером через інтерфейс USB–UART-перетворювача. Передача відбувається асинхронно, без окремого тактового сигналу, при цьому обидва пристрої повинні мати узгоджені параметри швидкості обміну, кількості біт даних, парності та стоп-бітів. Протокол UART є базовим у більшості мікроконтролерних систем, включно з платформами Arduino UNO та ESP32.

Інтерфейс SPI (Serial Peripheral Interface) використовується для синхронної передачі даних між мікроконтролером та периферійними пристроями. Його структура включає лінії передавання, приймання, синхронізації та вибору

пристрою. Така архітектура забезпечує швидкий обмін даними між основним пристроєм і підлеглими вузлами у межах однієї системи.

Протокол I²C (Inter-Integrated Circuit) передбачає використання двох ліній – SCL (Serial Clock Line) та SDA (Serial Data Line) – для послідовного передавання даних між кількома пристроями, підключеними до спільної шини. Мікроконтролер у цьому випадку виступає як основний пристрій (майстер), а датчики або периферія – як підлеглі вузли (слейви). I²C широко застосовується у вбудованих системах, де необхідна передача невеликих обсягів інформації між кількома модулями.

У випадках, коли обмін даними між контролером і комп'ютером здійснюється через мережеве середовище, використовуються протоколи, що функціонують поверх стеку TCP/IP. Одним із поширених рішень є MQTT (Message Queuing Telemetry Transport), який реалізує модель передавання повідомлень «публікація-підписка». У цій структурі контролер може виступати клієнтом, що публікує дані (наприклад, показники датчиків), а персональний комп'ютер або сервер – брокером або підписником, який приймає ці дані для подальшої обробки [12]. Схематично взаємодія з MQTT-брокером наведено на рисунку 1.3.

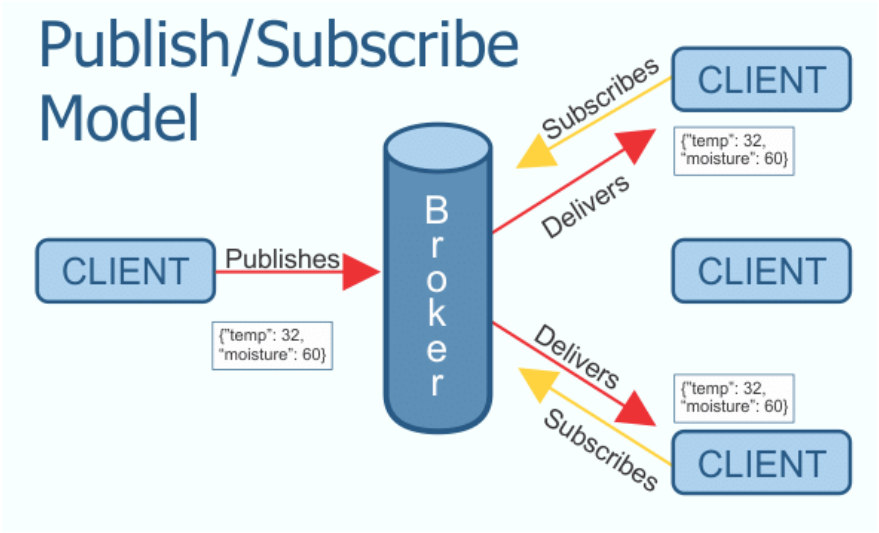

Рисунок 1.3 – Модель «публікація/підписка»

MQTT відзначається простою реалізацією протоколу, мінімальними вимогами до пропускної здатності мережі та підтримкою постійного двостороннього з’єднання між клієнтом і брокером. Завдяки цьому протокол застосовується при розробленні систем збору телеметрії, моніторингу та дистанційного керування, де контролери з бездротовим підключенням, зокрема ESP32 [13], взаємодіють із серверами або персональними комп’ютерами через мережу Інтернет.

Таким чином, вибір протоколу обміну залежить від архітектури системи, доступних апаратних засобів і вимог до способу передавання даних. Для локальної взаємодії між контролером і ПК зазвичай застосовується UART або USB-з’єднання, тоді як для мережевих IoT-систем доцільним є використання MQTT, що працює поверх TCP/IP-стеку.

1.4 Аналіз існуючих рішень моніторингу параметрів середовища та виробничих процесів

На ринку представлено декілька типів рішень для моніторингу параметрів середовища та виробничих процесів, що відрізняються апаратною реалізацією, функціональними можливостями та призначенням. Розгляд існуючих систем дозволяє виявити їх переваги та обмеження, що є основою для формування вимог до програмного забезпечення, що розробляється.

Термогігрометри Testo 608-H1 та Testo 608-H2 представляють категорію автономних цифрових приладів для безперервного моніторингу температури та вологості у приміщеннях. Прилад Testo 608-H1 забезпечує вимірювання температури в діапазоні від 0 до +50°C та відносної вологості від 10% до 95% з точністю ±3%. Модель Testo 608-H2 розширює діапазон температур до від -10°C до +70°C та підвищує точність вимірювання вологості до ±2%, що відповідає вимогам фармацевтичної та харчової промисловості.

Термогігрометри обладнані великим рідкокристалічним дисплеєм, що дозволяє зчитувати показники на відстані. Прилади відображають поточні

значення температури, вологості та температури точки роси, а також зберігають максимальні та мінімальні значення за період експлуатації. Модель Testo 608-H2 додатково має світлодіодну сигналізацію, що спрацьовує при виході параметрів за межі встановлених граничних значень (рисунок 1.4). Живлення здійснюється від батареї з терміном служби до одного року. Прилади кріпляться на стіну та не потребують підключення до комп'ютера або мережі.

Рисунок 1.4 – Термогігрометр Testo 608-H2

Недоліком цих приладів є відсутність можливості дистанційного доступу до даних та автоматичного збереження історії вимірювань у базі даних. Оператор повинен фізично підходити до пристрою для зчитування показників або перевірки стану сигналізації. Термогігрометри Testo призначені для локального моніторингу у фіксованій точці приміщення без централізованого збору інформації від декількох датчиків.

Рішення SenseCAP S2103 являє собою бездротовий датчик для моніторингу концентрації CO2, температури та вологості з використанням протоколу LoRaWAN [15]. Датчик забезпечує вимірювання концентрації вуглекислого газу в діапазоні від 400 до 10000 ppm, температури від -40°C до +85°C та вологості від 0% до 100%. Корпус пристрою має ступінь захисту IP66, що дозволяє розміщувати його у жорстких промислових умовах [16].

Передавання даних здійснюється через мережу LoRaWAN на відстань до 2км у міській забудові та до 10 км на відкритій місцевості. Енергоспоживання

датчика мінімізоване, що забезпечує автономну роботу від батареї терміном до 10 років. Конфігурування датчика та доступ до його налаштувань виконується через Bluetooth-з'єднання зі смартфоном (рисунок 1.5) [16]. Зібрані дані передаються до мережевого сервера LoRaWAN, звідки можуть інтегруватися з хмарними платформами IoT для візуалізації та аналізу.

Рисунок 1.5 – Датчик SenseCAP S2103 з підтримкою LoRaWAN

Система SenseCAP орієнтована на масштабовані розподілені мережі сенсорів та вимагає розгортання інфраструктури LoRaWAN, що включає шлюзи та мережевий сервер. Для невеликих локальних систем моніторингу така архітектура створює надлишкову складність та додаткові витрати. Інтеграція з настільним застосунком потребує налаштування передавання даних з мережевого сервера LoRaWAN до локальної бази даних через проміжні протоколи.

SONOFF TH Elite представляє категорію розумних перемикачів з функцією моніторингу температури та вологості на базі мікроконтролера ESP32 [16]. Пристрій підтримує підключення зовнішніх датчиків через роз'єм RJ9 – датчика температури та вологості THS01, водонепроникного датчика температури DS18B20 або датчика вологості ґрунту MS01. Перемикач обладнаний рідкокристалічним дисплеєм, що відображає поточні значення температури та вологості з оновленням кожні 5 секунд [16].

Основною функцією SONOFF TH Elite є автоматичне увімкнення або вимкнення підключеного електричного обладнання при досягненні заданих порогових значень температури чи вологості. Пристрій підтримує навантаження до 20 А, що дозволяє керувати потужними приладами, такими як обігрівачі, системи теплої підлоги або вентиляційне обладнання (рисунок 1.6) [16]. Через мобільний додаток eWeLink користувач отримує push-повідомлення при зміні стану пристрою та може переглядати історичні дані за останні 6 місяців з точністю до години. Дані можна експортувати у форматі xlsx для подальшого аналізу.

Рисунок 1.6 – Розумний перемикач SONOFF TH Elite з LCD-дисплеєм

Інтеграція з Wi-Fi-мережею дозволяє дистанційно керувати пристроєм та отримувати дані через хмарний сервіс eWeLink. Підтримується взаємодія з платформами розумного дому, зокрема Home Assistant, що надає можливість включення SONOFF TH Elite у складні сценарії автоматизації [16]. Недоліком рішення є залежність від хмарного сервісу виробника для доступу до історичних даних та обмежена можливість інтеграції з власними програмними рішеннями без використання API платформи eWeLink.

Аналіз розглянутих рішень показує, що промислові термогігрометри забезпечують локальний моніторинг з високою точністю, але не підтримують автоматизоване збирання даних у централізовану базу. Системи на базі LoRaWAN придатні для територіально розподілених об'єктів, проте вимагають

складної мережевої інфраструктури. Розумні перемикачі з функцією моніторингу орієнтовані на побутове застосування та автоматизацію управління обладнанням, але залежать від хмарних платформ виробників.

На основі розглянутих рішень формуються вимоги до програмного забезпечення для моніторингу виробничих параметрів. Система повинна забезпечувати збір даних від датчиків температури та вологості, підключених до мікроконтролера, передавання показників через MQTT-протокол до десктопного застосунку без залежності від комерційних хмарних сервісів, збереження історії вимірювань у локальній базі даних з прив'язкою до часових міток, візуалізацію поточних значень та графіків зміни параметрів у часі, а також налаштування граничних значень для сповіщення про відхилення. Програмне забезпечення має підтримувати як роботу з реальними датчиками, так і режим симуляції для демонстрації функціонування системи. Архітектура повинна базуватися на доступних компонентах – мікроконтролерах Arduino або ESP32, цифрових датчиках DHT11/DHT22, публічному MQTT-брокері для передавання даних та десктопному застосунку на платформі C# WPF з інтеграцією бази даних Microsoft SQL Server.

1.5 Висновки до 1 розділу

У першому розділі кваліфікаційної роботи було проаналізовано системи моніторингу виробничих параметрів, їх призначення, структуру та основні принципи функціонування. Встановлено, що моніторинг температури, вологості, концентрації газів та інших фізичних величин є важливою складовою забезпечення безпеки, стабільності та якості виробничих процесів. Розглянуто ієрархічну побудову систем моніторингу, яка включає датчики, мікроконтролери, засоби передавання даних та програмні компоненти верхнього рівня, а також визначено роль кожного з цих елементів у загальній архітектурі.

У ході аналізу апаратних засобів збору та передавання даних досліджено можливості сучасних датчиків і мікроконтролерних платформ, а також провідних і бездротових інтерфейсів зв’язку. Показано, що використання технологій Інтернету речей і бездротових протоколів дозволяє створювати гнучкі та масштабовані системи моніторингу з віддаленим доступом до даних. Проаналізовано протоколи обміну даними між контролером і персональним комп’ютером, за результатами чого обґрунтовано доцільність застосування MQTT як ефективного рішення для передавання телеметрії у системах моніторингу.

Також виконано огляд існуючих комерційних та промислових рішень моніторингу параметрів середовища, що дало змогу виявити їхні переваги та недоліки. Встановлено, що більшість готових систем або обмежені функціонально, або залежать від хмарних сервісів виробників, що знижує гнучкість використання у локальних виробничих умовах. На основі проведеного аналізу сформульовано вимоги до розроблюваної системи моніторингу, які стали підґрунтям для подальшого проєктування апаратної та програмної частин у наступних розділах роботи.

2 ПРОЄКТУВАННЯ СИСТЕМИ МОНИТОРИНГУ ВИРОБНИЦТВА

2.1 Обґрунтування вибору апаратної платформи та датчиків

Вибір апаратної платформи для системи моніторингу виробничих параметрів базується на вимогах до стабільності бездротового каналу зв’язку, обчислювальних можливостей контролера, сумісності з цифровими та аналоговими датчиками, а також доступності програмних інструментів, необхідних для швидкої розробки. Проведений аналіз у межах попередніх етапів проєктування засвідчив, що з погляду співвідношення продуктивності, функціональних можливостей та варіантів підключення сенсорів доцільним є застосування мікроконтролера ESP32. Архітектура цього контролера передбачає інтеграцію двоядерного процесора Xtensa LX6 із тактовою частотою до 240 МГц, внутрішню оперативну пам’ять обсягом 520 КБ, а також вбудовані радіомодулі Wi-Fi стандарту IEEE 802.11 b/g/n і Bluetooth. Завдяки наявності Wi-Fi ESP32 здатний встановлювати зашифровані з’єднання з віддаленими серверами, що відповідає вимогам системи, у якій дані передаються до хмарного брокера HiveMQ через протокол MQTT. Зовнішній вигляд плати ESP32 подано на рисунку 2.1.

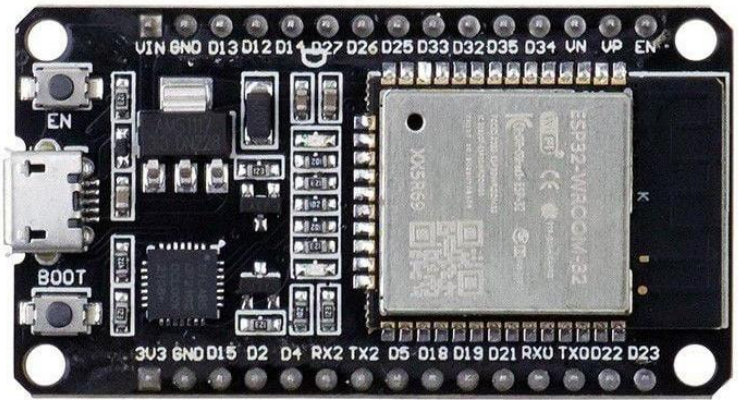

Рисунок 2.1 – Зовнішній вигляд плати ESP32

Кожен окремий вивід ESP32 виконує певну функцію, що впливає на спосіб підключення сенсорів. Серед основних контактів використовуються лінії живлення 3V3 та GND, цифрові входи-виходи GPIO, канали АЦП із перетворенням аналогових сигналів у цифрові коди, а також UART для програмування та налагодження. Підключення датчика MQ-2 здійснюється через один з входів ADC1, наприклад GPIO34, тоді як цифровий датчик DHT22 може працювати з будь-яким виводом GPIO, який допускає приймання сигналу 3,3 В. На рисунку 2.2 наведено схему розташування контактів модуля ESP32, що застосовується у системі.

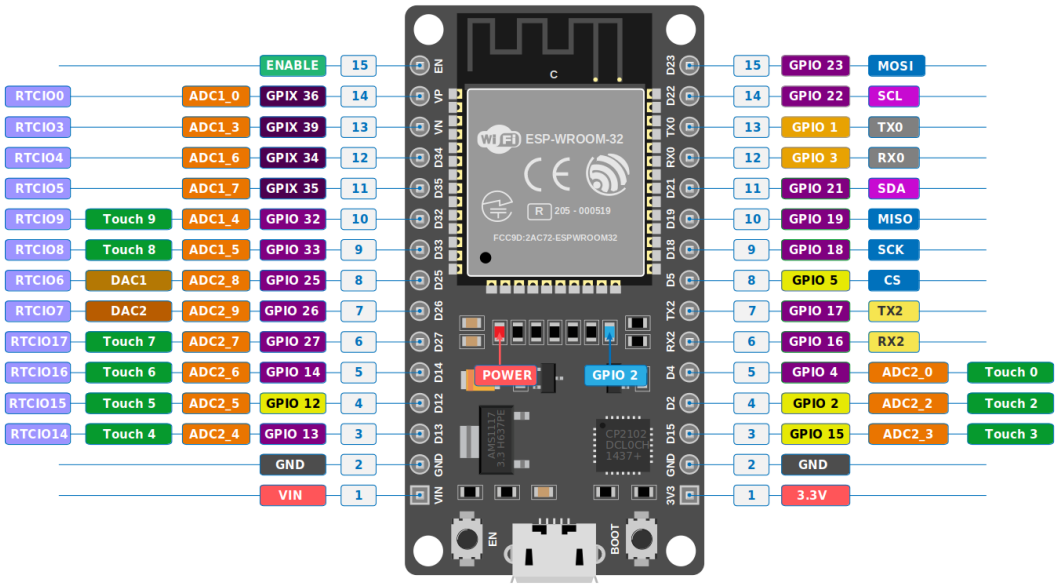

Рисунок 2.2 – Схема розташування контактів ESP32

Програмування контролера виконується у середовищі Arduino IDE, що містить високорівневі бібліотеки для роботи з периферійними модулями та стеком TCP/IP. Підтримка стандартного API дозволяє використовувати доступні бібліотеки для MQTT, цифрових сенсорів та модулів Wi-Fi без розроблення низькорівневих драйверів. Це зменшує складність підготовки прошивки та скорочує час розгортання системи.

Альтернативною платформою, яка часто застосовується у простіших системах, є Arduino UNO на базі ATmega328P. Проте у контексті даного проєкту

можливості Arduino UNO є недостатніми, оскільки для роботи з бездротовою мережею потрібен окремий радіомодуль, що збільшує обсяг апаратної частини та ускладнює логіку взаємодії. Крім того, невеликий обсяг оперативної пам’яті та тактова частота 16 МГц створюють обмеження для одночасної роботи з декількома датчиками та мережевими протоколами. Таким чином, ESP32 має функціональні переваги у порівнянні з AVR-платами та забезпечує достатній ресурс для подальшого розширення системи.

Для вимірювання температури та вологості повітря у системі використовується цифровий датчик DHT22. Пристрій підтримує вимірювання температури у діапазоні від -40°C до +80°C з точністю близько ±0,5°C, а також відносної вологості від 0% до 100% із точністю приблизно ±2%. Передавання даних виконується через один цифровий контакт за допомогою власного протоколу, що спрощує підключення до ESP32. Зовнішній вигляд DHT22 наведено на рисунку 2.3.

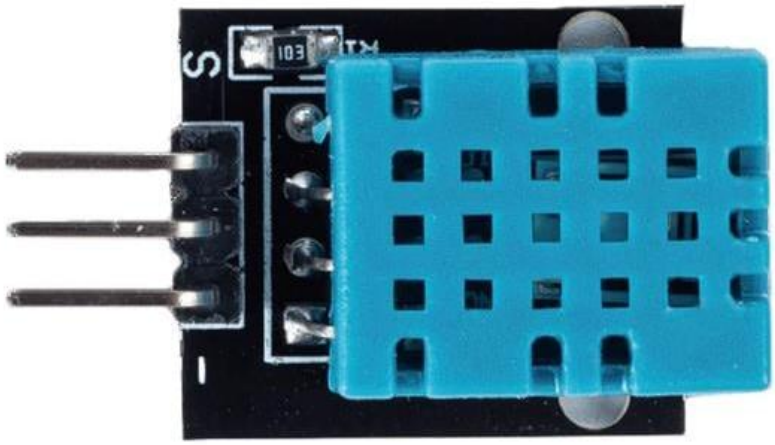

Рисунок 2.3 – Зовнішній вигляд датчика DHT22

Структура контактів DHT22 включає чотири виводи: VCC, DATA, NC та GND. Для передавання сигналу використовується підтягувальний резистор номіналом 10 кОм між DATA та VCC, що забезпечує стабільність рівня сигналу. Схему розташування контактів датчика наведено на рисунку 2.4.

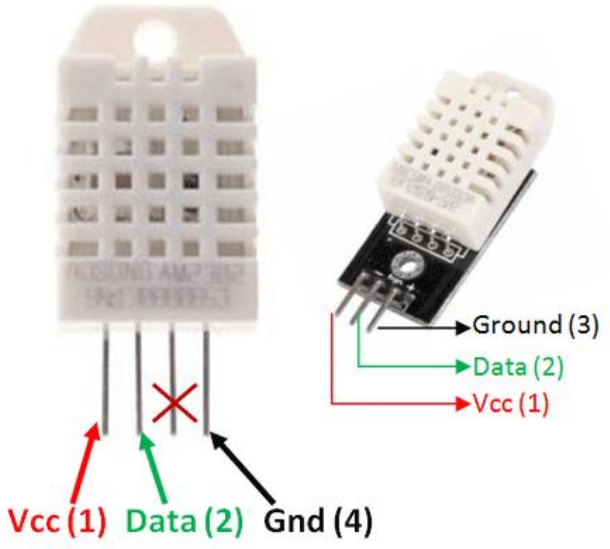

Рисунок 2.4 – Розташування контактів DHT22

Для вимірювання концентрації горючих газів застосовано напівпровідниковий сенсор MQ-2. Датчик реагує на метан, пропан, водень, чадний газ і частинки диму, що робить його придатним для роботи у виробничому середовищі. Робота сенсора базується на зміні опору чутливого шару. Зміна опору перетворюється у напругу, що надходить на аналоговий вхід контролера. Зовнішній вигляд MQ-2 наведено на рисунку 2.5.

Рисунок 2.5 – Зовнішній вигляд сенсора MQ-2

Газові сенсори серії MQ у базовій формі мають шість виводів, які поділяються на дві групи: чутливий елемент та нагрівач. Виводи A1 та A2 є однією стороною чутливого шару, B1 та B2 – іншою. Виводи H1 та H2 використовуються для живлення нагрівача, який створює робочу температуру для сенсорного шару. Розташування контактів MQ-2 подано на рисунку 2.6.

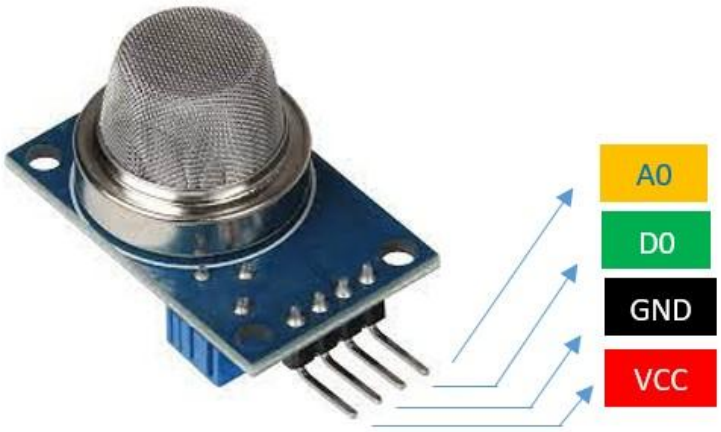

Рисунок 2.7 – Розташування контактів газового сенсора MQ-2

У проєкті використовується модульна версія MQ-2, яка містить аналоговий вихід A0, цифровий вихід D0, компаратор LM393, підлаштовувальний резистор та індикаторні світлодіоди. Наявність компаратора дозволяє реалізувати апаратний пороговий контроль, хоча для точного вимірювання концентрацій використовується саме аналоговий вихід A0. Стан нагрівача потребує врахування при виборі джерела живлення, оскільки струм споживання становить приблизно 150 мА, що є суттєвим навантаженням для малопотужних джерел. Завдяки живленню 5 В модуль можна підключати безпосередньо до живлення плати ESP32 через USB.

2.2 Обґрунтування вибору програмних технологій

Для програмування мікроконтролера ESP32 застосовано інтегроване середовище розробки Arduino IDE. Середовище включає редактор коду з підтримкою підсвічування синтаксису, автоматичного форматування та пошуку тексту, а також засоби компіляції та завантаження програм на плату. Arduino IDE використовує спрощену версію мови C++ та надає стандартизований програмний інтерфейс для роботи з периферійними пристроями мікроконтролера. Компіляція програмного коду відбувається за допомогою інструмента avrdude, що перетворює виконуваний код у текстовий файл у

шістнадцятковому форматі для завантаження через послідовне з'єднання або USB.

Початково Arduino IDE розроблялось для програмування плат на базі мікроконтролерів Atmel AVR, проте згодом підтримка була розширена на інші платформи, включно з ESP32. Для роботи з ESP32 необхідно встановити відповідні пакети підтримки через менеджер плат Arduino IDE, що забезпечує доступ до бібліотек для роботи з Wi-Fi, Bluetooth та периферійними інтерфейсами. Бібліотеки для датчиків DHT22 та MQ-2 доступні через менеджер бібліотек середовища, що дозволяє інтегрувати їх у проект без написання низькорівневого коду драйверів.

Альтернативними середовищами розробки для ESP32 є ESP-IDF – офіційний фреймворк від Espressif Systems, та PlatformIO – розширення для Visual Studio Code. ESP-IDF надає доступ до низькорівневих можливостей мікроконтролера та забезпечує вищу продуктивність, проте потребує глибшого розуміння архітектури ESP32 та більшого обсягу коду для реалізації базового функціоналу. Arduino IDE спрощує процес розробки завдяки абстрагуванню апаратних деталей та наявності готових бібліотек, що скорочує час розробки прошивки для системи моніторингу.

Для передавання даних між мікроконтролером та десктопним застосунком обрано протокол MQTT – відкритий стандарт обміну повідомленнями для Інтернету речей, що реалізує модель публікації-підписки. MQTT призначений для пристроїв з обмеженими ресурсами та мереж з низькою пропускною здатністю, високою затримкою або нестабільним з'єднанням [19]. Протокол визначає два типи мережевих сутностей: брокер повідомлень, що приймає дані від клієнтів-видавців та маршрутизує їх до клієнтів-підписників, та клієнти, що можуть публікувати або підписуватися на топіки. Інформація організована у ієрархію топіків, при цьому видавець не потребує відомостей про кількість або розташування підписників, а підписники не повинні бути налаштовані з даними про видавців.

Мінімальне керуюче повідомлення MQTT може становити лише два байти даних, тоді як максимальний розмір корисного навантаження досягає майже 256 мегабайт. Протокол використовує TCP для передавання даних, а захист інформації забезпечується через TLS-шифрування [20]. MQTT підтримує три рівні якості обслуговування: QoS 0 – доставка максимум один раз без підтвердження, QoS 1 – доставка принаймні один раз з можливістю дублювання, QoS 2 – доставка точно один раз з чотирьохетапним підтвердженням. Для системи моніторингу виробничих параметрів достатнім є рівень QoS 1, що гарантує доставку даних навіть при тимчасових збоях мережі.

Як брокер MQTT застосовано Eclipse Mosquitto – програмне забезпечення з відкритим кодом, що реалізує протокол MQTT версій 5.0, 3.1.1 та 3.1. Mosquitto є компактним та придатним для використання на пристроях від одноплатних комп'ютерів до повнофункціональних серверів [21]. Виконуваний файл брокера займає близько 120 кілобайт, а споживання оперативної пам'яті становить приблизно 3 мегабайти при підключенні 1000 клієнтів. Mosquitto надає інструменти командного рядка mosquitto_pub та mosquitto_sub для публікації та підписки на повідомлення, що дозволяє тестувати функціонування системи без розробки клієнтських застосунків. Брокер підтримує механізми автентифікації користувачів через паролі та TLS-сертифікати, що забезпечує захист даних при передаванні через відкриті мережі.

Альтернативними рішеннями для MQTT-брокера є HiveMQ, EMQX та VerneMQ, що пропонують розширений функціонал для промислового застосування, зокрема кластеризацію, веб-інтерфейс управління та інтеграцію з системами аналітики даних. Проте для локальної системи моніторингу з обмеженою кількістю датчиків та одним десктопним клієнтом функціональності Mosquitto достатньо, а відсутність ліцензійних обмежень дозволяє використовувати його у комерційних проектах без додаткових витрат.

Десктопний застосунок реалізовано на платформі Windows Presentation Foundation – фреймворку для розробки інтерфейсів користувача, що є частиною .NET. WPF використовує Extensible Application Markup Language для

декларативного опису інтерфейсу, що дозволяє відокремити зовнішній вигляд застосунку від його логіки [22]. WPF надає набір функцій для розробки застосунків, що включає елементи керування, прив'язування даних, систему компонування, двовимірну та тривимірну графіку, анімацію, стилі, шаблони та підтримку документів. Програмування бізнес-логіки здійснюється мовою C#, що забезпечує доступ до бібліотек .NET для роботи з мережею, базами даних та файловою системою.

Архітектура WPF базується на концепції властивостей залежності та маршрутизованих подій, що спрощує реалізацію прив'язування даних між інтерфейсом та моделями даних. Механізм прив'язування даних автоматично синхронізує відображення інформації в елементах керування з джерелами даних, що зменшує обсяг коду для оновлення інтерфейсу при зміні стану застосунку. Для візуалізації графіків зміни параметрів у часі можуть застосовуватися бібліотеки сторонніх розробників, зокрема LiveCharts або OxyPlot, що інтегруються з WPF через стандартні елементи керування.

Для зберігання історичних даних вимірювань застосовано систему керування базами даних Microsoft SQL Server. SQL Server Express є безоплатною редакцією, призначеною для розробки та продуктивного використання у настільних, вебових та малих серверних застосунках [23]. Редакція Express має обмеження на використання обчислювальних ресурсів – максимум чотири ядра процесора та 1410 мегабайт оперативної пам'яті для екземпляра бази даних, проте ці обмеження не є критичними для системи моніторингу з одним користувачем та кількома датчиками. Максимальний розмір бази даних становить 10 гігабайт, що достатньо для зберігання мільйонів записів телеметрії.

Взаємодія застосунку з базою даних здійснюється через бібліотеку System.Data.SqlClient, що входить до складу .NET Framework та надає класи для підключення, виконання запитів та обробки результатів. Альтернативним підходом є використання Entity Framework – об'єктно-реляційного відображення, що дозволяє працювати з даними через класи .NET замість написання SQL-запитів. Entity Framework спрощує операції створення, читання, оновлення та

видалення записів, проте вносить додатковий рівень абстракції та може знижувати продуктивність при роботі з великими обсягами даних.

Обрані технології забезпечують реалізацію повного циклу обробки даних у системі моніторингу: прошивка на Arduino IDE зчитує показники датчиків та публікує їх через MQTT-клієнт до брокера Mosquitto, десктопний застосунок на WPF підписується на відповідні топіки, отримує дані та зберігає їх у базі SQL Server Express з прив'язкою до часових міток. Всі компоненти є безоплатними для використання у некомерційних та комерційних проектах, що знижує загальну вартість розробки системи.

2.3 Схема підключення датчиків до контролера

У межах розробки системи моніторингу параметрів середовища застосовано два сенсори: газоаналізатор MQ-2 та датчик температури й вологості DHT22. Обидва сенсори під’єднані до мікроконтролера ESP32, який виконує опитування, сформування вимірювальних даних та передавання їх на MQTT-брокер для подальшої обробки програмним забезпеченням. Для коректної організації роботи сенсорних модулів необхідно визначити послідовність взаємодії між апаратними елементами, а також структуру сигналів, що передаються від сенсорів до контролера. З цією метою формується структурна схема, яка демонструє логіку взаємозв’язків між функціональними вузлами системи, не заглиблюючись у специфіку електричних параметрів окремих з’єднань.

Структурна схема дозволяє узагальнено представити, які елементи беруть участь у зборі даних та в якому напрямку здійснюється передача інформації. Такий підхід дає змогу відокремити функціональний рівень від принципових електричних рішень та створити базову модель, що визначає роль кожного елемента у загальній конфігурації. Зокрема, на схемі відображено канали, через які сенсори передають значення до ESP32, а також модульні складові мікроконтролера, що відповідають за обробку аналогових і цифрових сигналів.

Цей етап опису є підґрунтям для подальшого визначення конкретної електричної схеми підключення.

Побудована структурна схема наведена на рисунку (рисунок 2.8). Вона відображає взаємозв’язки між ESP32, модулем аналого-цифрового перетворювача, цифровим входом контролера та сенсорами MQ-2 і DHT22. Схема демонструє функціональні зв’язки без деталізації електричних параметрів.

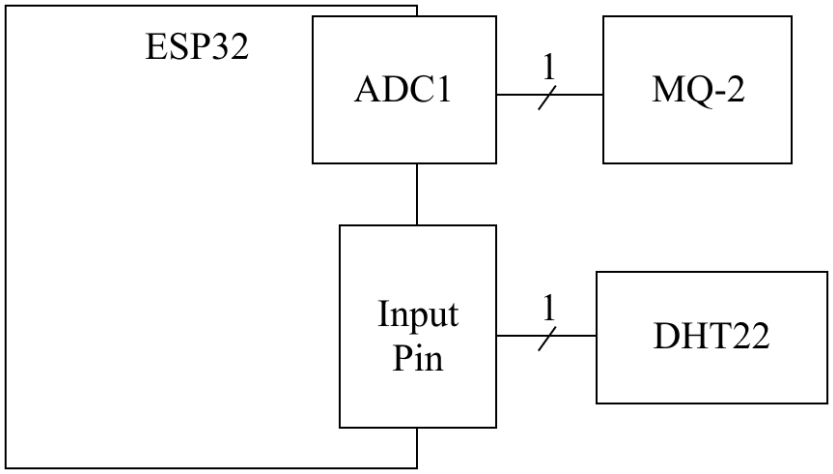

Рисунок 2.8 – Структурна схема підключення ESP32 та датчиків

Принципова схема підключення наведена на рисунку (рисунок 2.9). Вона містить детальну структуру з’єднань між ESP32 та сенсорами, а також елементами, що забезпечують коректне формування аналогових та цифрових сигналів.

Датчик DHT22 під’єднаний до цифрового входу ESP32 через сигнальний вивід DATA. Згідно зі специфікацією сенсора [24], для забезпечення стабільності фронтів сигналу використовується підтягувальний резистор номіналом 10 кОм між виводом DATA та шиною живлення +5 В. Така конфігурація забезпечує коректний обмін даними між контролером та датчиком.

Газовий сенсор MQ-2 підключений відповідно до загальної моделі підключення сенсорів серії MQ. Чутливий елемент має дві пари виводів (A1–A2 та B1–B2), а тепловий елемент живиться через виводи VH+ і VH–.

Вимірювальний струм формується за допомогою подільника напруги, що складається з внутрішнього опору сенсора Rs та зовнішнього резистора навантаження RL номіналом 5,1 кОм (R1). Вихід подільника під’єднано до аналого-цифрового входу ESP32 (A0), що забезпечує зчитування аналогового сигналу відповідно до характеристик АЦП контролера [25].

Для формування цифрового сигналу використано компаратор LM393, який порівнює виміряну напругу із налаштованою пороговою величиною. Опорна напруга формується за допомогою подільника на основі резистора RV1. У випадку перевищення встановленого порогу компаратор генерує логічний сигнал на виході D0. Така конфігурація відповідає конструкції типових модулів MQ-2 із цифровим виходом [26]. Вихід D0 застосовується як додатковий індикатор наявності газу.

Усі елементи схеми живляться від шини +5 В, причому ESP32, MQ-2, DHT22 та компаратор LM393 мають спільну точку заземлення. Це забезпечує коректність роботи як аналогових, так і цифрових ланцюгів.

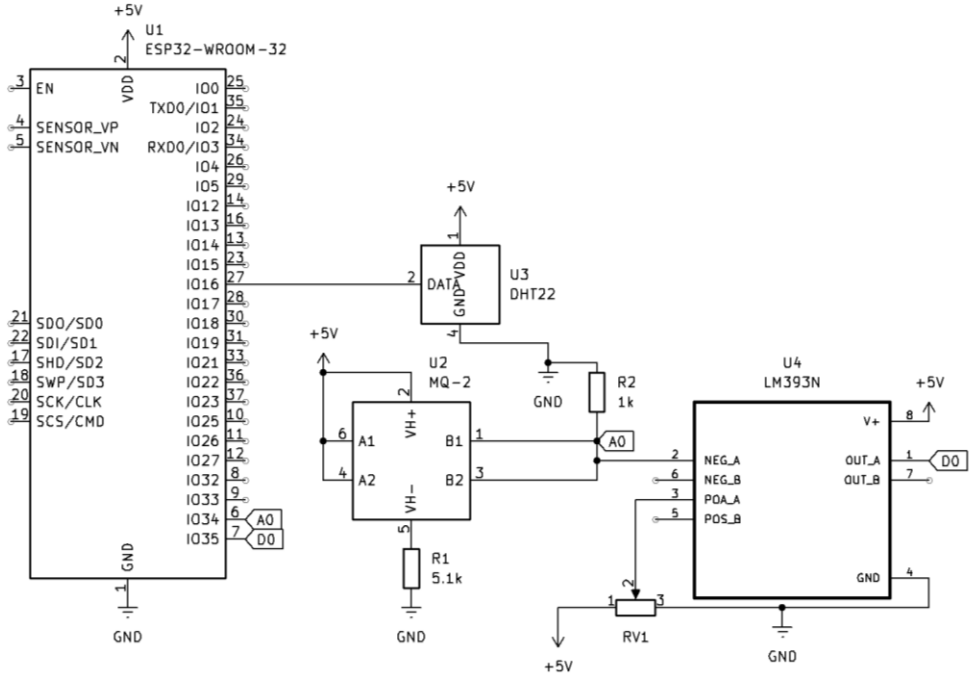

Рисунок 2.9 – Принципова схема підключення ESP32, DHT22 та MQ-2 з компаратором LM393

Застосована конфігурація забезпечує узгоджену взаємодію між ESP32 та датчиками за допомогою окремих інтерфейсів – аналогового, цифрового та силового. Розподіл ролей між компонентами схеми дає можливість формувати вимірювальні дані у вигляді сигналів, придатних для подальшої обробки програмним забезпечення.

2.4 Проєктування бази даних

Схема бази даних призначена для збереження інформації, сформованої контролером ESP32 на основі даних від сенсорів MQ-2 та DHT22, які публікуються через MQTT-брокер. База даних використовується у режимі журнального накопичення даних з можливістю побудови часових вибірок, перегляду значень у режимі реального часу та виконання вибірок за типами сенсорів.

Для зберігання даних передбачено дві таблиці – «SensorReadings» та «Users». Таблиця «SensorReadings» накопичує виміряні значення, а таблиця «Users» забезпечує створення облікових записів і автентифікацію у клієнтському застосунку. Логічна структура схеми представлена на рисунку (рисунок 2.10).

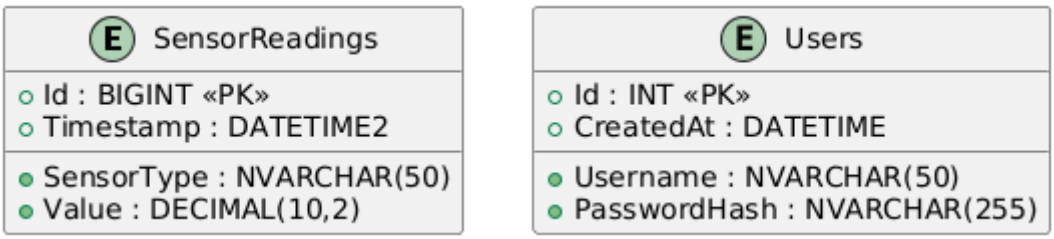

Рисунок 2.10 – ER-діаграма бази даних «SensorMonitoring»

Першою у системі формується таблиця «SensorReadings». Вона виконує роль основного журналу, в якому реєструється інформація про кожне отримане вимірювання. Структура подана у таблиці 2.1. У ній усі поля є атомарними, типи даних обрані відповідно до специфікацій SQL Server [27], а індекси дозволяють

виконувати вибірки за часовими інтервалами та типом сенсора з необхідною швидкістю.

Таблиця 2.1 – Структура таблиці «SensorReadings»

Колонка	Тип	Опис
Id	BIGINT	Первинний ключ (автоінкремент)
SensorType	NVARCHAR(50)	Тип сенсора («MQ-2», «DHT22-Temp» тощо)
Value	DECIMAL(10,2)	Значення параметра
Timestamp	DATETIME2	Дата та час надходження запису

Таблиця «Users» використовується для зберігання інформації про користувачів, які мають доступ до застосунку моніторингу. Структуру наведено у таблиці 2.2. Зберігання хешів паролів виконується відповідно до рекомендацій щодо безпечного зберігання автентифікаційних даних [28].

Таблиця 2.2 – Структура таблиці «Users»

Колонка	Тип	Опис
Id	INT	Первинний ключ (автоінкремент)
Username	NVARCHAR(50)	Унікальне ім’я користувача
PasswordHash	NVARCHAR(255)	SHA-256 хеш пароля
CreatedAt	DATETIME	Дата створення облікового запису

Проєктування структури бази даних виконується так, щоб забезпечити нормалізацію даних та уникнення надлишкових залежностей. Нормалізація підтверджує коректність побудови таблиць за вимогами 1НФ, 2НФ та 3НФ. У першій нормальній формі всі поля мають неподільні значення, а таблиці не містять повторюваних груп. У другій нормальній формі всі неключові атрибути

повністю залежать від первинного ключа, який у таблицях є простим. У третій нормальній формі виключено транзитивні залежності: у таблиці «SensorReadings» атрибути «SensorType», «Value» та «Timestamp» залежать виключно від «Id», а в таблиці «Users» значення залежать лише від первинного ключа. Така структура відповідає загальним вимогам до побудови схем, орієнтованих на журналювання вимірювань та часові ряди [29].

Застосована модель забезпечує поділ ролей між сутностями, не містить дублювання даних і дає можливість масштабувати систему за рахунок додавання нових типів сенсорів або розширення схемою нових таблиць без зміни існуючих залежностей.

2.5 Розробка архітектури системи моніторингу

Архітектура системи моніторингу сформована з урахуванням вимог до збирання, передачі, обробки та візуалізації даних у режимі, що наближений до реального часу. Система побудована за багаторівневим принципом і включає чотири функціональні вузли: сенсорний модуль ESP32, віддалений MQTT-брокер HiveMQ Cloud, серверну частину десктопного застосунку та клієнтську частину, що використовується для відображення даних. Такий підхід відповідає моделям телеметричних систем, у яких застосовується протокол MQTT для відокремлення джерел даних від споживачів [30].

Функції в системі розподілені між вузлами таким чином, щоб забезпечити відсутність взаємної залежності між етапами збору, транспортного обміну та подальшої обробки. ESP32 виконує зчитування даних з сенсорів, обробку вимірювань та формування MQTT-повідомлень. Віддалений брокер приймає повідомлення, утримує їх і передає підписникам згідно з параметрами топіків. Серверна частина десктопного застосунку отримує телеметрію, структурує дані та записує їх у базу даних MSSQL. Клієнтська частина забезпечує візуалізацію, формування графіків і здійснює запити на отримання історичних вибірок.

Роботу системи можна розглядати як послідовність операцій руху інформації між чотирма вузлами. Узагальнена схема потоків даних подана на рисунку (рисунок 2.11). На діаграмі показано, як дані проходять шлях від апаратного рівня (ESP32) до інтерфейсу користувача.

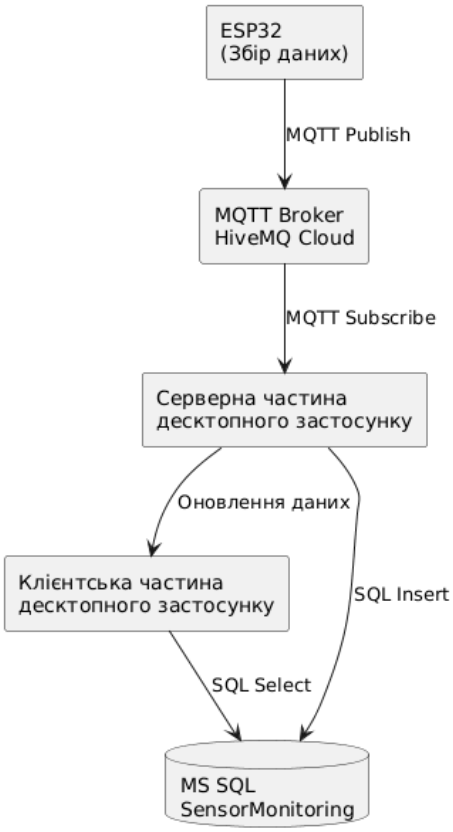

Рисунок 2.11 – DFD-діаграма рівня 0 системи моніторингу

ESP32 формує числові значення параметрів середовища (газовий індекс MQ-2, температура та вологість від DHT22) і публікує їх у вигляді MQTT-повідомлень до HiveMQ Cloud. Формат телеметрії уніфікований і містить значення, часову мітку та тип сенсора. Згідно з специфікацією MQTT, брокер функціонує як проміжна ланка, яка забезпечує доставку повідомлень усім активним підписникам, незалежно від їх розташування або стану з’єднання [31].

Серверна частина десктопного застосунку працює у режимі постійного підключення до топіків брокера та виконує отримання повідомлень. Після прийняття дані перетворюються у структури, придатні до запису у таблиці бази

даних. Механізм обробки включає перевірку типу сенсора, конвертацію числових значень та виконання транзакції у MSSQL Server. Запис у БД створює підґрунтя для побудови часових рядів та виконання аналітичних вибірок.

Клієнтська частина застосунку реалізує інтерфейс візуалізації. Вона виконує запити до БД на отримання останніх значень, формування графіків та аналіз часової динаміки. Взаємодія між серверною та клієнтською частинами передбачає можливість прямого доступу до БД або використання виділеного сервісу для агрегування даних.

На рисунку (рисунок 2.12) подано модель взаємодії компонентів на рівні логічної архітектури, де відображено функції кожного вузла та канали зв’язку між ними.

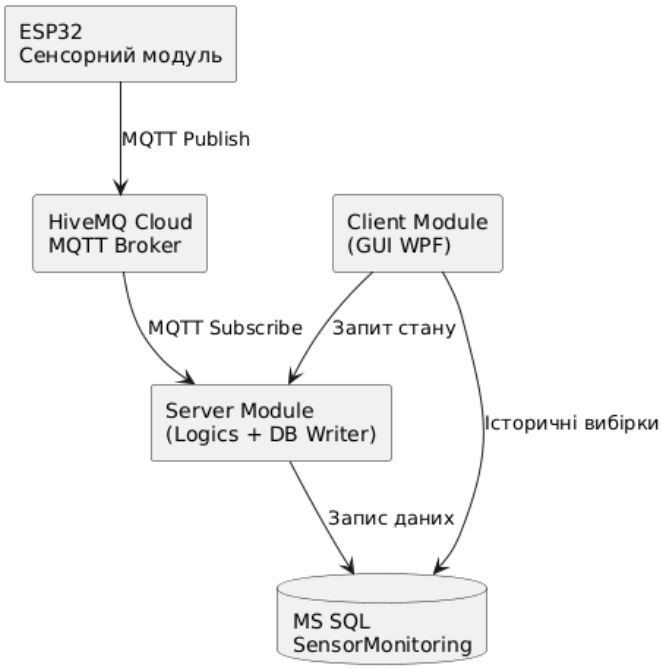

Рисунок 2.12 – Архітектурна діаграма компонентів системи моніторингу

ESP32 передає дані мережею Wi-Fi, використовуючи шифроване з’єднання TLS під час встановлення MQTT-сесії з зовнішнім брокером HiveMQ Cloud. Застосування TLS гарантує цілісність повідомлень у процесі передавання, а також забезпечує автентифікацію кінцевих точок відповідно до політики доступу MQTT-брокера. Для встановлення з’єднання ESP32 використовує MQTT-

клієнтську бібліотеку з підтримкою SSL-сертифікатів, що дає змогу передавати дані без прямої залежності від топології мережі та змін маршрутизації.

Серверна частина десктопного застосунку отримує доступ до MQTT-топіків через протокол TCP поверх TLS/SSL. Під'єднання здійснюється у режимі постійної підписки, тому серверна частина отримує кожне повідомлення одразу після його надсилання брокером. Логіка обробки включає чергу прийнятих повідомлень, механізм повторних спроб у разі втрати з'єднання, а також перетворення структури MQTT-повідомлення у формат, сумісний із моделлю бази даних. Після приймання телеметрії серверна частина виконує операції нормалізації значень, конвертації типів та вирівнювання часових міток із системним часом робочої станції.

Модуль запису у БД взаємодіє з MSSQL через драйвер SqlClient, який забезпечує виконання транзакцій з параметризованими запитами. Така схема дозволяє уникати появи SQL-ін'єкцій і забезпечує стабільність операцій вставки при значній кількості вхідних повідомлень. Для кожного вимірювання створюється окрема транзакція або ж пакет даних, залежно від режиму роботи застосунку. У разі збільшення навантаження можливе використання пулу підключень, що зменшує час встановлення з'єднання з базою даних та оптимізує використання ресурсів.

Клієнтська частина застосунку взаємодіє із серверною у межах одного процесу або окремого контексту програми залежно від конфігурації. У базовому режимі клієнт звертається безпосередньо до MSSQL-бази даних для виконання вибірок і побудови графіків. За необхідності може застосовуватися модель, у якій клієнтська частина взаємодіє з серверною через внутрішній API, що дозволяє ізолювати логіку обробки та реалізувати централізований контроль доступу. У межах клієнтської частини передбачено механізм кешування останніх вибірок для зменшення кількості звернень до бази даних при візуалізації без зміни інтервалів часу.

Порядок обміну даними між вузлами подано на діаграмі послідовності (рисунок 2.13). Вона демонструє покрокову логіку руху повідомлень:

формування телеметрії на ESP32, передавання брокеру HiveMQ Cloud, приймання серверною частиною, запис у базу даних та подальше відображення значень у клієнтському інтерфейсі. Така діаграма дозволяє представити часову послідовність операцій та залежності між вузлами на рівні протоколів, а також чітко відокремлює операції, що виконуються асинхронно, від операцій синхронного отримання даних.

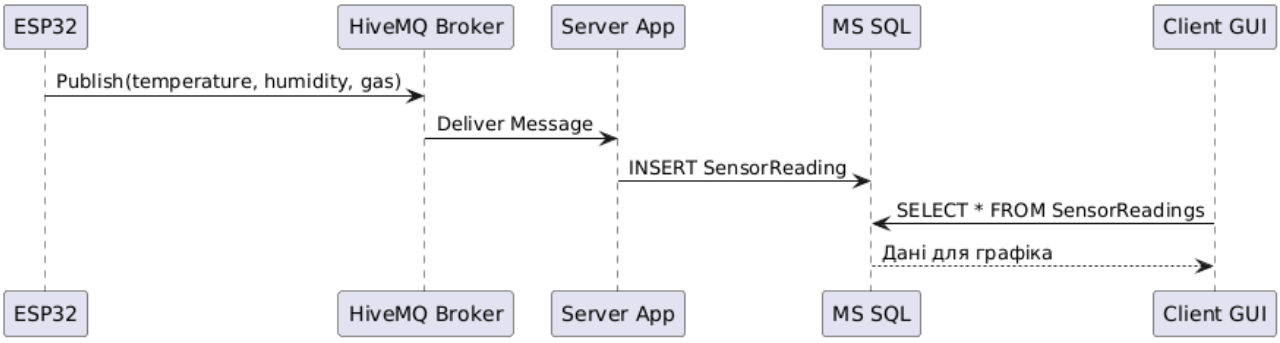

Рисунок 2.13 – Діаграма послідовності передавання показів датчиків

Дана модель дозволяє оцінювати часову структуру системи, визначати місця можливих затримок та точки розширення функціональності. Наприклад, у разі необхідності система може бути масштабована шляхом підключення додаткових ESP32 або введення проміжного сервісу для попередньої обробки даних. На основі цього можливо передбачити різні варіанти розподілу навантаження між серверною частиною, брокером та базою даних.

На рисунку (рисунок 2.14) подано схему розгортання, що описує фізичне розташування компонентів та мережеві з’єднання між ними. Схема відображає, що сенсорний модуль ESP32 розташований у локальній мережі та використовує доступ до Інтернету для встановлення TLS-підключення до HiveMQ Cloud. MQTT-брокер представлений як зовнішній сервіс, який не потребує розгортання на стороні користувача. Серверна частина та клієнтська частина розміщені на робочих станціях або серверах локального підприємства, а база даних MSSQL може бути встановлена локально або на окремому сервері. Мережеві

взаємозв’язки реалізуються через стандартні TCP-з’єднання, що дозволяє розміщувати компоненти у різних фізичних або віртуальних середовищах.

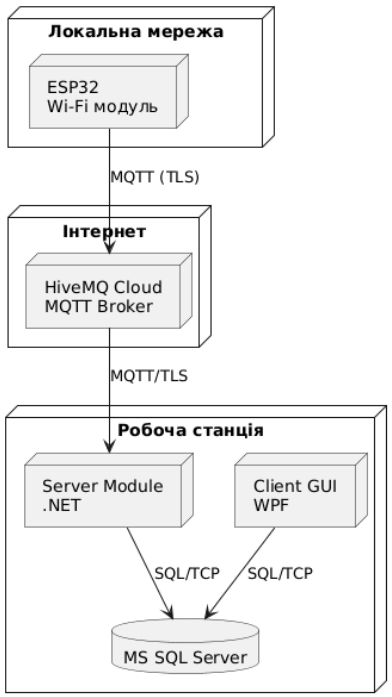

Рисунок 2.14 – Діаграма розгортання системи моніторингу

Така структура дає можливість масштабування шляхом підключення додаткових ESP32, розширення таблиць бази даних, а також створення віддалених клієнтських модулів без модифікації архітектурних рішень.

2.6 Висновки до 2 розділу

У другому розділі виконано проєктування системи моніторингу виробничих параметрів та обґрунтовано вибір її основних апаратних і програмних компонентів. Визначено доцільність використання мікроконтролера ESP32 як центрального елемента системи завдяки наявності вбудованого модуля Wi-Fi, достатній обчислювальній потужності та підтримці сучасних мережевих протоколів. Обґрунтовано вибір датчиків DHT22 і MQ-2, які забезпечують вимірювання температури, вологості та концентрації газів і відповідають вимогам системи за точністю та надійністю.

Також у розділі визначено програмні технології, необхідні для реалізації системи, зокрема використання середовища Arduino IDE для розробки прошивки контролера, протоколу MQTT для передавання даних та платформи C# WPF для створення десктопного застосунку з графічним інтерфейсом. Спроєктовано структуру бази даних та загальну архітектуру системи, що забезпечує збирання, передавання, збереження й візуалізацію даних у режимі реального часу. Результати другого розділу створили основу для подальшої реалізації програмного забезпечення та проведення тестування системи.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕСТУВАННЯ

3.1 Розробка прошивки контролера

Прошивка мікроконтролера ESP32 виконує функції збирання даних із датчиків, їх первинної обробки та передавання на віддалений сервер через протокол MQTT. Архітектура програмного коду передбачає модульну структуру, де кожен функціональний блок відповідає за конкретний аспект роботи системи.

Початкова конфігурація системи включає підключення бібліотек для роботи з мережею та периферійними пристроями. Бібліотека WiFi.h забезпечує з'єднання з бездротовою мережею, WiFiClientSecure.h додає підтримку TLS-шифрування для захищеного обміну даними, PubSubClient.h реалізує клієнтську частину протоколу MQTT, а DHT.h надає інтерфейс для зчитування показників температури та вологості з цифрового датчика DHT22 [31]. Визначення апаратних параметрів здійснюється через директиви препроцесора, що дозволяє централізовано керувати конфігурацією без модифікації основного коду (лістинг 3.1).

Лістинг 3.1 – Підключення бібліотек та визначення апаратних параметрів

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <DHT.h>

#define DHTPIN 16
#define DHTTYPE DHT22
#define MQ2PIN 34
```

Параметри мережевого з'єднання та автентифікації на MQTT-брокері зберігаються у константних рядках. Використання захищеного з'єднання через порт 8883 забезпечує шифрування трафіку відповідно до стандарту TLS, що

запобігає перехопленню даних під час передавання [31]. Топологія MQTT-комунікації передбачає публікацію даних у три окремі топіки, що дозволяє незалежно підписуватись на різні типи вимірювань та забезпечує гнучкість при масштабуванні системи (лістинг 3.2).

Лістинг 3.2 – Конфігурація MQTT-з'єднання та топіків публікації

```
const char* mqtt_server = "**********************************";
const int mqtt_port = 8883;
const char* mqtt_user = "**********";
const char* mqtt_pass = "**********";

const char* topic_temperature = "diploma2025/temperature";
const char* topic_humidity = "diploma2025/humidity";
const char* topic_gas = "diploma2025/gas";
```

Робота з напівпровідниковим газовим датчиком MQ-2 вимагає калібрування та застосування математичної моделі для перетворення значень АЦП у концентрацію газу. Датчик складається з чутливого шару оксиду олова, опір якого змінюється при взаємодії з молекулами горючих газів у навколишньому середовищі [32].

Обчислення опору сенсора Rs здійснюється через вимірювання напруги на навантажувальному резисторі VRL та використання закону Ома для резистивного дільника [33]:

$$R_S = R_L \cdot (\frac{V_C}{V_{RL}} - 1), \quad (3.1)$$

де Vc – напруга живлення схеми датчика (5 В);

VRL – напруга на навантажувальному резисторі;

RL – опір навантажувального резистора (5 кОм);

Напругу VRL визначають через зчитування значення АЦП та перерахунок у вольти з урахуванням опорної напруги АЦП мікроконтролера (3.3 В) та його розрядності (12 біт, що відповідає 4095 дискретним рівням).

Калібрування датчика полягає у визначенні базового опору R0, який відповідає стану сенсора в чистому повітрі без присутності цільових газів. Згідно з технічною документацією виробника, співвідношення Rs до R0 у чистому повітрі становить приблизно 9.83 [35]:

$$R_0 = \frac{R_S}{CleanAitFactor}, \tag{3.2}$$

де CleanAirFactor = 9.83. Процедура калібрування виконується на етапі ініціалізації системи шляхом серії вимірювань (50 зразків) та усереднення результатів для зменшення впливу випадкових флуктуацій.

Перетворення відношення Rs/R0 у концентрацію газу здійснюється через логарифмічну залежність, що апроксимує характеристичну криву чутливості датчика до зрідженого нафтового газу (LPG). Згідно з експериментальними даними виробника, ця залежність має вигляд [32]:

$$log_{10}(PPM) = \alpha \cdot log_{10}\frac{R_S}{R_0} + b, \tag{3.3}$$

де коефіцієнти a та b визначаються з апроксимації експериментальних кривих у напівлогарифмічних координатах. Для LPG використовуються значення a = –0.47 та b = 2.3, отримані методом найменших квадратів на основі точок з графіка чутливості датчика [35].

Реалізація математичної моделі газового датчика у програмному коді поділяється на три функції. Функція mq2_getRs() здійснює перетворення дискретного значення АЦП у опір сенсора Rs згідно з формулою (3.1). Функція mq2_calibrate() виконує процедуру калібрування та обчислює R0 за формулою (3.2), зберігаючи результат у глобальній змінній для подальшого використання. Функція mq2_getLPG_ppm() застосовує логарифмічну апроксимацію (3.3) для розрахунку концентрації газу у частках на мільйон (лістинг 3.3).

Лістинг 3.3 – Функції обробки даних газового датчика MQ-2

```
float mq2_getRs(int adcValue) {
  float vrl = (adcValue * MQ2_ADC_REF) / MQ2_ADC_RES;
  if (vrl <= 0.0) {
    return 1e6;
  }
  float rs = MQ2_RL_KOHM * (MQ2_VC - vrl) / vrl;
  return rs;
}

void mq2_calibrate() {
  Serial.println("Calibrating MQ-2...");
  float rsSum = 0.0;
  const int samples = 50;
  for (int i = 0; i < samples; i++) {
    int adc = analogRead(MQ2PIN);
    float rs = mq2_getRs(adc);
    rsSum += rs;
    delay(100);
  }
  float rsAvg = rsSum / samples;
  mq2_R0 = rsAvg / MQ2_CLEAN_AIR_FACTOR;
  mq2_calibrated = true;
  Serial.print("MQ-2 R0 = ");
  Serial.print(mq2_R0, 2);
  Serial.println(" kOhm");
}

float mq2_getLPG_ppm(int adcValue) {
  if (!mq2_calibrated) {
    return -1;
  }
  float rs = mq2_getRs(adcValue);
  float rs_ro = rs / mq2_R0;
  float logRSR0 = log10(rs_ro);
  float logPPM = (logRSR0 - LPGCurve[1]) / LPGCurve[2] + LPGCurve[0];
  float ppm = pow(10, logPPM);
  return ppm;
}
```

Функція встановлення бездротового з'єднання connectWiFi() ініціює підключення до точки доступу та очікує на встановлення з'єднання у блокуючому режимі. Функція reconnectMQTT() реалізує процедуру підключення до MQTT-брокера з автентифікацією та автоматичним повторенням спроб у разі

невдачі. Генерація унікального ідентифікатора клієнта запобігає конфліктам при одночасному підключенні декількох пристроїв (лістинг 3.4).

Лістинг 3.4 – Функції встановлення мережевих з'єднань

```
void connectWiFi() {
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println(" connected");
}

void reconnectMQTT() {
  while (!client.connected()) {
    Serial.print("Connecting to MQTT...");
    String clientId = "ESP32-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str(), mqtt_user, mqtt_pass)) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Збирання та публікація даних датчиків реалізована у функції publishSensorData(), яка послідовно зчитує показники температури та вологості з DHT22, вимірює концентрацію газу через MQ-2 та публікує отримані значення в окремі MQTT-топіки. Перевірка на коректність зчитаних даних здійснюється через функцію isnan(), що дозволяє виявити помилки зв'язку з датчиком. Параметр retained у функції publish() забезпечує збереження останнього значення на брокері, що дає змогу новим клієнтам одразу отримати поточний стан системи при підписці (лістинг 3.5).

Лістинг 3.5 – Функція збирання та публікації даних датчиків

```
void publishSensorData() {
 float temperature = dht.readTemperature();
 float humidity = dht.readHumidity();
 int gasRaw = analogRead(MQ2PIN);
 float gasPPM = mq2_getLPG_ppm(gasRaw);

 if (!isnan(temperature)) {
  String tempStr = String(temperature, 2);
  client.publish(topic_temperature, tempStr.c_str(), true);
 }

 if (!isnan(humidity)) {
  String humStr = String(humidity, 2);
  client.publish(topic_humidity, humStr.c_str(), true);
 }

 if (gasPPM >= 0) {
  String gasStr = String(gasPPM, 0);
  client.publish(topic_gas, gasStr.c_str(), true);
 }
}
```

Основний цикл програми реалізує неперервний моніторинг стану з'єднань та періодичне збирання даних. Функція setup() виконує одноразову ініціалізацію всіх підсистем, включаючи послідовний інтерфейс для діагностики, датчики, мережеве підключення та калібрування газового сенсора. Функція loop() підтримує активність MQTT-з'єднання через періодичний виклик client.loop() та публікує дані з інтервалом 1000 мілісекунд, що визначається порівнянням поточного часу з часом останньої публікації (лістинг 3.6).

Лістинг 3.6 – Функції ініціалізації та основного циклу програми

```
void setup() {
 Serial.begin(115200);
 dht.begin();
 pinMode(MQ2PIN, INPUT);
 connectWiFi();
 wifiClient.setInsecure();
 client.setServer(mqtt_server, mqtt_port);
 mq2_calibrate();
}
void loop() {
 if (!client.connected()) {
  reconnectMQTT();}
```

Продовження лістингу 3.6

```
client.loop();
unsigned long currentTime = millis();
if (currentTime - lastSendTime >= sendInterval) {
  lastSendTime = currentTime;
  publishSensorData();
}}
```

Виклик wifiClient.setInsecure() у функції setup() дозволяє встановлювати TLS-з'єднання без верифікації сертифіката сервера, що спрощує процес налагодження системи, проте в промисловому середовищі рекомендується використовувати повну верифікацію ланцюжка сертифікатів для забезпечення автентичності брокера [2]. Використання функції millis() для відліку часу замість затримок через delay() забезпечує неблокуючу архітектуру програми, що дозволяє мікроконтролеру паралельно обробляти MQTT-повідомлення та підтримувати стабільність з'єднання навіть під час виконання операцій збирання даних. Така структура програмного коду відповідає принципам побудови вбудованих систем реального часу, де критичним є своєчасна обробка мережевих подій та запобігання втраті з'єднання через тайм-аути протоколу.

3.2 Реалізація MQTT-комунікації з брокером

Взаємодія клієнтського застосунку з MQTT-брокером частково описана в попередньому підрозділі з точки зору публікації даних мікроконтролером. Клієнтська частина системи моніторингу реалізована як настільний застосунок на платформі WPF (Windows Presentation Foundation) з використанням бібліотеки MQTTnet версії 4.x, що забезпечує повну підтримку специфікації MQTT 5.0 та асинхронну модель програмування на основі Task-based Asynchronous Pattern [35].

Архітектура клієнтського застосунку побудована за принципом Model-View-ViewModel (MVVM), що забезпечує розділення бізнес-логіки та представлення даних. Центральним компонентом є клас MqttService, який

інкапсулює всю логіку роботи з MQTT-протоколом та надає інтерфейс для інших компонентів системи через механізм подій C# (лістинг 3.7).

Лістинг 3.7 – Клас MqttService та метод встановлення з'єднання

```
public class MqttService
{
    private IMqttClient? _mqttClient;
    public event Action<SensorReading>? OnDataReceived;
    public event Action<string>? OnStatusChanged;

    public async Task ConnectAsync()
    {
        try
        {
            var factory = new MqttFactory();
            _mqttClient = factory.CreateMqttClient();

            var options = new MqttClientOptionsBuilder()
                .WithTcpServer("9813d476b2c6412da20337b36a2e3c95.s1.eu.hivemq.cloud",
8883)
                .WithClientId($"WPF_Client_{Guid.NewGuid()}")
                .WithCredentials("admin", "Admin123")
                .WithTls(new MqttClientOptionsBuilderTlsParameters
                {
                    UseTls = true,
                    AllowUntrustedCertificates = true,
                    IgnoreCertificateChainErrors = true,
                    IgnoreCertificateRevocationErrors = true
                })
                .WithCleanSession()
                .Build();

            _mqttClient.ApplicationMessageReceivedAsync += OnMessageReceived;
            await _mqttClient.ConnectAsync(options);
            OnStatusChanged?.Invoke("Підключено до MQTT");

            await _mqttClient.SubscribeAsync("diploma2025/temperature");
            await _mqttClient.SubscribeAsync("diploma2025/humidity");
            await _mqttClient.SubscribeAsync("diploma2025/gas");
        }
        catch (Exception ex)
        {
            OnStatusChanged?.Invoke($"Помилка: {ex.Message}");
        }
    }
}
```

Метод ConnectAsync() реалізує асинхронне встановлення з'єднання з хмарним MQTT-брокером HiveMQ Cloud через захищений канал TLS на порту 8883. Використання патерну Builder для конфігурації параметрів з'єднання дозволяє декларативно визначити всі необхідні налаштування, включаючи адресу сервера, облікові дані та параметри шифрування [35]. Генерація унікального ідентифікатора клієнта через Guid.NewGuid() забезпечує можливість одночасного підключення декількох екземплярів застосунку без конфліктів на рівні протоколу. Параметр WithCleanSession() вказує брокеру не зберігати стан сесії між підключеннями, що спрощує логіку роботи клієнта, оскільки всі підписки встановлюються заново при кожному з'єднанні.

Після успішного встановлення з'єднання виконується підписка на три топіки, що відповідають різним типам вимірювань. Асинхронна природа методу SubscribeAsync() дозволяє виконувати підписки послідовно без блокування основного потоку виконання застосунку. Обробка вхідних повідомлень здійснюється через делегат, який прив'язується до події ApplicationMessageReceivedAsync інтерфейсу IMqttClient (лістинг 3.8).

Лістинг 3.8 – Обробник вхідних MQTT-повідомлень

```
        private Task OnMessageReceived(MqttApplicationMessageReceivedEventArgs e){
            try{
                var topic = e.ApplicationMessage.Topic;
                var payload = Encoding.UTF8.GetString(e.ApplicationMessage.PayloadSegment);
                OnStatusChanged?.Invoke($"Отримано: {topic} = {payload}");
                if (decimal.TryParse(payload, NumberStyles.Any, CultureInfo.InvariantCulture, out
var value)) {
                    var sensorType = topic switch{
                        "diploma2025/temperature" => "Temperature",
                        "diploma2025/humidity" => "Humidity",
                        "diploma2025/gas" => "Gas",
                        _ => "Unknown"
                    };
                    var reading = new SensorReading{
                        SensorType = sensorType,
                        Value = value,
                        Timestamp = DateTime.Now
                    };
                    SaveToDatabase(reading);
                    OnDataReceived?.Invoke(reading);
                    OnStatusChanged?.Invoke($"Оброблено: {sensorType} = {value}");
```

Продовження лістингу 3.8

```
        }else{
            OnStatusChanged?.Invoke($"Не вдалося розпарсити: {payload}");
        }
    }
    catch (Exception ex) {
        OnStatusChanged?.Invoke($"Помилка обробки: {ex.Message}"); }
    return Task.CompletedTask;}
```

Обробка вхідних повідомлень передбачає декодування корисного навантаження з UTF-8 та розбір рядкового представлення числа у тип decimal. Використання CultureInfo.InvariantCulture при парсингі забезпечує коректну обробку чисел з крапкою як десятковим роздільником незалежно від регіональних налаштувань операційної системи, що критично для міжнародної сумісності [36]. Оператор switch expressions у C# 8.0 дозволяє компактно виконати маршрутизацію назви топіка до відповідного типу датчика.

Модель даних SensorReading представляє окреме вимірювання датчика та включає ідентифікатор запису, тип датчика, виміряне значення та мітку часу. Використання типу decimal замість float або double обумовлено необхідністю точного представлення десяткових дробів без похибок округлення, що властиві форматам з плаваючою комою IEEE 754 [37]. Тип DateTime зберігає мітку часу отримання даних на стороні клієнта, що дозволяє враховувати затримки мережі при аналізі часових рядів (лістинг 3.9).

Лістинг 3.9 – Модель даних вимірювання датчика

```
namespace SensorMonitor.Models
{
    public class SensorReading
    {
        public long Id { get; set; }
        public string SensorType { get; set; } = string.Empty;
        public decimal Value { get; set; }
        public DateTime Timestamp { get; set; }
    }
}
```

Збереження отриманих вимірювань у реляційній базі даних реалізовано через Entity Framework Core – об'єктно-реляційний mapper (ORM), що забезпечує абстракцію доступу до даних та автоматичне перетворення між об'єктами C# та записами SQL [38]. Клас SensorDbContext успадковує DbContext та визначає набори сутностей, що відображаються на таблиці бази даних (лістинг 3.10).

Лістинг 3.10 – Контекст бази даних Entity Framework Core

```
using Microsoft.EntityFrameworkCore;
using SensorMonitor.Models;

namespace SensorMonitor.Data
{
    public class SensorDbContext : DbContext
    {
        public DbSet<SensorReading> SensorReadings { get; set; }
        public DbSet<User> Users { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer(
"Server=localhost;Database=SensorMonitoring;Trusted_Connection=True;TrustServerCertificate
=True;");
        }
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<SensorReading>(entity =>
            {
                entity.HasKey(e => e.Id);
                entity.Property(e => e.SensorType).HasMaxLength(50).IsRequired();
                entity.Property(e => e.Value).HasPrecision(10, 2);
                entity.Property(e => e.Timestamp).HasDefaultValueSql("GETDATE()");
                entity.HasIndex(e => e.Timestamp);
                entity.HasIndex(e => new { e.SensorType, e.Timestamp });
            });

            modelBuilder.Entity<User>(entity =>
            {
                entity.HasKey(e => e.Id);
                entity.Property(e => e.Username).HasMaxLength(50).IsRequired();
                entity.Property(e => e.PasswordHash).HasMaxLength(255).IsRequired();
                entity.Property(e => e.CreatedAt).HasDefaultValueSql("GETDATE()");
                entity.HasIndex(e => e.Username).IsUnique();
            });
        }
    }
}
```

Метод OnConfiguring() визначає рядок підключення до Microsoft SQL Server, що працює локально з використанням автентифікації Windows (Trusted_Connection). Параметр TrustServerCertificate дозволяє встановлювати з'єднання без перевірки сертифіката сервера, що прийнятно для розробки, проте в промисловому середовищі рекомендується використовувати валідні сертифікати [37]. Метод OnModelCreating() містить конфігурацію схеми бази даних через Fluent API, що дозволяє декларативно визначити первинні ключі, обмеження, типи даних та індекси.

Створення індексів на полях Timestamp та комбінованого індексу на парі (SensorType, Timestamp) забезпечує швидкий доступ до даних при виконанні типових запитів – фільтрація за часовим діапазоном та вибірка вимірювань конкретного датчика. Композитний індекс оптимізує запити виду "вибрати всі вимірювання температури за останню годину", оскільки СУБД може використовувати індекс для обох умов WHERE одночасно [37].

Метод SaveToDatabase() виконує збереження об'єкта SensorReading у базу даних через створення екземпляра контексту, додавання сутності до колекції та виклику SaveChanges(), що генерує та виконує відповідний SQL-запит INSERT (лістинг 3.11).

Лістинг 3.11 – Метод збереження даних у базу

```
private void SaveToDatabase(SensorReading reading)
{
    try
    {
        using var db = new SensorDbContext();
        db.SensorReadings.Add(reading);
        db.SaveChanges();
    }
    catch (Exception ex)
    {
        OnStatusChanged?.Invoke($"Помилка БД: {ex.Message}");
    }
}
```

Використання конструкції using забезпечує автоматичне вивільнення ресурсів підключення до бази даних після завершення операції, що відповідає патерну Dispose у .NET [7]. Обробка виключень дозволяє продовжити роботу застосунку навіть у разі тимчасової недоступності бази даних, повідомляючи користувача про проблему через механізм подій.

Інтеграція служби MQTT з графічним інтерфейсом реалізована через клас MainWindow, що представляє головне вікно застосунку. Встановлення MainViewModel як DataContext дозволяє використовувати механізм data binding для автоматичної синхронізації стану ViewModel з візуальними елементами інтерфейсу [40]. Асинхронна ініціалізація у обробнику події Loaded забезпечує завантаження даних та встановлення з'єднань після повного завершення створення вікна (лістинг 3.12).

Лістинг 3.12 – Головне вікно застосунку та ініціалізація ViewModel

```
using System.Windows;
using SensorMonitor.ViewModels;

namespace SensorMonitor
{
    public partial class MainWindow : Window
    {
        private readonly MainViewModel _viewModel;

        public MainWindow()
        {
            InitializeComponent();
            _viewModel = new MainViewModel();
            DataContext = _viewModel;
            Loaded += MainWindow_Loaded;
        }

        private async void MainWindow_Loaded(object sender, RoutedEventArgs e)
        {
            await _viewModel.InitializeAsync();
        }
    }
}
```

Розділення відповідальності між класами MqttService, SensorDbContext та MainViewModel забезпечує модульність архітектури, де кожен компонент

виконує чітко визначену функцію. Така структура спрощує тестування окремих модулів та подальше розширення функціональності системи без необхідності модифікації існуючого коду.

3.3 Тестування системи моніторингу

Тестування системи моніторингу виконувалося з метою перевірки коректної роботи апаратної частини, передавання телеметрії через MQTT-брокер та функціонування клієнтського застосунку, який здійснює зчитування, обробку та візуалізацію даних. Випробування проводилися у симуляційному середовищі Wokwi, де апаратна конфігурація ESP32, датчика MQ-2 та датчика DHT22 відтворюється у цифровій моделі. Це забезпечило можливість отримати повторювані результати, керувати зміною параметрів датчиків та спостерігати за реакцією всієї системи без використання фізичного обладнання.

Першим етапом було виконано запуск симуляції у Wokwi. На цьому етапі перевірялася коректність підключення сенсорів до ESP32, робота підтягувального резистора на лінії даних DHT22 та стабільність генерації аналогових значень з MQ-2. У процесі роботи контролер здійснював зчитування даних, формував MQTT-повідомлення та публікував їх на віддалений брокер HiveMQ. У нижній частині інтерфейсу Wokwi виводилися поточні значення від датчика MQ-2 у вигляді сирих показів ADC, а також обчислений рівень газу в ppm. На рисунку 3.1 представлено фрагмент симуляції та частину логів, що підтверджують формування телеметрії.

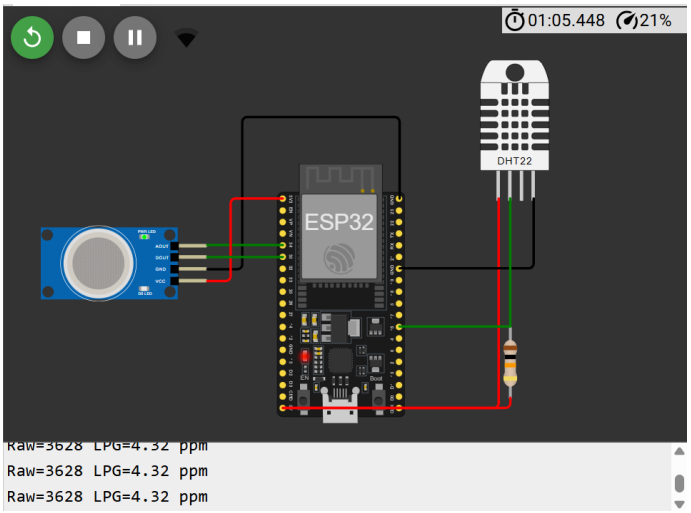

Рисунок 3.1 – Робота системи у середовищі Wokwi та виведення даних датчика MQ-2

Після публікації повідомлень у брокер виконувалася перевірка коректності приймання даних за допомогою програми MQTT Explorer. Цей інструмент використовується для спостереження за ієрархією топіків, перегляду вмісту отриманих значень та контролю стабільності каналу зв'язку. На рисунку 3.2 показано відображення трьох телеметричних параметрів: температури, вологості та рівня газу. Наявність усіх опублікованих полів підтвердила коректне функціонування MQTT-клієнта в прошивці ESP32 та стабільне встановлення TLS-з'єднання з HiveMQ Cloud.

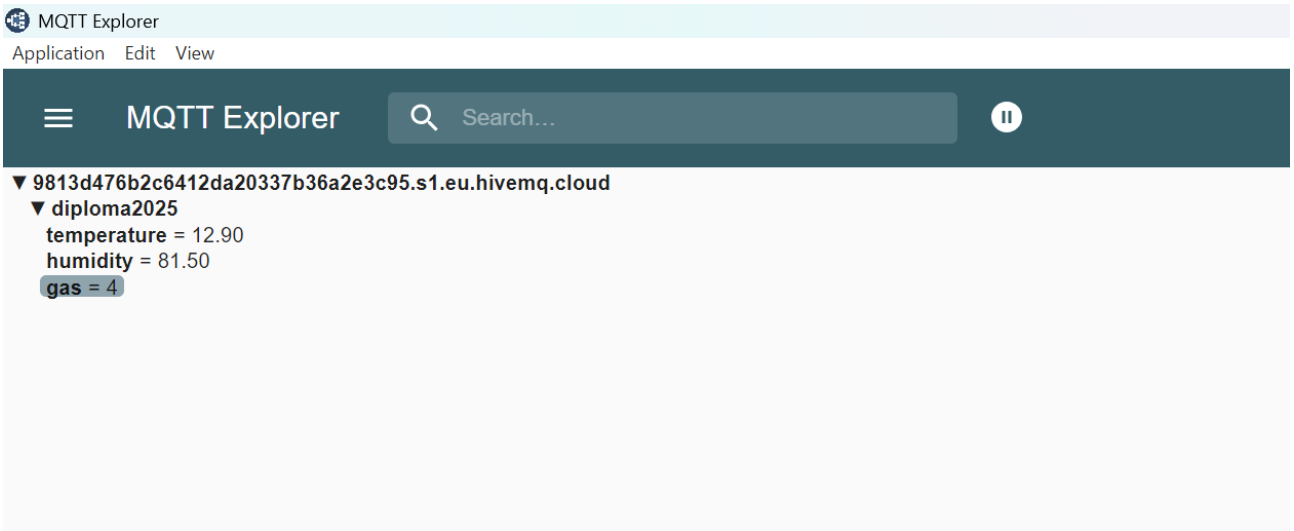

Рисунок 3.2 – Перегляд телеметрії у MQTT Explorer

Після перевірки роботи брокера здійснювалося тестування серверної та клієнтської частини моніторингового застосунку. Першим кроком була перевірка модуля авторизації, оскільки доступ до інтерфейсу здійснюється через систему облікових записів. На рисунку 3.3 наведено форму авторизації, яка забезпечує введення імені користувача та пароля. У процесі тестування було перевірено реєстрацію нового користувача, створення запису в таблиці «Users» та подальше успішне входження в систему.

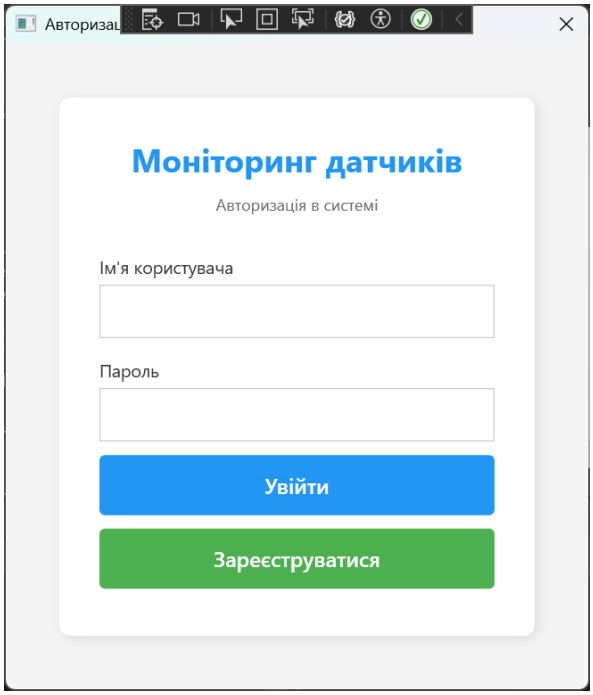

Рисунок 3.3 – Вікно авторизації клієнтського застосунку

Після входу до системи відкривається основний дашборд, який містить числові відображення трьох сенсорних параметрів та три графіки, сформовані на основі даних із бази. Тестування показало, що застосунок успішно опрацьовує отримані через MQTT повідомлення, записує їх у таблицю «SensorReadings» та паралельно оновлює інтерфейс користувача. На рисунку 3.4 подано фрагмент роботи дашборда, де виведено фактичні значення температури, вологості та концентрації газу, а також графічні відображення динаміки змін параметрів.

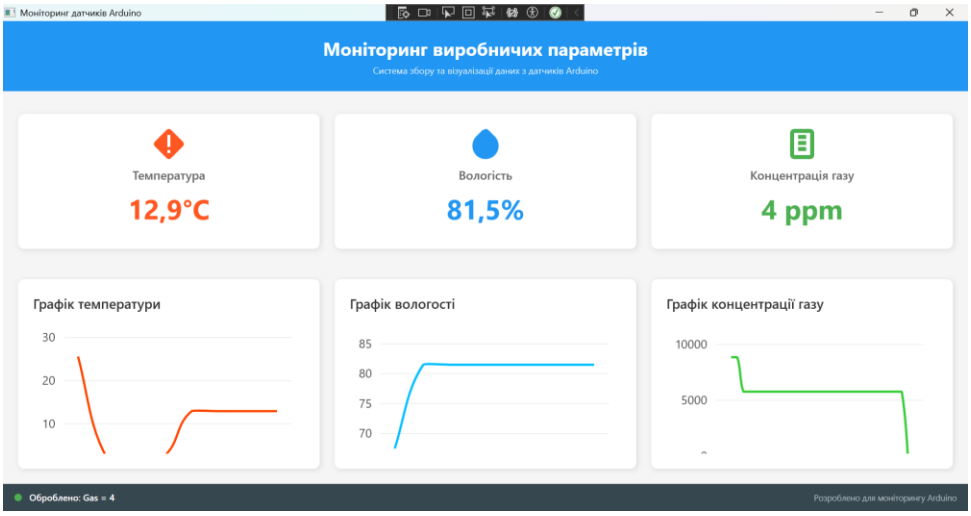

Рисунок 3.4 – Відображення телеметрії та графіків у дашборді моніторингу

Окремим етапом було тестування реакції клієнтського застосунку на зміну параметрів датчиків у режимі реального часу. У симуляції Wokwi передбачена можливість змінювати температуру та вологість на DHT22 за допомогою повзунків. При зміні значень у симуляторі нові покази негайно відправлялися до MQTT-брокера, після чого серверна частина застосунку зчитувала їх і відображала у графічному інтерфейсі. Такий підхід дозволив оцінити затримку між моментом зміни параметра та його появою на дашборді. На рисунку 3.5 наведено одночасне відображення змінених показів DHT22 у Wokwi та відповідну реакцію системи моніторингу.

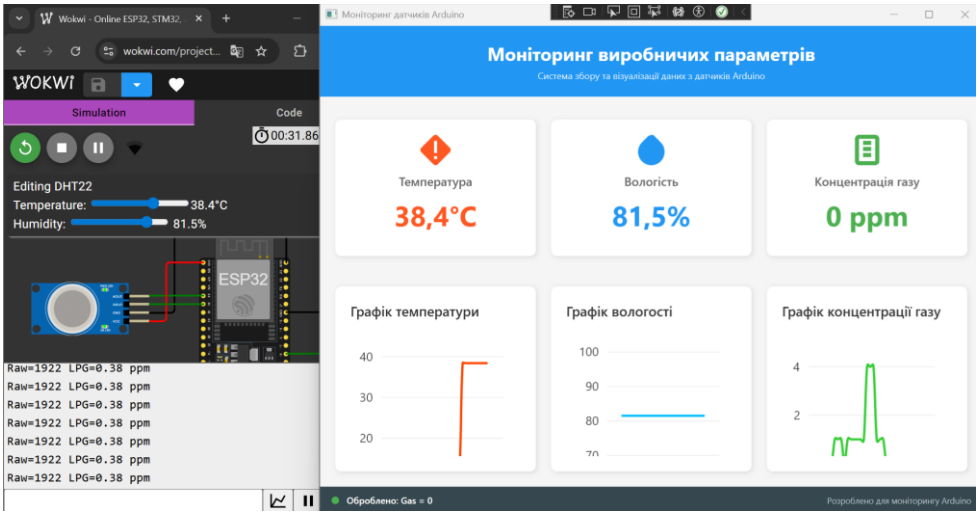

Рисунок 3.5 – Синхронна робота Wokwi-сенсора та дашборда моніторингу

Проведене тестування показало стабільне функціонування всіх елементів архітектури: апаратної частини в симуляції, каналу передавання даних через MQTT, запису в базу даних, а також коректного відображення значень у клієнтському застосунку. Така послідовність перевірок дозволяє підтвердити працездатність системи та її готовність до адаптації для роботи з фізичним обладнанням.

3.4 Висновки до 3 розділу

У третьому розділі було реалізовано програмне забезпечення системи моніторингу виробничих параметрів та виконано його тестування. Розроблено прошивку мікроконтролера ESP32, яка забезпечує коректне зчитування даних з датчиків температури, вологості та концентрації газів, їх первинну обробку та передавання до MQTT-брокера. Реалізація клієнтської частини MQTT підтвердила можливість стабільної передачі телеметрії в режимі реального часу з урахуванням особливостей бездротового з’єднання.

У межах розділу реалізовано десктопний застосунок на платформі C# WPF, який виконує підписку на відповідні MQTT-топіки, приймання повідомлень від контролера, збереження отриманих даних у базі даних Microsoft SQL Server та їх візуалізацію у вигляді таблиць і графіків. Реалізовано механізм прив’язки даних, що забезпечує автоматичне оновлення інтерфейсу користувача при надходженні нових вимірювань, а також підтримку режиму симуляції для демонстрації роботи системи без підключення фізичних датчиків.

Проведене тестування системи підтвердило коректність обміну даними між усіма компонентами, стабільність роботи MQTT-комунікації та правильність збереження показників у базі даних з прив’язкою до часових міток. Отримані результати свідчать про працездатність і надійність розробленого програмного забезпечення та його готовність до подальшої експлуатації або розширення функціональних можливостей системи.

4 ЕКСПЛУАТАЦІЯ ТА ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Інструкція користувача

Перед початком роботи необхідно запустити застосунок та перейти до форми авторизації. У відповідних полях вводиться ім’я користувача та пароль, після чого активується кнопка входу до системи. У випадку відсутності облікового запису використовується процедура реєстрації, що передбачає створення нового запису у системі з подальшим переходом до форми авторизації.

Після успішного входу користувач потрапляє на головну панель моніторингу. На ній відображаються актуальні покази датчиків, що надходять через MQTT-брокер у вигляді температури, відносної вологості та концентрації газу у повітрі. Оновлення значень виконується автоматично відповідно до частоти формування телеметрії з контролера. На панелі розташовані числові значення кожного параметра та графічні елементи для їх візуального відображення.

Окремо надається можливість перегляду графіків за певний період часу. Для цього користувач може перемикати вкладки або використати фільтри, якщо вони передбачені конфігурацією застосунку. Дані для графіків зчитуються з бази даних «SensorMonitoring», у якій містяться записи, отримані під час роботи контролера. Графіки відображають тенденцію зміни параметрів та можуть бути використані для аналізу динаміки у виробничому середовищі.

У нижній частині інтерфейсу передбачено інформаційний рядок, який відображає повідомлення про оброблені дані, статус з’єднання із серверною частиною та іншу службову інформацію. Це дає змогу контролювати стан взаємодії між застосунком, брокером і базою даних.

Для завершення роботи користувач може закрити застосунок стандартним засобом операційної системи. Якщо необхідно змінити користувача, доступна операція виходу з облікового запису. Після виходу користувач повертається на

форму авторизації, звідки можна виконати повторне входження або створити новий обліковий запис.

У разі використання системи у середовищі з постійним надходженням даних рекомендується перевіряти стабільність підключення до мережі та правильність роботи брокера. При зупинці надходження даних користувач може виконати повторний запуск застосунку або перевірити налаштування мікроконтролера ESP32.

Описаний порядок взаємодії з інтерфейсом забезпечує доступ до вимірювань, перегляд поточних показів, аналіз часових рядів та здійснення базових операцій роботи із системою моніторингу.

4.2 Охорона праці та техніка безпеки

Розроблення та експлуатація системи моніторингу, що включає використання комп’ютерної техніки, програмного забезпечення та електронних компонентів, потребує дотримання вимог охорони праці. Організація робочого місця розробника має враховувати ергономічні параметри, які зменшують фізичне навантаження під час тривалої роботи за комп’ютером. Робоче місце повинно мати стіл достатньої площі, стійке крісло з можливістю регулювання, а монітор слід розташовувати так, щоб верхня частина екрана знаходилася на рівні очей користувача. Освітлення робочої зони повинно забезпечувати рівномірний розподіл світла та запобігати появі відблисків на поверхні монітора.

Під час тривалої роботи з комп’ютером може виникати втома зору та м’язове перенавантаження. Для зниження фізичного навантаження доцільно робити перерви протягом дня, виконувати вправи для очей та розминку м’язів рук, шиї та спини. Дотримання режиму перерв позитивно впливає на стан опорно-рухового апарату та допомагає підтримувати працездатність протягом робочого дня.

Робота з електронними компонентами, зокрема з мікроконтролерами, датчиками та допоміжними модулями, вимагає дотримання правил безпечної

взаємодії з електрообладнанням. Під час монтажу електронних схем необхідно забезпечити вимкнення живлення, уникати коротких замикань та стежити за правильністю підключення контактів. Використання стабільних джерел живлення та дотримання номінальних напруг є необхідною умовою для запобігання пошкодженню обладнання та виникненню аварійних ситуацій. Робоче місце має бути обладнане нековзною поверхнею для розміщення електронних компонентів, а маніпуляції з проводами слід виконувати акуратно, щоб уникнути перегинів та порушення ізоляції.

Під час роботи з комп’ютерною технікою та електронними пристроями виникає невеликий рівень електромагнітного та електростатичного випромінювання. З цієї причини рекомендується дотримуватися відстані між користувачем і монітором не менше ніж 50 сантиметрів. Приміщення повинно провітрюватися, щоб забезпечити нормальний повітрообмін та комфортні умови для тривалої роботи.

Пожежна безпека є окремим аспектом охорони праці. Електронне обладнання повинно підключатися через справні електричні розетки та кабелі. У робочому приміщенні має бути доступ до засобів пожежогасіння, а персонал повинен бути ознайомлений із планом евакуації та правилами дій у разі загоряння. Усі пристрої, що не використовуються, слід вимикати від мережі після завершення роботи.

Крім фізичної безпеки, у процесі розробки та експлуатації програмного забезпечення слід приділяти увагу захисту інформації. Робочі станції, серверна частина застосунку та база даних мають бути захищені механізмами автентифікації, паролями та обмеженням доступу. Застосування користувачів повинно функціонувати таким чином, щоб доступ до даних отримували лише уповноважені особи. Системи мають містити засоби контролю за входом у систему, а паролі повинні зберігатися у зашифрованому вигляді.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено програмно-апаратну систему для моніторингу виробничих параметрів на основі мікроконтролерної платформи ESP32, сенсорів температури, вологості та газу, а також комплексу програмних засобів для передавання, зберігання та візуалізації даних. У ході роботи було виконано повний цикл створення системи – від аналізу предметної області до тестування готового рішення.

На етапі аналізу було досліджено особливості моніторингу параметрів виробничого середовища та визначено вимоги до системи збору даних. Проведено огляд апаратних засобів і сенсорних модулів, що застосовуються у технологічних процесах, а також виконано порівняльний аналіз протоколів передавання даних між контролером і персональним комп’ютером. Дослідження існуючих рішень дало змогу визначити структуру майбутньої системи та сформувати вимоги до її функціональних властивостей.

Під час проєктування системи було обґрунтовано вибір мікроконтролера ESP32, датчиків DHT22 та MQ-2, а також програмних технологій, необхідних для реалізації комунікації через MQTT-протокол та опрацювання даних у десктопному застосунку. Розроблено структурну та принципову схеми підключення сенсорів, що визначають конфігурацію апаратної частини. Спроєктовано базу даних «SensorMonitoring», яка забезпечує зберігання отриманих показників разом із часовими мітками та підтримує виконання запитів для побудови графіків і перегляду історії вимірювань. Визначено архітектуру системи моніторингу, що включає вузол контролера, хмарний MQTT-брокер, серверну та клієнтську частини застосунку.

На етапі реалізації створено прошивку контролера, яка забезпечує зчитування даних з сенсорів, обробку аналогового та цифрового сигналу, формування структурованих MQTT-повідомлень і їх передавання на віддалений брокер HiveMQ. Реалізовано механізм захищеного з’єднання через TLS, що дає змогу передавати телеметрію у зашифрованому вигляді. У десктопному

застосунку реалізовано приймання повідомлень, запис інформації до бази даних, формування графічних відображень та роботу інтерфейсу користувача. Додатково впроваджено можливість роботи у режимі симуляції, що дозволяє демонструвати функціонування системи без фізичних сенсорів.

Проведене тестування охоплювало симуляцію роботи апаратної частини у середовищі Wokwi, перевірку коректності передавання даних через MQTT-брокер та роботу програмного забезпечення під навантаженням. У процесі випробувань підтверджено стабільність взаємодії між компонентами системи, коректність збереження показників, узгодженість роботи графічного інтерфейсу та відповідність фактичних результатів очікуваній логіці роботи.

Отримана система забезпечує збирання, передавання, зберігання та відображення параметрів середовища у режимі реального часу, що дозволяє використовувати її як основу для впровадження засобів контролю виробничих процесів. Архітектура рішення дає можливість розширення функціональності, підключення додаткових сенсорів та адаптації для різних галузей, у яких потрібне безперервне відстеження стану середовища.

Результати роботи підтверджують досягнення поставленої мети та виконання всіх завдань, визначених у вступі. Розроблена система може використовуватися як базова платформа для подальшого вдосконалення засобів моніторингу та інтеграції в інформаційні системи підприємств.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – К.: ДП «УкрНДНЦ», 2016. – 31 с.

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка освітньо професійних програм: «Комп'ютерно-інтегровані технологічні процеси і виробництва»; «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2024. 57 с.

3. Системи і датчики для контролю параметрів зовнішнього середовища. URL: https://xreferat.com/38/2286-1-sistemi-datchiki-dlya-kontrolyu-parametr-v-zovn-shn-ogo-seredovisha.html (дата звернення: 30.10.2025)

4. Датчики технологічних параметрів. URL: https://helpiks.org/8-85172.html (дата звернення: 30.10.2025)

5. SCADA система / DIGITAP. URL: https://digitap.com.ua/scada-systema-shho-cze-take/ (дата звернення: 01.11.2025)

6. Інтернет речей у промисловості: як це працює? Kyivstar Business Hub. URL: https://hub.kyivstar.ua/articles/internet-rechej-u-promislovosti-yak-cze-praczyuye (дата звернення: 01.11.2025)

7. Впровадження системи нагляду SCADA zenon – вигоди для кожної галузі / Quantum. URL: https://quantum-int.com/news/vprovadzhennya-sistemi-naglyadu-scada-zenon-vigodi-dlya-kozhnoyi-galuzi/ (дата звернення: 01.11.2025)

8. Порівняння датчиків DHT11, DHT22 та BME280: характеристики, приклади коду та вибір. URL: https://myproject.com.ua/porivniannia-datchykiv-temperatury-ta-volohosti-dht11-vs-dht22-vs-bme280.html (дата звернення: 02.11.2025)

9. Мікроконтролери. URL: https://e-lab.com.ua/microcontrollers.html (дата звернення: 02.11.2025)

10. ESP32 проти Arduino: Який мікроконтролер ви повинні вибрати?. URL: https://ua.allelcoelec.com/blog/ESP32-vs-Arduino-Which-Microcontroller-Should-You-Choose.html (дата звернення: 03.11.2025)

11. MQTT Version 5.0 Protocol Specification. OASIS Standard, 2020.

12. Berman D., Kurniawan A. Internet of Things with ESP32: Build IoT Applications Using ESP32 and MQTT. Apress, 2021.

13. Цифровий гігрометр testo 608-H2, термогігрометр. URL: https://www.testo.kiev.ua/ua/testo-608-n2.html (дата звернення: 04.11.2025)

14. SenseCAP S2103 – Датчик CO2, температури та вологості – LoRaWAN. URL: https://magazin-apelsin.net/ua/p1868453927-sensecap-s2103-datchik.html (дата звернення: 05.11.2025)

15. Espressif Systems. ESP32 Technical Reference Manual. URL: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf (дата звернення: 06.11.2025)

16. Датчик газу MQ-2. URL: https://arduino.ua/prod1115-datchik-gaza-mq-2 (дата звернення: 06.11.2025)

17. OASIS Standard. MQTT Version 5.0. 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (дата доступу: 08.11.2025)

18. Mishra B. Performance Evaluation of MQTT Broker Servers. Lecture Notes in Computer Science. 2020. Vol. 12024. P. 599–609.

19. Eclipse Foundation. Eclipse Mosquitto Documentation URL: https://mosquitto.org/documentation/ (дата доступу: 08.11.2025)

20. Microsoft Learn. Windows Presentation Foundation Documentation. 2024. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/ (дата доступу: 08.11.2025)

21. Microsoft SQL Server. SQL Server 2022 Express Edition URL: https://www.microsoft.com/en-us/sql-server/sql-server-downloads (lata доступу: 08.11.2025)

22. Random Nerd Tutorials. DHT11/DHT22 Temperature and Humidity Sensor for ESP32: Wiring and Code Examples, 2021. URL: https://randomnerdtutorials.com

23. OneTransistor. Attempts at Reading Data from MQ-2 Gas Sensor, 2022. URL: https://www.onetransistor.eu/2022/12/read-mq-sensor-resistance-arduino.html

24. Instructables. ESP32 Smoke Detector Project With MQ-2 Sensor, 2024. URL: https://www.instructables.com/ESP32-Smoke-Detector-Project-With-MQ-2-Sensor.

25. Microsoft Docs. SQL Server Data Types and Table Design Guidelines, 2022. URL: https://learn.microsoft.com

26. OWASP Foundation. Password Storage Cheat Sheet, 2023. URL: https://owasp.org

27. IBM Knowledge Center. Database Normalization Guidelines, 2021. URL: https://www.ibm.com

28. Adafruit Industries. DHT Sensor Library Documentation. 2023. URL: https://github.com/adafruit/DHT-sensor-library (дата звернення: 13.11.2025).

29. Banks A., Briggs E., Borgendale K., Gupta R. MQTT Version 5.0. OASIS Standard. 2019. 137 p. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (дата звернення: 13.11.2025).

30. Hanwei Electronics. MQ-2 Semiconductor Sensor for Combustible Gas: Technical Data. 2020. 4 p.

31. Horowitz P., Hill W. The Art of Electronics. 3rd ed. Cambridge: Cambridge University Press, 2015. 1220 p.

32. STEMpedia. MQ Gas Sensors: Calibration and Usage Guide. 2021. URL: https://thestempedia.com/tutorials/mq-gas-sensors/ (дата звернення: 13.11.2025).

33. Cremers C., Dehnel-Wild M., Milner K. Secure Authentication in the Grid: A formal analysis of DNP3: SAv5. Journal of Computer Security. 2020. Vol. 28. P. 763–801.

34. Albahari J., Albahari B. C# 10 in a Nutshell: The Definitive Reference. Sebastopol: O'Reilly Media, 2022. 1061 p.

35. IEEE Standard for Floating-Point Arithmetic: IEEE Std 754-2019 (Revision of IEEE 754-2008). IEEE, 2019. 84 p.

36. Smith J., Lerman J. Programming Entity Framework Core. 2nd ed. Sebastopol: O'Reilly Media, 2021. 538 p.

37. Ramakrishnan R., Gehrke J. Database Management Systems. 3rd ed. New York: McGraw-Hill, 2003. 1065 p.

38. MacDonald M. Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5. 4th ed. Berkeley: Apress, 2012. 1031 p.