НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Інформаційна система моніторингу та керування кліматом офісного приміщення з використанням мобільного робота»

Виконав:
студент IV курсу, групи ІК-11
Левенко Олексій Вячеславович _____

Керівник:
доцент, кандидат технічних наук
Ткач Михайло Мартинович _____

Рецензент:
доцент, ктм, доцент каф. ІПІ
Лісовиченко Олег Іванович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Левенку Олексію Вячеславовичу

1. Тема проєкту «Інформаційна система моніторингу та керування кліматом офісного приміщення з використанням мобільного робота», керівник проєкту Ткач Михайло Мартинович, доцент, к.т.н., затверджені наказом по університету від «23» травня 2025 р. № 1705-с

2. Термін подання студентом проєкту: «9» червня 2025 року.

3. Вихідні дані до проєкту: об'єкт проєктування має забезпечувати високий рівень точності регулювання мікроклімату в приміщеннях, із мінімальними відхиленнями в розмірі до 5% від заданих параметрів температури, вологості та якості повітря. Система повинна оперативно реагувати на зміни умов та виявлені відхилення. Клімат-контроль має працювати автономно протягом тривалого часу, забезпечуючи надійну і безперебійну роботу без постійного втручання оператора.

4. Зміст пояснювальної записки: проаналізувати предметну область та поставити задачі, сформувати вимоги до системи та обрати технології розробки, реалізувати програмне забезпечення, випробувати та оцінити ефективність системи.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): схема симулятора даних датчиків, схема роботи програми контролю клімату, схема архітектури системи, ER-діаграма бази даних

6. Дата видачі завдання: «7» березня 2025 року.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Аналіз предметної області	15.04.2025	
2	Огляд офісних приміщень	22.04.2025	
3	Огляд існуючих реалізацій	29.04.2025	
4	Постановка задачі	06.05.2025	
5	Вибір засобів та технологій проєктування	13.05.2025	
6	Реалізація програмного забезпечення	20.05.2025	
7	Тестування системи	27.05.2025	
8	Оформлення документації проєкту	04.06.2025	

Студент Олексій ЛЕВЕНКО

Керівник Михайло ТКАЧ

АНОТАЦІЯ

Інформаційна система моніторингу та керування кліматом офісного приміщення з використанням мобільного робота

Проєкт містить 63 с. тексту, 27 рисунків, 1 таблицю, посилання на 25 літературні джерела, додатки та 4 конструкторських документів.

КЕРУВАННЯ КЛІМАТОМ, МОБІЛЬНИЙ РОБОТ, ОФІСНЕ ПРИМІЩЕННЯ, РЕГУЛЯЦІЯ, РОБОТОТЕХНІКА, СИСТЕМА.

Об’єктом розробки є автоматизована система керування кліматом в офісному приміщенні.

Мета розробки — підвищення продуктивності підприємства шляхом покращення комфорту його працівників.

У дипломному проєкті розроблено систему керування кліматом офісного приміщення з використанням сучасних технологій та інструментів. Для збору даних з датчиків та передачі його на контролер було використано протокол MQTT, а сама передача відбувається по бездротовому протоколу Wi-Fi. Контроль системою виконує мікроконтролер ESP32-S3. Програмне забезпечення контролера написано з використанням фреймворку мови програмування C ESP-IDF. Зберігання даних про систему відбувається в базі даних MySQL. Програма, що симулює дані з датчиків, написана на мові програмування Python.

Отримані результати можуть бути корисними для подальшого розвитку систем автоматизованого клімат-контролю, що може знайти застосування в різних комерційних і громадських об'єктах, таких як офісні центри, торговельні приміщення, медичні установи, навчальні заклади та інші.

SUMMARY

Information system for monitoring and controlling the climate of an office space using a mobile robot

The project contains 63 pages of text, 27 figures, 1 table, references to 25 literature sources, appendices and 4 design documents.

Keywords: climate control, mobile robot, office space, regulation, robotics, system.

The object of development is an automated climate control system in an office space.

The purpose of the development is to increase the productivity of the enterprise by improving the comfort of its employees.

In the diploma project, an office space climate control system was developed using modern technologies and tools. The MQTT protocol was used to collect data from sensors and transmit it to the controller, and the transmission itself takes place via the Wi-Fi wireless protocol. The system is controlled by the ESP32-S3 microcontroller. The controller software is written using the ESP-IDF C programming language framework. System data is stored in a MySQL database. The program that simulates data from sensors is written in the Python programming language.

The results obtained may be useful for the further development of automated climate control systems, which may be used in various commercial and public facilities, such as office centers, retail premises, medical institutions, educational institutions, and others.

№ рядка	Формат	Позначення	Найменування	Кіл. аркушів	№ екз.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІК11.180БАК.006 ПЗ	Пояснювальна записка	63		
6						
7	А3	ІК11.180БАК.006 Д1	Інформаційна система моніторингу	1		
8			та керування кліматом офісного			
9			приміщення з використанням			
10			мобільного робота			
11			Алгоритм програми симулятора			
12			даних датчиків			
13						
14	А3	ІК11.180БАК.006 Д2	Інформаційна система моніторингу	1		
15			та керування кліматом офісного			
16			приміщення з використанням			
17			мобільного робота			
18			Алгоритм програми контролю			
19			клімату			
20						
21	А3	ІК11.180БАК.006 Д3	Інформаційна система моніторингу	1		
22			та керування кліматом офісного			
23			приміщення з використанням			
24			мобільного робота			
25			Схема архітектури системи			
26						
27						

					ІК11.180БАК.006 ТП		
Зм.	Лист	№ докум.	Підпис				
Розробив		Левенко			Інформаційна система моніторингу та керування кліматом офісного приміщення з використанням мобільного робота. Відомість дипломного проєкту	Літ.	Арк. 1 / Аркушів 2
Перевірив		Ткач				Т	
Затв.						КПІ ім. Ігоря Сікорського Група ІК-11	

№ рядка	Формат	Позначення	Найменування	Кіл. аркушів	№ екз.	Примітка
28	А3	ІК11.180БАК.006 Д4	Інформаційна система моніторингу	1		
29			та керування кліматом офісного			
30			приміщення з використанням			
31			мобільного робота			
32			ER-діаграма бази даних			
33						
34						
35						
36						
37						
38						
39						
40						
41						
42						
43						
44						
45						
46						
47						
48						
49						
50						
51						
52						
53						
54						
55						
56						
57						
58						
59						

Пояснювальна записка
до дипломного проєкту
на тему:
«Інформаційна система моніторингу та керування кліматом офісного приміщення з використанням мобільного робота»

Київ – 2025 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

BLE – Bluetooth Low Energy;

HVAC – Heating, Ventilation, & Air Conditioning;

I/O – Input and Output;

IoT – Internet of Things.

ВСТУП

Дипломний проєкт присвячений розробці модуля керування мікрокліматом для автоматизованих систем. Автоматизація таких процесів є актуальною в промисловості, сільському господарстві, «розумних» будинках і на об’єктах критичної інфраструктури. Зростання вимог до енергоефективності, комфорту та ресурсозбереження підсилює значущість теми.

Суть роботи — створення мобільного модуля, що автоматично регулює температуру, вологість, вентиляцію та інші параметри мікроклімату відповідно до змін довкілля. Такий модуль може застосовуватись як у побуті, так і в промисловості, сприяючи економії енергії й підвищенню комфорту. Особливо корисною система буде у середовищах, де потрібен стабільний клімат, таких як теплиці, серверні, лабораторії тощо.

Проєкт базується на таких передумовах:

— зростаюча потреба в автономних кліматичних системах;

— необхідність автоматизованого моніторингу параметрів середовища;

— відсутність доступних універсальних рішень;

— потреба у компактних, легко інтегрованих модулях.

Попри наявність готових рішень (Nest, Ecobee, Tado), вони переважно орієнтовані на побут і мають обмежений функціонал або високу вартість. Метою проєкту є створення доступного, гнучкого модуля, який можна адаптувати для офісів, виробництва, аграрного сектору чи навчальних закладів.

Розробка зумовлена відсутністю рішень, що поєднують точність, адаптивність і простоту інтеграції. Система дозволяє в реальному часі збирати й аналізувати кліматичні дані та керувати обладнанням згідно з заданими умовами.

Вибір теми обумовлений сучасними вимогами до екологічності та швидкої адаптації технічних систем. Система має реагувати на зміни навколишнього середовища та підлаштовуватись під конкретні умови.

Об'єкт дослідження — система контролю клімату.

Предмет дослідження — розробка апаратної та програмної частин блоку управління кліматом, а також алгоритмів його роботи.

Мета роботи — підвищення продуктивності підприємства шляхом покращення комфорту його працівників.

Для досягнення цієї мети необхідно вирішити наступні завдання:

— аналіз існуючих рішень та технологій в сфері кліматичного контролю;

— розробка технічної концепції та вибір необхідних компонентів;

— розробка алгоритмів керування та програмного забезпечення;

— проведення тестування та оцінка ефективності роботи модуля;

— підготовка до впровадження в реальні умови.

Практичне значення полягає у створенні універсального модуля, який легко інтегрується в більші системи або використовується автономно. Проєкт може слугувати основою для подальших досліджень у сфері IoT та енергоефективності.

Основні результати:

— модуль з датчиками температури, вологості, якості повітря і виконавчими елементами;

— алгоритми автоматичного регулювання;

— результати тестування, що підтверджують стабільну та ефективну роботу системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Підтримка сприятливих умов є важливою частиною забезпечення високої продуктивності працівників на підприємстві, в тому числі і в офісних приміщеннях. Ці умови формуються в тому числі й мікрокліматом – сукупністю параметрів внутрішнього середовища приміщення, які впливають на самопочуття, здоров’я та продуктивність людини, а також на збереження обладнання, меблів чи продукції. Основними параметрами є температура повітря, його вологість та швидкість руху, рівень забрудненості, температура поверхонь та освітленість. Впливати на ці параметри можна багатьма способами, як і звичайним провітрюванням приміщення, так і радіаторним опаленням та кондиціонерами. Хоча ці способи дієві, часто вони не є ефективними, оскільки потребують ручного керування, змушуючи працівників відволікатися від робочого процесу через новостворений дискомфорт.

З метою автоматизувати процес та підвищити ефективність керуванням мікрокліматом, були створені спеціальні системи керування мікрокліматом, які також відомі як клімат-контроль.

1.1 Загальний огляд об’єкта дослідження

Клімат-контроль — це автоматизована система регулювання температури, вологості, чистоти та руху повітря в приміщенні чи транспортному засобі. Вона підтримує комфортні умови для людей чи тварин або необхідний мікроклімат для техніки, продукції чи інших об’єктів. Її основними функціями є автоматичне регулювання температури — обігрів або охолодження повітря до заданої температури, контроль вологості — зволоження або осушення повітря, фільтрація повітря — очищення повітря від пилу, алергенів, запахів, бактерій тощо, вентиляція — забезпечення припливу свіжого повітря та видалення використаного, циркуляція повітря — рівномірний розподіл повітря по приміщенню.

Клімат контроль складається з кількох компонентів. Першим виступають датчики. Вони збирають дані, такі як температура, вологість, якість повітря, його швидкість та ін. та надсилають на блоки керування. Ці блоки є електронними контролерами, що виконують обчислення, результат якого впливає на сигнали, що подається далі на пристрої керування мікрокліматом. Цими пристроями виступають кондиціонери, вентиляція, обігрів, зволожувачі, фільтри тощо. Зазвичай такі системи мають інтерфейс, за допомогою якого користувач може взаємодіяти та встановлювати бажані параметри. Таким інтерфейсом може виступати як і вбудовані апаратні засоби керування, як клавіши та ручки, або ж програмні засоби, наприклад веб-застосунок.

Оскільки фокусом даного дипломного проєкту є створення інформаційної системи для офісних приміщень, важливо описати середовище, в якому буде застосовуватися такий виріб.

Офісне приміщення — це спеціально облаштований простір у будівлі, призначений для виконання адміністративних, управлінських, аналітичних або інших службових завдань. У таких приміщеннях розміщуються робочі місця співробітників, технічне обладнання, засоби комунікації та, за потреби, зони для переговорів, відпочинку або зберігання документів. Це багатокомпонентне приміщення, кожне з яких потребує власних умов. Зазвичай, воно складається з «оупен-спейсу», де більшість працівників мають робочі місця та проводять більшу частину робочого дня, кухні та санвузлів, які характеризуються загалом низькою активністю впродовж дня за виключенням обідньої перерви, серверна, де власне розташований сервер, переговорні, які використовуються лише за потреби і зазвичай безлюдні та окремі кабінети, в яких розташовані керівники відділів.

«Оупен-спейси» характерні великими відкритими просторами (що випливає з імені) з великою кількістю персоналу та обладнання, які генерують тепло. Ці приміщення є основним фокусом роботи клімат-контролю як найбільші кімнати в приміщенні, і, як результат, мають найбільший вплив на середовище. Важливим є охолодження температури повітря та забезпечення постійної циркуляції повітря, яке не буде створювати дискомфорту при своєму русі.

Кухні та санвузли мають вищу вологість, ніж інші приміщення, мають середній чи малий розмір кімнат, і у випадку з санвузлами, як правило є замкнутими. В цих приміщеннях постійно забруднюється повітря в результаті процесів життєдіяльності працівників, тому важливо забезпечувати постійну вентиляцію кімнат з виведенням надмірної вологи.

Серверні характерні мінімальною людською присутністю, значним тепловиділенням та замкнутістю приміщення через значний шум, що створюють розташовані там самі сервери. Важливо підтримувати низьку температуру та вологість аби вберегти обладнання від деградації та несправностей, утворених перегрівом чи конденсатом.

Переговорні, як правило, є кімнатами середніх розмірів, які рідко використовуються, тому не потребують постійної підтримки комфортних умов. Основними параметром, які необхідно забезпечувати, є температура повітря, на яку впливають люди та обладнання, що перебувають всередині, та циркуляція повітря.

Окремі кабінети керівництва характеризуються низькою кількістю людей, як правило в кількості одної особи, що постійно перебувають в кімнаті разом з працюючим обладнанням. Це генерує певну кількість тепла та потребує циркуляції, однак не інтенсивної.

Існують й інші типи кімнат в офісах, проте вони трапляються рідше і не варті опису[1].

1.2 Аналіз існуючих рішень

На ринку вже представлені численні реалізації таких систем, які керують мікрокліматом в офісних приміщеннях. Для постановки задачі необхідно оглянути їх, виділити їх переваги та недоліки, на основі чого сформулювати концепцію майбутнього виробу. Розглянемо найпопулярніші з них.

HCE80 – контролер для управління окремими зонами клімату. Вміє керувати фанкойлом, підігрівом підлоги, клапанами на батареях та освітленням за допомогою сценаріїв. Має інтерфейс BACnet MS/TP — легко інтегрується в

Building Management System. Підтримуэ датчики температури, вологості, CO_2. Може контролювати присутныстіь людей, чи є відкриті вікна. Дозволяє ручне налаштування[2].

Даний контролер використовується для керування температурою в окремих кімнатах, які під'єднано до центрального блоку, такого як HCC100.

HCC100 — контролер центрального нагріву/охолодження. Цей блок керує центральними системами подачі тепла або холоду, такими, як бойлер, чілер, центральна вентиляція. Він підтримує BACnet, Modbus. Регулює температурні графіки залежно від зовнішньої температури, часу доби тощо. Може керувати насосами, клапанами, котлами, вентиляторами. Доступне підключення для датчиків температури, тиску, потоку. HCC100 відповідає за роботу будівлі є котельні, чілера чи вентиляційної установки на основі запитів від зональних контролерів (наприклад, HCE80)[3].

Honeywell WEBs-AX — програмно-апаратна платформа, що є системою управління всією будівлею BMS, яка інтегрує кондиціонування, освітлення, безпеку, споживання енергії тощо. Побудована на платформі Niagara Framework. Підтримує більшість відомих протоколів, як BACnet, Modbus, LonWorks, KNX. Має веб-інтерфейс для адміністраторів. Можна створювати графіки роботи, отримувати сповіщення та попередження, оформлювати звіти, аналізувати температурний режим. Можливе керування з мобільного пристрою. Підтримує інтеграцію з іншими системами, як пожежна сигналізація, сигналізація, відеоспостереження[4].

Перевами системи Honeywell є наступні особливості:

— відмінна надійність системи;

— широка підтримка протоколів;

— гнучкість. Підходить для малих та великих приміщень;

— готові сценарії з можливістю програмування;

— можливість дистанційного керування.

Недоліками слугують наступні особливості:

— вартість. Вартість одного контролера HCE80 складає 200$ та більше, а HCC100 ще дорожчий;

— WEBs-AX потребує окремий сервер або контролер із Niagara

— встановлення системи потребує сертифікованих фахівців, а налаштування можливе лише з використанням спеціалізованого програмного забезпечення.

Siemens Climatix — це серія контролерів, розроблена спеціально для систем клімат контролю. Дозволяє керувати чилерами, котлами, вентиляційними установками, фанкойлами, тепловими насосами тощо. Існують три основні моделі контролерів Climatix: C600, C400, C200, які варуються за потужністю. Програмуються через Climatix Engineering Tool (CET). Наявна вбудована підтримка BACnet/IP, Modbus, KNX. Можуть працюють автономно або як частина Desigo.

Даний контролер має декілька корисних функцій. Доступний моніторинг стану обладнання. Наявна підтримка веб-інтерфейсу та хмарного доступу (через Climatix IC Cloud). Дозволяє збирати дані, створювати графіки роботи, та сповіщувати користувачів.

Siemens Desigo CC — це флагманська система керування всією будівлею. Вона об’єднує в одну платформу всі підсистеми: кондиціонування, освітлення, сигналізацію, пожежна сигналізація, відеоспостереження, ліфти тощо.

Дана система складається з двох частин:

— Desigo CC Server – сервер BMS з веб-інтерфейсом;

— Desigo PX Controllers – польові контролери для HVAC.

PXC4/PXC5/PXC7 – сучасні контролери з підтримкою IoT TX-I/O. Є модулями вводу/виводу сигналів від датчиків та пристроїв керуванням мікрокліматом Наявна підтримка протоколів BACnet/IP, Modbus, OPC UA, MQTT, KNX, DALI, SNMP.

Дана система має значні можливості. Повний SCADA-функціонал: візуалізація, графіки, історія, тренди Інтеграція з аналізом енергоефективності Можна створити віртуальні зони керування, які динамічно змінюються

Автоматизація на основі сценаріїв: графіки, присутність, погода Web-доступ і мобільні застосунки[5].

Переваги Siemens Desigo / Climatix

— професійне рішення рівня «Smart Building»;

— масштабованість — від одного офісу до цілих кампусів;

— підтримка хмарного моніторингу, IoT, кібербезпеки;

— повна інтеграція зі сторонніми системами;

— потужна графічна SCADA;

— проста інтеграція з ERP, системами обліку енергії.

Недоліки:

— висока вартість. Climatix C600 – від 500$, залежно від конфігурації. Desigo PX контролери – від 600$, а Desigo CC залежить від об'єкта;

— потребує проектування, ліцензій, та інженерного впровадження.

Johnson Controls Metasys — це програмно-апаратна платформа для автоматизації та моніторингу всіх інженерних систем в будівлі. Складається з наступних частин:

— ADS/ADX (Application Data Server) – сервер зберігання даних, керування логікою, візуалізує роботу;

— NAE (Network Automation Engine) – контролери системи на рівні мережи (кондиціонування, енергія);

— SNE/SNC (Supervisory Engines) – контролери з вбудованим веб-інтерфейсом;

— FEC/IOM/FX – польові контролери . Виконують безпосереднє керування обладнанням;

— User Interface (Metasys UI) - веб-інтерфейс із панелями, графіками, сповіщенням, та картами приміщень;

— Sensors & Actuators – датчики, освітлення, клапани та приводи.

Має підтримку протоколів BACnet/IP, Modbus, LonWorks, KNX, SNMP, OPC, MQTT.

Дозволяє в себе інтегрувати керування температурою та повітрям, освітленням, безпекою, відеоспостереженням, пожежною сигналізацією. Дозволяє слідкувати за споживанням електроенергії, води, тепла та створювати звіти за цими показниками. Можливий віддалений контроль через мобільний застосунок Metasys.

Вся система відображається в веб-інтерфейсі: інтерактивні карти будівлі, графіки температур, CO_2, стан обладнання. Можна керувати через ПК, планшет, смартфон. Підтримується система ролей та прав доступу для персоналу. Є графіки, логування, алерти, звіти про відхилення та збої. Можна будувати сценарії: наприклад, по графіку вмикати вентиляцію, знижувати температуру вночі, вимикати світло в порожніх кімнатах[6].

Переваги:

— повна інтеграція з будь-якою інженерною системою;

— сучасний інтерфейс та мобільний доступ;

— економія споживання енергії;

— обширна аналітика, автоматизація, розумні сценарії;

— високий рівень безпеки та доступу;

— гнучкість під різні об'єкти — від школи до хмарочоса.

Недоліки:

— висока вартість системи: контролери NAE/SNE – від ~$1000–2000 за одиницю, польові пристрої (FEC/IOM) – 200-500$, сервер ADS/ADX – залежить від масштабу і ліцензій, через що вартість проєкту може становити від десятків до сотень тисяч доларів;

— потреба в професіональному встановлені та налаштуванні системи.

SmartX — це серія контролерів і пристроїв від Schneider Electric, які застосовуються для керування системами кондиціонування (чилери, вентиляція, фанкойли), побудови автоматизації будівель та інтеграції з іншими системами через відкриті протоколи. Вони працюють у складі EcoStruxure Building Operation (EBO) — основного програмного забезпечення BMS від Schneider.

Основні компоненти SmartX:

— SmartX AS-P/AS-B – автоматизовані сервери — основні контролери для вузлів BMS;

— MP-x Controllers (MP-C, MP-V) – контролери для зонального керування (зони кондиціонування, фанкойли);

— RP-C Controllers – контролери приміщень, керують освітленням, жалюзі, датчиками;

— SmartX Sensors – датчики температури, вологості, CO_2, присутності людей;

— EcoStruxure Building Operation (EBO) – програмна частина. Включає в себе SCADA, тренди, сценарії, аналітика, доступ.

Дана система має ключові можливості, такі як підтримка основних протоколів BACnet/IP, Modbus, KNX, LON, MQTT, веб-доступ до всіх функцій та керування з планшета, смартфона або ПК, повністю програмовані сценарії (логіка, графіки, сповіщення)

Дана система дозволяє інтегруватися з системами кондиціонування (чилери, AHU, котли), освітленням (включення/димування), системами доступу та безпеки, лічильниками[7].

В даного ї системи наявні наступні переваги. Вона підтримує всі основні протоколи, і при цьому легко інтегрується. Керування системою виконується крізь зручний веб-інтерфейс, що дозволяє отримувати доступ з мобільного пристрою, при цьому можливе графічне відображення усіх зібраних даних. Це дозволяє виконувати ефективний моніторинг споживання, формувати звіти та аналітику. Можливе створення графіків роботи, сценарії використання та реакція на події (вимкнення кондиціонеру при відкритті вікна). Можлива хмарна інтеграція.

Недоліками ж виступає висока вартість. AS-P контролер коштує від 1500$, MP-C/RP-C контролери — від 300$, датчики SmartX – 100$. Також необхідно отримувати спеціальну ліцензію, вартість якої варується від розміру системи. А налаштування відбувається лише через сертифікованих інтеграторів. Необхідність в прорієтарних датчиках також збільшує вартість проєкту.

Оглянувши ці системи, ми можемо дійти до наступного висновку. Всі наявні системи є гнучкими, підходять під приміщення та будівлі різних розмірів, від маленьких офісів до цілих хмарочосів, мають підтримку всіх основних пристроїв, що керують мікрокліматом, мають зручне керування за допомогою веб-застосунку та можуть самостійно підлаштовувати параметри. Однак, всі ці системи є дороговартісними, та потребують встановлення сертифікованими спеціалістами. Це обмежує доступність таких систем для підприємств, що не мають великої фінансової спроможності чи розташовані в локаціях, де немає необхідних спеціалістів.

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначення розробки інформаційної системи моніторингу та керування кліматом офісного приміщення з використанням мобільного робота полягає у створенні автономної платформи, здатної ефективно моніторити параметри мікроклімату, приймати рішення, регулювати їх без участі людини та мати можливість мобільності. Це означає, що система зможе самостійно підтримувати оптимальні умови у приміщенні — температуру, вологість, якість повітря тощо — що є ключовою вимогою для сучасних інтелектуальних будівель, та не потребувати фізичного встановлення.

Мобільність системи дозволяє виконувати керування мікрокліматом з будь-якого місця в приміщенні. Це забезпечується компактністю контролера, що дозволяє розміщувати майже в будь-якій точці та переноситись між зонами, що забезпечує гнучкість розгортання системи, та бездротовим підключенням до мережі, завдяки чому немає потреби в фізичному з’єднанні. У перспективі така мобільність дозволяє інтегрувати систему з реальними роботизованими платформами для зонального моніторингу або розширеного обслуговування.

Локалізація зон — ще один важливий компонент системи. Вона забезпечує точне визначення параметрів у конкретних ділянках офісу або будівлі (наприклад,

у переговорній, серверній чи зоні очікування), що необхідно для точного керування мікрокліматом у межах кожної зони. Завдяки цьому система може адаптувати подачу повітря, охолодження чи зволоження, орієнтуючись на актуальні умови.

Для адаптивного регулювання клімату використовується спеціалізоване програмне забезпечення. Система дозволяє оновити свої налаштування, враховуючи нові дані параметри датчиків і зміну умов у приміщенні.

Розробка такої системи передбачає її використання у офісних приміщеннях різних розмірів. Завдяки цьому вона є універсальною, масштабованою і здатною адаптуватися до завдань будь-якої складності, що значно підвищує її цінність для користувачів і власників будівель.

1.3.2 Мета і задачі розробки

На основі проведеного аналізу існуючих підходів до контролю клімату в офісному приміщенні метою даної роботи є підвищення продуктивності підприємства шляхом покращення комфорту його працівників.

Для досягнення поставленої мети необхідно вирішити наступні:

— дослідження і аналіз технологій зонального моніторингу, системи датчиків та сценарного керування;

— розробка програмного забезпечення:

— реалізація логіки адаптивного регулювання параметрів у різних зонах будівлі;

— написання коду для зв’язку між сенсорами, контролером і виконавчими механізмами (наприклад, заслінками, вентиляторами, зволожувачами);

— проведення симуляцій і тестування системи в умовах змінної температури, вологості та інших параметрів повітря.

Висновок до розділу 1

У даному розділі було розглянуто об’єкт дослідження — система клімат-контролю. Це автономна система, призначена для автоматичного регулювання мікрокліматичних параметрів у приміщенні, таких як температура, вологість, вентиляція та якість повітря. Такі системи забезпечують комфортні умови перебування для людей без потреби у постійному ручному налаштуванні. Залежно від сфери застосування, клімат-контроль може використовуватися в офісах, лікарнях, виробничих об’єктах, складах, житлових будівлях тощо.

Офіс є багатофункціональним внутрішнім простором, призначеним для виконання адміністративних, управлінських та координаційних завдань. Він зазвичай включає робочі місця співробітників, зони переговорів, кімнати відпочинку, архіви та технічні приміщення. Кожна з цих зон має власні вимоги до мікроклімату, що зумовлює необхідність гнучкого та зонального управління параметрами повітря.

Особливістю офісного середовища є змінне теплове навантаження, яке залежить від кількості присутніх людей, роботи комп’ютерної техніки, освітлення та зовнішніх кліматичних умов. Також важливим фактором є нерівномірність використання простору протягом дня — одні приміщення можуть бути перевантажені, тоді як інші — порожні. Це створює додаткові вимоги до системи клімат-контролю, яка повинна не лише підтримувати стабільні параметри повітря, а й адаптуватися до змін у режимі реального часу. Були порівняні існуючі реалізації систем клімат контролю. Усі вони характеризуються адаптивністю, можливістю керування багатьма пристроями, можливістю керування через застосунок, чудовою масштабованістю та інтеграцією з іншими системами. Проте такі системи є дороговартісними та потребують сертифікованих спеціалістів для встановлення.

Було сформовано призначення, задачі та цілі розробки. Результуюча система повинна отримувати дані з датчиків, виконувати обрахунки на їх основі, і на основі результатів цих обрахунків надсилати їх на відповідні пристрої. Дана система повинна підтримувати зональність і дозволяти масштабуватися.

2 ВИБІР ЗАСОБІВ ДЛЯ ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Загальний огляд

Інформаційна система клімат контролю офісного приміщення, як і будь-яка інша система, складається з кількох компонентів, що забезпечують її роботу. В її основу полягає програмне забезпечення, яке і виконуватиме всі обрахунки та реалізуватиме всю логіку роботи. Це програмне забезпечення пишеться на певній мові програмування та в певному середовищі розробки. Мова програмування, в свою чергу, залежить від пристрою, на якому результуюча програма і буде виконуватися, оскільки кожен пристрій має обмежену підтримку мов та засобів розробки. Тому вибір середовища програмування буде залежати від мови програмування, яка, в свою чергу, залежить від обраного пристрою.

Блок керування мікрокліматом для функціонування потребує дані, на основі яких і надсилатиме команди пристроям. Ці дані передаються з датчиків. Однак, оскільки фокусом даної роботи є саме розробка інформаційної системи, то можна обмежитися випадково генерованим набором даних, що надсилаються до мікроконтролера.

Конфігурацію системи клімат контролю можна зберігати в базі даних. Ця конфігурація складається з кімнат, датчиків та пристроїв, на які керують мікрокліматом.

Для обміну даних треба обрати спосіб під’єднання та протоколи, що забезпечуватимуть мінімальну затримку, швидкість та надійність зв’язку, дозволять під’єднатися до необхідних пристроїв, бажано у великій кількості, оскільки виконуватиметься керування декількома пристроями одночасно.

В цьому розділі буде розглянуто та порівняно доступні реалізації, та в результаті буде обрано оптимальний варіант для розробки інформаційної системи.

2.2 Вибір контролера

Основою блоку клімат контролю є мікроконтролер. Саме він приймає дані з датчиків, виконує обрахунки та надсилає керуючі сигнали на пристрої. Тому важливо обрати найбільш оптимальну модель, що запропонує чудову продуктивність, великий функціонал, зручний набір інструментів розробки та інші переваги. Для цього розглянемо найпопулярніші та доступні на ринку моделі.

Серія STM32H7 базується на потужному ядрі ARM Cortex-M7, що працює на частоті до 550 МГц. Це найшвидше ядро серед серій STM32, оптимізоване для високошвидкісних обчислень, роботи з числами з плаваючою комою та цифрової обробки сигналів. У деяких моделях наявне друге ядро Cortex-M4, що працює паралельно з основним.

Характеристики в цій серії є наступними. ОЗП становить до 1 МБ вбудованої пам’яті, поділеної на кілька банків для окремого доступу. Флеш-пам’ять становить до 2 МБ. Має підтримку SDRAM/PSRAM/HyperRAM, та наявні інтерфейси для підключення NAND/NOR Flash, QSPI. Присутня кеш-пам’ять L1 рівня.

Процесор має наступні характеристики. Частота ядра піднімається до 550 МГц. Присутня апаратна підтримка одинарної та подвійної точності, підтримка SIMD-інструкцій для швидкої обробки масивів. Має декілька каналів з підтримкою подвійної буферизації. Підтримує паралельну роботу периферії та ядра

STM32H7 підтримує велику кількість інтерфейсів, які дозволяють під’єднувати різноманітні пристрої, такі як:

— USB OTG — підтримка USB-хост та пристроїв;

— Ethernet MAC з підтримкою з точним синхронізованням часування;

— швидкий CAN для автомобільної промисловості;

— USART/UART/SPI/I2C/I2S/SDMMC/QSP — для підключення датчиків, дисплеїв, аудіо;

— LCD-TFT контролер — можливість створення інтерфейсів із графікою;

— ADC (16-біт), DAC, OPAMP, компаратори — для обробки аналогових сигналів;

— Timers/PWM/Encoder Interface — для керування двигунами, ШІМ, часових задач.

Серія STM32H7 не орієнтована на енергозбереження, проте він має енергетично ефективний режим STOP, а також динамічне керування тактуванням і живленням, що дозволяє оптимізувати споживання в енергочутливих системах.

Даний контролер має власний набір інструментів для розробки програмного забезпечення. Він складається з середовища розробки STM32CubeIDEЮ, HAL/LL бібліотеки – високо та низькорівневі API для зручної розробки, STM32CubeMX – утиліти для генерації конфігураційного коду. Також наявна підтримка CMSIS, FreeRTOS, TouchGFX для графіки, USB Device/Host Stack

Серія ESP32-S3 має два ядра Xtensa LX7, кожне з яких працює на частоті до 240 МГц. Це ядра власної архітектури від Espressif, оптимізовані для енергоефективної роботи й одночасної обробки багатьох задач. Мікроконтролер підтримує інструкції SIMD (Single Instruction Multiple Data) для паралельної обробки масивів.

Внутрішня оперативна пам’ять в даної серії становить до 512 КБ. Наявна підтримка зовнішньої PSRAM до 8 МБ та флеш-пам’яті до 16 МБ (зовнішня через QSPI).

В даної серії наявна вбудована підтримка бездротових підключень Wi-Fi 4 (802.11 b/g/n)**,** з підтримкою SoftAP і клієнтського режиму, та Bluetooth 5.0 Low Energy (LE)[8].

ESP32-S3 має апаратне прискорення для штучного**,** яке підтримує виконання нейронних мереж. Існує фреймворк від Espressif ESP-DL для запуску AI-моделей на пристрої. Наявна підтримка TensorFlow Lite for Microcontrollers, що дозволяє розгортати ML-моделі прямо на пристрої, без хмари, в режимі офлайн.

ESP32-S3 має велику кількість периферійних інтерфейсів:

— USB OTG — підтримка USB-хост та пристроїв;

— SPI, I2C, UART, I2S — для підключення датчиків, дисплеїв, аудіо;

— LCD interface — для підключення дисплеїв;

— JTAG, SDIO, PWM, GPIO — для відладки, карти пам’яті, сервомоторів;

— Камери (DVP interface**)** — сумісний з камерами типу OV2640.

ESP32-S3 має апаратне шифрування AES, SHA, RSA, HMAC, та підтримує Secure Boot, Flash Encryption й OTA (оновлення прошивки по повітрю) з підписом.

Дана серія підтримує значну кількість засобів розробки програмного забезпечення

— ESP-IDF (офіційна платформа SDK);

— Arduino Core for ESP32 **—** для швидкого прототипування;

— PlatformIO;

— Visual Studio Code.

Також були розроблені спеціальні фрейморки ESP-DL / ESP-MDF / ESP-ADF — для ML, аудіо, мереж відповідно[9].

Існують варіації плати з вже вбудованим функціоналом, підлаштованим під конкретні задачі:

— ESP32-S3-DevKitC – набір розробника;

— ESP32-S3-EYE — з камерою для комп'ютерного зору;

— ESP32-S3-Box — з дисплеєм, мікрофоном, динаміком.

i.MX RT1060 розроблений для задач, де потрібна максимальна продуктивність мікроконтролера, без складності операційних систем мікропроцесорного рівня. Даний контролер має ядро ARM Cortex-M7 на частоті до 600 МГц, що є одним з найшвидших таких ядер на ринку. Цей процесор має одинарну точність в числах з плаваючою комою. Підтримує DSP-інструкції SIMD та DMA із кількома каналами та підтримкою «scatter-gather»

На відміну від класичних мікроконтролерів, **i.**MX RT1060 не має вбудованої флеш пам'яті**,** а використовує зовнішню через QSPI або FlexSPI — це забезпечує гнучкість у виборі обсягу пам'яті й високу швидкість завантаження. Внутрішня оперативна пам'ять складає 512 КБ. Наявна високошвидкісна шина FlexRAM для доступу до неї. Наявна підтримка HyperRAM/PSRAM, External QSPI Flash, SDRAM/NOR/NAND та кеш-пам'яті для прискорення роботи з зовнішньою пам'яттю. До цього ж наявна можливість XIP (execute in place) – виконання коду безпосередньо в флеш пам'яті. Це дозволяє працювати з великими об'ємами даних

у режимі реального часу, включаючи зображення, відео, аудіо, великі масиви сенсорних даних.

i.MX RT1060 має значну підтримку периферії та пристроїв, таких як:

— LCD контролер (до 1024x768) — для графічних дисплеїв;

— Camera Interface (CSI) — підтримка підключення камери;

— Ethernet 10/100 Mbps;

— USB OTG (x2, High Speed);

— SD/MMC, SPI, I2C, UART, CAN, PWM, ADC, DAC;

— SAI (аудіо інтерфейс) — підключення мікрофонів, колонок, DSP-чіпів;

— QSPI/FlexSPI — для підключення високошвидкісної Flash або PSRAM.

Даний мікроконтролер підтримує шифрування даних AES-128, AES-256, SHA, RSA. Наявна можливість безпечного старту (Secure Boot), та також True Random Number Generator (TRNG**).** Це дозволяє використовувати i.MX RT в комерційних, медичних, промислових і навіть захищених IoT-застосуваннях[10].

Даний контролер має наступні засоби розробки:

— MCUXpresso IDE — офіційна IDE від NXP (на базі Eclipse);

— MCUXpresso SDK — готові драйвери, бібліотеки, приклади;

— підтримка FreeRTOS**,** LVGL (графічний фреймворк)**,** EdgeLock для безпеки;

— Segger J-Link**,** Keil MDK**,** IAR Embedded Workbench — сторонні інструменти.

Популярні плати на базі RT1060

— i.MX RT1060 EVK (Evaluation Kit) — повнофункціональна плата з дисплеєм, камерою, аудіо, USB, SD;

— Teensy 4.1 — популярна DIY-платформа на базі i.MX RT1062 (аналог RT1060), використовується для швидкого прототипування.

Raspberry Pi Pico W RP2040 — це двоядерний 32-бітний мікроконтролер на базі ядер ARM Cortex-M0+**,** розроблений Raspberry Pi для навчання, прототипування та побутових/IoT-застосувань. У версії Pico W до цієї платформи

додано Wi-Fi через чип Infineon CYW43439. В цього контролера процесор має 2 ядра ARM Cortex-M0+ з тактовою частотою до 133 МГц з 3-стадійнм Pipeline, що дозволяє економічно виконувати інструкції. Немає апаратної підтримки чисел з плаваючою комою, однак її можна реалізувати програмно. Наявна оперативна пам'ять 264 КБ, що розділена між ядрами та периферією. Немає вбудованої флеш пам'яті і використовує зовнішню 2 МБ Flash на платі Pico W. Підтримка XIP (execute-in-place) з флеш пам'яті.

Контролер Pico W оснащено чипом бездротового зв'язку Infineon CYW43439, що підтримує Wi-Fi 802.11n 2.4 ГГц. Присутній SoftAP, клієнтський режим. Апаратна реалізація Bluetooth на чипі є, але наразі офіційно неактивований (на рівні прошивки).

Даний контролер має певні унікальні можливості. Завдяки Pio (Programmable I/O – апаратно-програмованим блокам), дозволяється реалізовувати свої власні протоколи (наприклад, VGA, DVI, протоколи датчиків). Наявні DMA-контролери. Також присутні контролери Timers, PWM, ADC (12-біт) для аналогових входів. Має до 26 GPIO, з можливістю програмного призначення. Присутній UART, SPI, I2C — по кілька ліній кожного. Достуаний USB 1.1 Device (Full-Speed), завдяки чому здійснюється підключення до ПК як HID, CDC тощо.

Одним з пріоритетів даного контролера є енергоспоживання. Він працює на низькій напрузі 1.8–3.3 В. Можливо переведення в режими sleep, dormant та deep sleep. Це дозволяє живитися від батареї при правильному налаштуванн[11].

Даний контролер має наступні засоби для розробки:

— C/C++ SDK — офіційний SDK для низькорівневого доступу;

— MicroPython — модифікація мови Python, зручна для початківців;

— CircuitPython — від Adafruit, розширення MicroPython;

— Arduino Core для RP2040 — підтримка у звичному середовищі Arduino IDE;

— Підртримка з Thonny, Visual Studio Code, PlatformIO, Pico SDK.

Оглянувши ці моделі, було вирішено обрати ESP32-S3. Цей контролер має двоядерний процесор, потужніший ніж в RP2040. Хоча в інших наведених

пристроях частота ядра більша, в них ядро одне, що дещо позначується на мультизадачності. Також в нього присутня вбудована підтримка досить сучасна версія Bluetooth 5.0 та наявний Wi-Fi 4. Завдяки цьому до контролера можна під'єднати пристрої керування кліматом та датчики бездротово. Це спрощує розміщення блоку керування, не вимагаючи додаткових рішень.

2.3 Вибір мови програмування та середовища розробки.

Обраний контролер ESP32-S3 має підтримку багатьох мов програмування. Логічним постає питання її вибору. Найбільш популярними варіантами наразі є C++, а саме офіційний фреймворк ESP-IDF чи Arduino Core for ESP32. MicroPython, Lua, Rust. Розглянемо кожну з них.

C++ є основною мовою програмування даного контролера. Існує два найпоширеніших фреймворки Arduino Core та ESP-IDF. Arduino має значну кількість вже готових бібліотек, що дозволяє підібрати функціонал під власну задачу, простий для опанування та роботи синтаксис, та велика спільнота, утворена довгими роками навколо екосистеми Arduino. ESP-IDF в свою чергу пропонує глибше налаштування, повний апаратний контроль та офіційну підтримку розробників контролера. Використання цього фреймоворку є дещо складнішим, проте велика свобода дій дозволяє чудово оптимізовувати програмне забезпечення під даний пристрій[12].

MicroPython — це легка версія Python 3, спеціально створена для мікроконтролерів, таких як ESP32. Вона дозволяє писати скрипти, які виконуються безпосередньо на мікроконтролері, без компіляції. Завдяки успадкованим від Python особливостям, його синтаксис є простим, має швидший цикл розробки програмного забезпечення, оскільки відсутня компіляція. До того ж, можлива робота через Wi-FI[13].

Lua — це легка, низькорівнева скриптова мова, яка дуже добре підходить для мікроконтролерів завдяки невеликому затребуваному обсягу пам'яті, простому синтаксису, та його динамічності. Lua зазвичай реалізується з використанням

NodeMCU, який був дуже популярний для ESP8266. Для ESP32 також існують проєкти, які дозволяють писати на Lua, але вони менш активні, ніж MicroPython або Arduino. Через це рівень підтримки є меншим, а сам статус імплементації носить характер експериментальності, а кількість існуючого матеріалу порівняно менша з C++ та MicroPython[14].

Rust — це сучасна системна мова програмування з фокусом на високу продуктивність, порівняну з C/C++, більшу безпеку пам'яті (без null, без segfault), одночасність (concurrency) без небезпеки «перегонів». Ця мова дозволяє аналогічно C++ низькорівнево керувати апаратною частиною. Вона швидко зростає в популярності, тому вже має значну кількість інструментів, матеріалів та проєктів, пов'язаних з нею. Для контролерів ES{ була створена esp-rs, спільнота, що спеціалізується на розробці та адаптації мови під нього. Наразі під контролер ESP32-S3 немає повноцінної підтримки, але вона очікується згодом[15].

З огляду на вищезазначені мови та їх особливості, було обрано C++, а саме фреймворк ESP-IDF. Офіційна підтримка, глибокий контроль, велика кількість навчального матеріалу робить цю мову чудовим вибором для розробки програмного забезпечення.

Щодо вибору середовища розробки, то ESP-IDF має офіційний модуль для Visual Studio Code. В ньому вбудована простіша інтеграція з дебагером, автодоповненням і системою збірки CMake, а також наявні шаблони для швидшого написання програми. Це та повний доступ до всіх частин мікроконтролера робить вибір однозначним.

Для написання програми, яка буде симулювати дані з датчиків, то оптимальним рішенням є Python. Ця мова має мінімалістичний синтаксис, легке налаштування та велику кількість бібліотек, що дозволяє підібрати інструмент під будь-яку задачу.

2.4 Вибір баз даних

Для зручності налаштування системи та її адаптивності, чудовим рішенням для збереження конфігурації системи є база даних.

Обраний мною контролер не має значного об’єму пам’яті, а керування мікрокліматом всього офісного приміщення може бути об’ємною задачею за великої кількості приміщень та пристроїв, тому база даних буде розміщена на сервері. Цим сервером слугуватиме мій персональний комп’ютер, до якого і буде під’єднаний контролер обраним чином.

Із задачею зберігання та передавання інформації про систему клімат контролю чудово впорається реляційна база даних. Ця модель заснована на зв’язках між таблицями. Кожна з таблиць складається з колонок, які називають атрибутом, та рядків, що називають записами. В кожного запису є атрибут-ідентифікатор, що називається ключем. Використовуючи номер унікального ключа запису одної таблиці в іншій, та поєднавши їх, ми тим самим отримуємо зв’язок та можемо отримати відповідний запис.

Завдяки особливостям реляційної моделі, ми можемо швидко збудувати базу даних, наповнити її необхідними даними та розширювати її за потреби.

Існує безліч реалізацій цієї моделі. Найпопулярнішими серед них є MS SQL, MySQL, PostgreSQL та SQLite. Задля обрання оптимального варіанту, розглянемо кожну з них та порівняємо їх.

Microsoft SQL Server — це потужна та функціональна реляційна система управління базами даних (СУБД), створена компанією Microsoft. Вона забезпечує надійне зберігання, доступ і керування великими обсягами даних, і є однією з найпопулярніших СУБД у світі. SQL Server використовується як у локальних середовищах, так і в хмарних інфраструктурах (наприклад, Azure SQL Database).

Основою SQL Server є Database Engine. Це основний рушій бази даних, який обробляє запити, зберігає дані, забезпечує цілісність інформації, транзакції та безпеку. Цей компонент виконує T-SQL-команди, здійснює індексацію, сортування та управління транзакціями.

T-SQL (Transact-SQL) — це розширення мови SQL, розроблене Microsoft. Це розширення має власний додатковий функціонал, такий як змінні, цикли, умовні оператори, обробка помилок, збережені процедури, тригери та інше. Це дозволяє виконувати певні обчислювальні дії прямо в базі даних, при цьому підтримує всі стандартні запити.

MS SQL Server складається з наступних елементів:

— SSMS (SQL Server Management Studio) — графічна оболонка для керування базами, написання запитів, створення схем;

— SQL Server Agent — служба для старту автоматизованих завдань (наприклад, резервного копіювання);

— SSIS (SQL Server Integration Services) — інструмент для інтеграції та обробки даних (ETL-процеси);

— SSRS (SQL Server Reporting Services) — система для створення інтерактивних звітів;

— SSAS (SQL Server Analysis Services) — засіб для аналітики та побудови OLAP-кубів.

MS SQL Server має наступні засоби, корисні при захисті даних:

— аутентифікація користувачів (Windows або SQL);

— ролі та дозволи: можна обмежити доступ до окремих таблиць або операцій;

— шифрування: Transparent Data Encryption (TDE) захищає файли БД, а функція Always Encrypted — самі значення в таблицях;

— аудит — запис усіх змін і доступу до даних.

У MS SQL Server є наступні бази даних:

— master — головна база, зберігає конфігурацію сервера;

— msdb — база для SQL Agent (завдання, плани);

— model — шаблон для створення нових БД;

— tempdb — тимчасове сховище для проміжних результатів;

— користувацькі бази — створені користувачами для конкретних застосунків[16].

MS SQL Server має наступні переваги:

— інтеграція з Microsoft Azure, Power BI, Visual Studio, що значно спрощує розробку застосунку та його подальшу підтримку;

— висока продуктивність і стабільність, що критично в системах інтернет-речей;

— розвинені засоби моніторингу, оптимізації запитів, що корисно для відслідковування аномалій в показниках датчитків;

— візуальний інструментарій для адміністрування (SSMS), що дозволяє не досвідченим користувачам вносити необхідні зміни;

— підтримка великих обсягів даних, складних структур і запитів.

MS SQL Server має наступні недоліки:

— ліцензування — платна система (особливо у великих розгортаннях). Це суттєво підвищує вартість підтримки системи, вимушуючи платити щомісячно за продовження користування

— часткова залежність від ОС Windows (Linux підтримується з версії 2017, але з обмеженнями). Це частково унеможливлює розгортання на слабших комп’ютерах, оскільки Linux є значно менш вимогливим до можливостей системи та піддається суттєвим змінам, дозволяючи відключити непотрібні частини системи, що не використовуються чи впливають на продуктивність;

— складність налаштування в порівнянні з простішими СУБД.

MySQL — це одна з найпопулярніших у світі систем управління реляційними базами даних (СУБД) з відкритим кодом. Належить корпорації Oracle. MySQL є основою безлічі веб-застосунків, зокрема таких як WordPress, Facebook, YouTube, X(Twitter), та багатьох SaaS-платформ. Ця СУБД розрахована на велику кількість користувачів та великі обсяги даних. Підтримує стандартну мову SQL.

Завдяки відкритому коду, ця СУБД працює на більшості сучасних платформах, таких як Windows, Linux, macOS. Це дозволяє розгорнути та розроблювати MySQL на переважаючій більшості пристроїв та систем, не

вимагаючи придбання нового обладнання, при цьому є легкою в розгортанні та адмініструванні.

Дана СУБД чудово інтегрується з безліччю мов програмування та фреймворків, таких як PHP, Python, Java, .NET, що тільки підкреслює гнучкість та кросплатформеність.

MySQL складається з наступних частин:

— MySQL Server — основний сервер для роботи з БД;

— MySQL Client — інструмент командного рядка для взаємодії з сервером;

— MySQL Workbench — офіційна графічна оболонка для створення, адміністрування та проєктування баз даних;

— Storage Engines — рушії збереження даних. Найвідоміші: InnoDB — підтримує транзакції, зовнішні ключі, ACID; та MyISAM — старіший рушій, швидкий для читання, але без транзакцій.

MySQL забезпечує базові механізми безпеки:

— аутентифікація користувачів;

— керування привілеями через GRANT/REVOKE;

— SSL-шифрування з'єднань;

— ролі користувачів та ізоляція доступу.

MySQL підтримує декілька засобів реплікації та масштабування:

— майстер-слейв реплікацію (master-slave);

— мульти-мастер реплікацію (з MySQL Group Replication);

— шардінг (розділення даних між кількома серверами вручну або через сторонні системи);

— хмарні рішення — Amazon RDS for MySQL, Google Cloud SQL, Azure Database for MySQL[17].

Переваги в MySQL є такими:

— безкоштовна ліцензія для особистого використання і відкритий код;

— простота використання, налаштування та подальшої підтримки;

— швидкодія системи при віддачі даних;

— велика спільнота розробників, багато інструментів і бібліотек;

— гнучка архітектура з багатьма типами збереження даних.

Основними недоліками MySQL виступають наступні особливості:

— обмежені можливості складної аналітики (у порівнянні з PostgreSQL чи MS SQL Server);

— дещо простіша реалізація SQL-стандартів;

— реплікація може бути складною в налаштуванні в користувачів без досвіду;

— комерційний функціонал доступний лише у платній версії (MySQL Enterprise Edition).

SQLite — це дуже легка реляційна база даних для, що не потребує окремого серверного програмного забезпечення. Це одна з найпоширеніших баз даних у світі, яку використовують в мобільних застосунках, браузерах, вбудованих пристроях, десктоп-програмах та деяких веб-сайтах. Це повністю відкрита бібліотека C, яка реалізує повноцінну реляційну СУБД. Вона вбудовується безпосередньо в застосунок і зберігає всі дані в одному файлі на диску розміром близько 500 КБ. В її основу взято принципи простоти, надійності і мінімальності вимоги до ресурсів. Не дивлячись на свою мінімалістичність, повністю підтримує мову SQL.

SQLite складається з наступних частин:

— SQLite CLI — утиліта командного рядка для роботи з базою (sqlite3);

— файлова база даних — наприклад, app_data.db;

— SQLite API — доступна у багатьох мовах: C/C++, Python (sqlite3), JavaScript (через WebAssembly), Java (Android), Swift (iOS), .NET, Go, Rust тощо.

Через свою простоту SQLite сам по собі не має складної моделі безпеки, тому виникає необхідність у власних рішеннях безпеки, таких як:

— рівень доступу до файлу;

— підтримує шифрування через сторонні рішення, як: SQLCipher — версія SQLite із повним AES-шифруванням; SEE (SQLite Encryption Extension);

— офіційна, але платна версія[18].

В SQLite має наступні переваги:

— надзвичайно проста у використанні — жодної інсталяції, все в одному файлі;

— універсальність – можливість імплементації в будь-який код на будь-якому пристрої;

— портативність — можна перенести базу просто як файл;

— швидкість для невеликих/локальних застосунків;

— безкоштовна та без ліцензійних обмежень**;**

— чудово підходить для застосунків, не потребуючих інтернет підключення.

Недоліками SQLite виступають наступні особливості:

— не підходить для високонавантажених багатокористувацьких вебсервісів;

— немає розділення ролей або складної системи безпеки;

— однопотоковість при записі: лише один запис одночасно (але отримувати дані можуть багато процесів);

— менше оптимізацій та додаткового функціоналу, ніж у MySQL чи PostgreSQL (наприклад, немає збережених процедур, розділення даних на схеми).

PostgreSQL — це потужна об’єктно-реляційна система керування базами даних (ORDBMS) з відкритим кодом. Вона відома своєю надійністю, масштабованістю, розширюваністю та строгим дотриманням SQL-стандартів. PostgreSQL використовується такими компаніями, як Apple, Instagram, Skype, Reddit, Spotify, TripAdvisor. Це системна СУБД, яка поєднує реляційний підхід до зберігання даних із підтримкою об’єктних типів, JSON, геоданих та модифікацій, що додають власні типи, оператори, функції та індекси. Вона працює як серверна база даних, яка обробляє численні запити одночасно, підтримуючи транзакції, індекси, тригери, збережені процедури тощо. Дана СУБД має відкритий код та чи не найбільшу спільноту розробників, утворену навколо неї. Працює на більшості сучасних ОС: Windows, Linux, macOS.

PostgreSQL складається з таких компонентів:

— PostgreSQL Server (postgres) — головний компонент бази даних;

— psql — інтерфейс командного рядка;

— pgAdmin — офіційна GUI-оболонка;

— Extensions — модулі для розширення функціональності (PostGIS, pg_stat_statements, pgcrypto тощо);

— libpq —бібліотека API (наприклад, для Python, Node.js, Java).

В даної СУБД безпека забезпечується наступним чином:

— ролі та права доступу (гнучка модель доступу через GRANT/REVOKE);

— аутентифікація через паролі, SSL, GSSAPI, LDAP, Kerberos;

— шифрування з'єднань (SSL);

— Row-level security — обмеження на рівні рядків таблиць;

— аудит (через pgAudit або інші розширення)[19].

Перевагами PostgreSQL виступають:

— надійність, стабільність і висока продуктивність;

— відкрите, безкоштовне ПЗ (PostgreSQL Global Development Group);

— висока відповідність SQL-стандартам;

— гнучкість, розширюваність, масштабованість;

— сучасні типи даних: JSONB, XML, масиви;

— підходить для OLTP, OLAP, GIS, time-series і навіть ML.

Недоліками PostgreSQL виступають:

— складніший для початківців, ніж SQLite чи MySQL;

— може мати вищу складність конфігурації та адміністрування;

— у деяких випадках (веб-хостинг) менш підтримується за MySQL;

— відсутність деяких бізнес-функцій, які є в Oracle або MS SQL (але є аналоги).

Проаналізувавши наведені СУБД, було обрано MySQL. Не дивлячись на легкість та простоту імплементації, SQLite не є дуже гнучкою та потребує автоматизації при розширені та додаткової імплементації безпеки. При цьому не є настільки ж зручною в адмініструванні, потребуючи розробки власного рішення.

PostgreSQL та MS SQL більш громіздкими системами з орієнтацією на веб-застосунки.

2.5 Вибір протоколу з’єднання та обміну інформації

Для обміну даних з контролером необхідно визначитися зі способом підключення. ESP32-S3 дозволяє підключатися як дротовим, так і бездротовим способом. Бездротовий спосіб дозволяє розміщувати контролер в будь-якій точці приміщення за умови доступності сигналу, не потребуючи прокладки дротів до інших пристроїв.

ESP32-S3 підтримує два бездротових протоколи: Wi-Fi та Bluetooth. Розглянемо їх детальніше.

Wi-Fi — це основний спосіб бездротового підключення ESP32-S3 до локальної мережі або інтернету. Чип ESP32-S3 має вбудований Wi-Fi-модуль, що підтримує стандарт IEEE 802.11 b/g/n у діапазоні 2.4 ГГц. Може працювати в режимах станції, точки доступу та обох одночасно. Максимально доступна швидкість передачі даних становить 150 Мбіт/с з максимальною потенційною відстанню зв’язку до 100 м у відкритому просторі. Доступні протоколи безпеки WPA, WPA2 та частково WPA3, також доступне шифрування AES. Можливе підключення до інтернету, що дозволяє отримувати додатково дані для обрахунків. Енергоспоживання в цього протоколу є високим, проте спосіб використання не передбачає живлення від батареї чи обмеження в електроенергії[20].

Bluetooth Low Energy (BLE) — це енергоефективна версія Bluetooth, орієнтована на обмін невеликими порціями даних на коротких відстанях. BLE чудово підходить для інтернету речей, сенсорів, носимих пристроїв тощо. ESP32-S3 підтримує лише BLE 4.2/5.0, а не класичну версію. Це обмежує дистанцію до 10 метрів та швидкість передачі фактично становить до 40 Кбіт/с. Безпека забезпечується лише шифруванням AES-128, що достатньо для використання в інтернеті речей. Енергоспоживання є наднизьким, однак, як вже було зазначено вище, не є критерієм для розробки системи[21].

Протокол BLE чудово підходить для обміну даних з датчиками, проте в даному проєкті блок керування буде отримувати дані з сервера, який отримуватиме дані з кількох датчиків та надсилатиме сигнали на декілька пристроїв керування мікрокліматом. Тому важливим аспектом є саме швидкість передачі і обираємо Wi-Fi.

Для обміну даними з використанням Wi-Fi також потрібен програмний протокол. З ESP32-S3 зазвичай використовують ESP-NOW, MQTT, HTTP/REST API, WebSocket. Розглянемо кожен з них та оберемо оптимальний.

ESP-NOW – протокол, розроблений виробником контролера Espressif. Він базується на кадрах управління Wi-Fi (Wi-Fi Action Frames), що дозволяє пристроям спілкуватися напряму без встановлення TCP-з'єднання. Передача відправляється за MAC-адресою іншого пристрою. Цей протокол дозволяє створити «mesh» мережу чи мережу зірку з кількох пристроїв ESP. Цей протокол може дублюватися з Wi-Fi для більшої багатозадачності.

Цей протокол має кілька режимів роботи:

— Controller + Slaves (зірка) — один ESP32-S3 надсилає/отримує дані до/від кількох пристроїв;

— Peer-to-Peer — обидва пристрої і відправляють, і приймають;

— Комбінований режим з Wi-Fi — один інтерфейс обслуговує ESP-NOW і стандартне Wi-Fi-з'єднання (але не одночасно як AP і STA)[22].

Цей протокол має декілька переваг. Він не потребує встановлення з'єднання, має низьке енергоспоживання, не потребує роутера та підтримує шифрування AES-128.

Однак цей протокол має і наступні недоліки. Розмір пакету обмежений і становить 250 байт. Немає можливості режиму роботи як точки доступа і станції одночасно. Без знання MAC-адреси обмін даними неможливий, а обмін даними з іншими пристроями, окрім як ESP неможливий в принципі.

MQTT (Message Queuing Telemetry Transport) — це легкий мережевий протокол публікації/підписки, створений для надійної передачі повідомлень між пристроями в умовах обмежених ресурсів: низька швидкість, висока затримка,

обмежене енергоживлення. MQTT працює за моделлю "Publish/Subscribe". Вінн складається з видавця, підписника та брокера. Видавець (Publisher) — відправляє повідомлення на певну тему (topic). Підписник (Subscriber) — отримує всі повідомлення з тем, на які він підписаний. Брокер (Broker) — сервер-посередник, який пересилає повідомлення від видавців до підписників.

Повідомлення MQTT має наступну структуру:

— Topic — текстовий шлях;

— Payload — вміст повідомлення;

— QoS (Quality of Service): 0 – отримання без підтвердження, 1 – отримання підтвердження отримання мінімум один раз, 2 – отримання підтвердження лише один раз;

— Retained message — зберігається на брокері і надсилається новим підписникам[23].

MQTT володіє наступними перевагами. Цей протокол є енергоефективним, мінімально завантажує трафік мережі, підтримує двосторонній зв’язок, дозволяє інтеграцію з хмарою та має просту структуру.

Недоліками ж виступають необхідність брокерів, відсутність шифрування за замовчуванням, потребує Wi-Fi і стабільності зв’язку.

HTTP/REST — це стандартний інтерфейс обміну даними між пристроями в інтернеті або локальній мережі, який базується на протоколі HTTP та концепції REST (Representational State Transfer). ESP32-S3 може виступати як HTTP-клієнт — надсилає запити (GET, POST, PUT, DELETE) на сервер, та HTTP-сервер — приймає запити від браузера, мобільного додатку чи іншого пристрою[24].

Контролер підтримує версію протоколу 1.1. Його архітектура заснована на RESTful і є ресурсно орієнтованою. Підтримує JSON та XML. Безпека забезпечується протоколом HTTPS. Також є підтримка браузерів.

В цього протокола наступні переваги. Його сумісність з браузерами дозволяє реалізувати керування крізь веб-застосунки. Він легко тестується. Його поширеність та використання на протязі десятиліть створило велику базу знань,

заснованих на великої кількості різноманітних проєктів, що спрощує розробку. Це в свою чергу дозволяє інтегрувати з іншими системами.

Недоліками ж є високе енергоспоживання, помітно вищі затримки в порівнянні з іншими протоколами, відсутній механізм опитування в реальному часі та потребує Wi-Fi.

WebSocket — це мережевий протокол, який забезпечує двосторонній зв’язок у реальному часі між клієнтом (наприклад, браузером або ESP32-S3) та сервером через одне постійне TCP-з'єднання. На відміну від HTTP, WebSocket дозволяє серверу самостійно надсилати повідомлення клієнту без запиту — це називається push-комунікація. Протокол має двосторонню передачу. Затримки дуже невеликі і вимірюються в мілісекундах. Як і HTTP підтримує JSON і підтримується браузерами. Безпека забезпечується протоколом wss:// (SSL/TLS)[25].

В даного протокола є наступні переваги. Він підтримує двосторонній зв’язок, що дозволяє обом сторонам ініціювати передачу. Затримка сигналу в нього мінімальна. Працює з браузерами, що дозволяє реалізувати веб-застосунок. При цьому не навантажує трафік мережі.

Недоліками ж виступають необхідність в Wi-Fi, складність імплементації захисту на ESP та необхідність підтримки з обох сторін.

З огляду на перелічені протоколи, найоптимальнішим вибором є MQTT. Він простий в реалізації, легкий в роботі, економний в мережевих та енергоресурсах та чудово підходить під централізований збір даних завдяки брокерам.

Висновок до розділу 2

Апаратною основою блоку керування кліматом в офісному приміщенні було обрано ESP32-S3. Цей контролер має досить потужний двоядерний процесор, що забезпечує швидкість роботи та мультизадачність. Наявність вбудованих бездротових підключень Wi-Fi та Bluetooth Low Energy дозволяє реалізовувати велику кількість підключень без необхідності прокладати дроти.

Мова програмування була обрана ESP-IDF, яка є офіційним фреймворком мови C++. Незважаючи на порівняно більшу складність в порівнянні з Arduino Core, фреймворк надає повний контроль над апаратними можливостями контролера, що дозволяє оптимізувати програму під конкретну задачу та раціональніше використовувати доступні обчислювальні ресурси.

Для написання програм, що симулюють дані з датчиків для мікроконтролера, було обрано мову Python. Ця мова чудово підходить для написання невеликих програм завдяки мінімалістичному синтаксису, великому набору доступних бібліотек, що легко встановлюються та відсутності компіляції.

Середовищем розробки було обрано Visual Studio Code з офіційним модулем ESP-IDF. Це спрощує розробку, оскільки цей модуль містить всі необхідні компоненти для розробки, та вже готові шаблони.

Джерелом інформації було обрано базу даних MySQL. Ця база даних легко розгортається й адмініструється, чудово працює з великим обсягом даних та є кросплатформеною.

Способом обміну інформацією виступатиме програмний протокол MQTT, що забезпечується протоколом Wi-Fi. Його простота імплементації, легка структура та централізованість через брокерів чудово підходять для даного проєкту.

3 РОЗРОБЛЕННЯ СИСТЕМИ

3.1 Архітектура системи

У дипломному проєкті реалізовано розподілену систему автоматизованого клімат-контролю для офісного приміщення, яка побудована за модульним принципом із використанням сучасних технологій обміну даними. Основу архітектури складає взаємодія між кількома ключовими компонентами: контролером, MQTT-брокером, програмою-симулятором сенсорів, бази даних. Компоненти з'єднані між собою через мережу за допомогою протоколу MQTT — легковагового мережевого протоколу, спеціально створеного для швидкого, надійного і масштабованого обміну даними в системах автоматизації.

Контролер є центральною обчислювальною одиницею, яка виконує функції логічного керування системою мікроклімату. У межах цього проєкту контролером є програмно-апаратний модуль, що реалізований на базі мікроконтролера ESP32-S3, але завдяки фреймворку ESP-IDF, за потреби можна реалізувати на будь-якому іншому контролері ESP32, в якому наявний Wi-Fi, при чому зміни в код програми треба вносити мінімальні, чи взагалі не потрібно. На ньому працює спеціально розроблене програмне забезпечення, яке виконує такі основні функції:

— отримання даних про стан середовища: контролер підписується на MQTT-топіки, у яких надходять значення температури, вологості, концентрації CO_2 та інших параметрів мікроклімату;

— аналіз отриманих даних: у реальному часі виконується обробка даних згідно з заданими алгоритмами регулювання. Наприклад, якщо температура перевищує певний поріг, система автоматично активує охолодження;

— формування керуючих сигналів: залежно від отриманих даних, генеруються команди на виконавчі механізми (нагрівачі, вентилятори, зволожувачі тощо), які також передаються через MQTT.

MQTT-брокер є комунікаційним центром усієї системи. Саме він відповідає за доставку повідомлень між усіма учасниками обміну. Брокер функціонує як посередник: він приймає повідомлення, опубліковані одним клієнтом (наприклад,

симулятором датчика), і передає їх усім підписаним на відповідний топік клієнтам (в даному випадку контролеру).

У даному проєкті брокер реалізовано з використанням відкритого програмного забезпечення Eclipse Mosquitto, яке розгорнуто на окремому комп'ютері в локальній мережі. Цей брокер працює постійно у фоновому режимі, забезпечуючи стабільну маршрутизацію повідомлень.

MQTT-топіки, якими обмінюються компоненти системи, мають структуровану ієрархічну назву:

— rooms/room/sensor_id;

— command/room/device_id.

Оскільки на етапі розробки та тестування використання реальних фізичних датчиків не завжди є зручним або доцільним, у системі передбачено спеціальну програму-симулятор. Вона імітує роботу сенсорів, що вимірюють параметри навколишнього середовища. Програма створює синтетичні дані, які за структурою та частотою може відповідати реальним показникам з сенсорів, і, за потреби, можливо задати власні параметри.

Наприклад, симулятор може кожні 5 секунд генерувати значення температури в діапазоні від 20 до 30 °C. Такі значення надсилаються на MQTT-брокер у відповідні топіки. Таким чином, контролер «вважає», що отримує дані від реальних пристроїв, і реагує на них відповідно до закладеного алгоритму.

Програма підвантажує з бази даних структуру системи моніторингу, на основі чого формулює список датчиків, які симулюються, разом з їх оптимальними показниками.

Програма-симулятор написана мовою Python із використанням бібліотеки paho-mqtt, яка дозволяє зручно працювати з MQTT-протоколом, та mysql.connector для під'єднання до бази даних.

У розробленій системі клімат-контролю важливу роль відіграє підсистема збереження даних, яка реалізована на основі реляційної бази даних MySQL. Вона забезпечує структуроване збереження інформації про приміщення, типи сенсорів, самі сенсори та їх порогові значення. Така система дозволяє не лише

відслідковувати стан мікроклімату в реальному часі, але й створює основу для аналізу історичних даних, виявлення тенденцій та оптимізації алгоритмів керування.

Однією з важливим частин розробленої інформаційної системи моніторингу та керування кліматом є її мобільність. Вона забезпечується завдяки використанню мікроконтролера ESP32-S3, який є компактним, енергоефективним пристроєм із вбудованими модулями бездротового зв'язку Wi-Fi. Завдяки цьому ESP32-S3 не потребує підключення через кабелі Ethernet або інші дротові інтерфейси. Контролер автоматично підключається до локальної мережі Wi-Fi, що дозволяє йому отримувати дані з сенсорів та надсилати команди виконавчим пристроям без необхідності прокладання будь-яких додаткових комунікацій.

Таким чином, для початку роботи системи достатньо лише під'єднати контролер до джерела живлення (через USB або портативний акумулятор). Це дозволяє розташовувати пристрій у будь-якій частині офісу, наприклад, у переговорній кімнаті, серверній або зоні відпочинку, і за потреби швидко переміщати до іншої зони.

Оскільки програмне забезпечення контролера підтримує автоматичне підключення до хмарного або локального брокера MQTT, це забезпечує незалежність від фізичної інфраструктури. Уся логіка керування реалізована так, що переміщення пристрою не порушує його роботу.

Діаграма архітектури системи представлена на кресленику ІК11.180БАК.006 ДЗ.

3.2 Розробка бази даних

У межах дипломного проєкту було розроблено реляційну базу даних Climate_control, яка є фундаментальною складовою програмно-апаратної системи автоматизованого клімат-контролю офісного середовища. Процес розробки бази даних охоплював етапи проєктування структури, створення таблиць, визначення зв'язків між ними, а також початкове заповнення даними.

На першому етапі була виконана ініціалізація — видалення старої версії бази даних (у разі її наявності) та створення нової бази з іменем Climate_control. Це забезпечує чисте середовище для роботи без залишкових попередніх даних. Після цього встановлюється контекст роботи з новоствореною базою. Створення таблиці наведено на рисунку 3.1.

```
DROP DATABASE Climate_control;

CREATE DATABASE IF NOT EXISTS Climate_control;

USE Climate_control;
```

Рисунок 3.1 – Ініціалізація бази даних

Після створення бази розпочався процес створення її структурних елементів. Усього було розроблено три таблиці, які логічно описують систему.

Перша таблиця – Rooms. Ця таблиця містить список приміщень, у яких встановлюються сенсори. Вона має поля:

— ID_Room — унікальний ідентифікатор кімнати;

— Room_name — назва приміщення (наприклад, "Openspace", "Kitchen").

Таблиця є базовою для просторового розподілу сенсорів. Створення таблиці наведено на рисунку 3.2.

```
CREATE TABLE IF NOT EXISTS Rooms (
    ID_Room int NOT NULL,
    Room_name varchar(30),
    PRIMARY KEY (ID_Room)
);
```

Рисунок 3.2 – Створення таблиці Rooms

Друга таблиця – Sensor_types. Ця таблиця описує категорії сенсорів:

— ID_Sensor_Type — унікальний ідентифікатор типу;

— Sensor_type_name — назва типу сенсора (наприклад, температура, вологість, якість повітря).

Ця таблиця дозволяє стандартизувати перелік типів сенсорів, які використовуються в системі. Створення таблиці наведено на рисунку 3.3.

```
CREATE TABLE IF NOT EXISTS Sensor_types (
    ID_Sensor_Type int NOT NULL,
    Sensor_type_name varchar(30),
    PRIMARY KEY (ID_Sensor_Type)
);
```

Рисунок 3.3 – Створення таблиці Sensor_types

Третя таблиця – Sensors. Центральна таблиця, яка описує кожен сенсор окремо. Вона має поля:

— ID_Sensor — унікальний ідентифікатор сенсора;

— Room — зовнішній ключ на таблицю Rooms;

— Type_sensor — зовнішній ключ на таблицю Sensor_types;

— Sensor_name — логічна назва сенсора (наприклад, “Hall_temperature”);

— Threshold — граничне значення, при недотриманні якого система має реагувати.

Таблиця Sensors пов’язана з двома іншими таблицями зовнішніми ключами, що забезпечує цілісність даних і запобігає логічним помилкам (наприклад, присвоєння сенсора неіснуючому приміщенню чи неіснуючому типу). Створення таблиці наведено на рисунку 3.4.

```
CREATE TABLE IF NOT EXISTS Sensors (
    ID_Sensor int NOT NULL,
    Room int NOT NULL,
    Type_sensor int NOT NULL,
    Sensor_name varchar(30),
    Threshold float,
    PRIMARY KEY (ID_Sensor),
    FOREIGN KEY (Room) REFERENCES Rooms(ID_Room),
    FOREIGN KEY (Type_sensor) REFERENCES Sensor_types(ID_Sensor_Type)
);
```

Рисунок 3.4 – Створення таблиці Sensors

Після створення таблиць база даних була наповнена початковими даними. Заповнення таблиці Rooms наведено на рисунку 3.5.

```
INSERT INTO Rooms (ID_Room, Room_name)
VALUES
(1, "Openspace"),
(2, "HR_cabinet"),
(3, "TechLead_cabinet"),
(4, "Kitchen"),
(5, "WC"),
(6, "Conference_room");
```

Рисунок 3.5 – Заповнення даними таблиці Rooms

Заповнення таблиці Sensor_types наведено на рисунку 3.6.

```
INSERT INTO Sensor_types (ID_Sensor_Type, Sensor_type_name)
VALUES
(1, "Temperature"),
(2, "Humidity"),
(3, "Air_quality");
```

Рисунок 3.6 – заповнення даними таблиці Sensor_types.

Заповнення таблиці Sensors наведено на рисунку 3.7.

```
INSERT INTO Sensors (ID_Sensor, Room, Type_sensor, Sensor_name, Threshold)
VALUES
(1, 1, 1, "Hall_temperature", 25.0),
(2, 1, 2, "Hall_humidity", 60.0),
(3, 1, 3, "Hall_air_quality", 45.0),
(4, 2, 1, "HR_temperature", 24.5),
(5, 2, 2, "HR_humidity", 62.0),
(6, 2, 3, "HR_air_quality", 43.5),
(7, 3, 1, "TechLead_temperature", 26.0),
(8, 3, 2, "TechLead_humidity", 58.0),
(9, 3, 3, "TechLead_air_quality", 47.0),
(10, 4, 1, "Kitchen_temperature", 24.0),
(11, 4, 2, "Kitchen_humidity", 63.5),
(12, 4, 3, "Kitchen_air_quality", 44.0),
(13, 5, 1, "WC_temperature", 25.5),
(14, 5, 2, "WC_humidity", 61.0),
(15, 5, 3, "WC_air_quality", 46.0),
(16, 6, 1, "Conference_temperature", 23.5),
(17, 6, 2, "Conference_humidity", 59.5),
(18, 6, 3, "Conference_air_quality", 42.5);
```

Рисунок 3.7 – Заповнення даними таблиці Sensors

Для кожного сенсора задається відповідне порогове значення — це значення, яке є граничним допустимим показником у нормальному режимі роботи. Наприклад, для сенсора температури в openspace зоні встановлено поріг у 25.0 °C, а для вологості в кухні — 63.5%.

Сутності бази даних наведені в таблиці 3.1.

Таблиця 3.1 – сутності бази даних Climate_control

Таблиця	Поле	Тип даних
Rooms	ID_Room	Primary key
	Room_name	Varchar(30)
Sensor_types	ID_Sensor_Type	Primary key
	Sensor_type_name	Varchar(30)
Sensors	ID_Sensor	Primary key
	Room	Foreign key
	Type_sensor	Foreign key
	Sensor_name	Varchar(30)
	Threshold	Float

ER-діаграма представлена на кресленику ІК11.180БАК.006 Д4.

3.3 Розробка програми контролю клімату

У розробленій системі клімат-контролю важливу роль відіграє програма, реалізована мовою програмування С для мікроконтролера ESP32 з використанням середовища ESP-IDF (ESP IoT Development Framework). Ця програма є головним виконавчим компонентом, що забезпечує безперервну взаємодію між пристроєм, мережею Wi-Fi, MQTT-брокером та зовнішніми джерелами даних, зокрема сенсорами. Основна ідея реалізації полягає у прийомі повідомлень від сенсорів, перевірці отриманих значень на відповідність порогам і формуванні коригувальних команд у разі відхилення.

На початку роботи програма ініціалізує системну пам’ять, а також реалізує підключення до бездротової мережі Wi-Fi. Для цього створюється подієво-орієнтована структура, де обробник подій відповідає за запуск процесу з'єднання після старту та спроби повторного з'єднання у разі втрати доступу до мережі. У випадку успішного отримання IP-адреси пристрій вважається повністю

підключеним до мережі, після чого переходить до наступного етапу — підключення до MQTT-брокера.

MQTT-з’єднання встановлюється за допомогою клієнтської бібліотеки ESP-IDF. Програма ініціалізує MQTT-клієнт, вказуючи адресу брокера, що знаходиться в локальній мережі (за IP 192.168.0.200). Після встановлення з’єднання виконується підписка на певну групу топіків, зокрема rooms_lev/+/+, де + є шаблоном для підключення до всіх підтем: наприклад, rooms_lev/room_1/sensor_2 або rooms_lev/room_3/sensor_1. Це дозволяє системі отримувати повідомлення з будь-якої кімнати та будь-якого сенсора без жорсткого зв’язування з конкретними ID.

Після отримання повідомлення MQTT система виконує його обробку. Усі події обробляються асинхронно через подієвий обробник, що перевіряє ID приміщення та сенсора, вичитуючи їх із назви топіка. Дані, що містяться у повідомленні, інтерпретуються як числове значення, яке порівнюється із заздалегідь заданим порогом. Ці пороги зберігаються у внутрішньому масиві sensor_thresholds, який містить список усіх сенсорів і відповідних порогових значень, що були попередньо встановлені.

Після обчислення відхилення від порогового значення виконується логіка прийняття рішення. Якщо значення відхиляється від порогу на більше ніж 10%, система розраховує нове значення — часткову корекцію у напрямку норми. Це значення формує команду у вигляді нового повідомлення MQTT, яке публікується в окремий керуючий топік command_lev/room_X/device_Y. Якщо ж відхилення є незначним, тобто значення залишається в межах допустимої похибки, система просто публікує назад порогове значення як цільове, сигналізуючи, що серйозного коригування не потрібно.

Важливо зазначити, що вся система побудована як подієва, тобто не виконує безперервне опитування, а реагує лише на ті події, які надходять. Це значно знижує навантаження на процесор і мережу. Крім того, завдяки використанню FreeRTOS система є багатозадачною, стабільною й здатною до обробки великої кількості повідомлень без втрати продуктивності.

Програма є тісно пов’язаною з архітектурою бази даних і логікою MQTT-топіків. Наприклад, сенсор із ідентифікатором 6, що відповідає за якість повітря в HR-кабінеті, повинен мати порогове значення, яке збігається з даними в БД (43.5), і цей поріг використовується в обчисленнях. Таким чином, наявність єдиного інформаційного простору між мікроконтролером, базою даних і MQTT забезпечує цілісність системи.

Діаграма алгоритму програми представлена на кресленику ІК11.180БАК.006 Д1.

Розглянемо кожну функцію даної програми.

Функція event_handler() – універсальний обробник подій Wi-Fi та IP. Він контролює процес з’єднання з точкою доступу та обробляє події, пов’язані з мережею. Коли пристрій запускається (WIFI_EVENT_STA_START), виконується спроба підключення до Wi-Fi (esp_wifi_connect()). Якщо з’єднання втрачене (WIFI_EVENT_STA_DISCONNECTED), виконується до 10 повторних спроб з’єднання. Якщо пристрій отримав IP (IP_EVENT_STA_GOT_IP), встановлюється відповідний біт у групі подій, сигналізуючи, що Wi-Fi з’єднання готове. Код функції наведено на рисунку 3.8.

```
static void event_handler(void* arg, esp_event_base_t event_base,
                          int32_t event_id, void* event_data)
{
    if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START) {
        esp_wifi_connect();
    } else if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_DISCONNECTED) {
        if (s_retry_num < 10) {
            esp_wifi_connect();
            s_retry_num++;
            ESP_LOGI(TAG, "Retry to connect to the AP");
        } else {
            xEventGroupSetBits(s_wifi_event_group, WIFI_FAIL_BIT);
        }
        ESP_LOGI(TAG,"Connection to the AP has failed");
    } else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP) {
        ip_event_got_ip_t* event = (ip_event_got_ip_t*) event_data;
        ESP_LOGI(TAG, "Got IP:" IPSTR, IP2STR(&event->ip_info.ip));
        s_retry_num = 0;
        xEventGroupSetBits(s_wifi_event_group, WIFI_CONNECTED_BIT);
    }
}
```

Рисунок 3.8 – Код функції event_handler()

Функція get_threshold_for_sensor() виконує пошук порогового значення для заданого sensor_id. Пошук відбувається у масиві sensor_thresholds. Якщо сенсор знайдено — повертається його поріг. Якщо не знайдено — повертається -1, що сигналізує про помилку. Код функції наведено на рисунку 3.9.

```c
// ==== Функція пошуку порогового значення ====
float get_threshold_for_sensor(int sensor_id) {
    for (int i = 0; i < MAX_SENSORS; i++) {
        if (sensor_thresholds[i].sensor_id == sensor_id) {
            return sensor_thresholds[i].threshold;
        }
    }
    return -1; // не знайдено
}
```

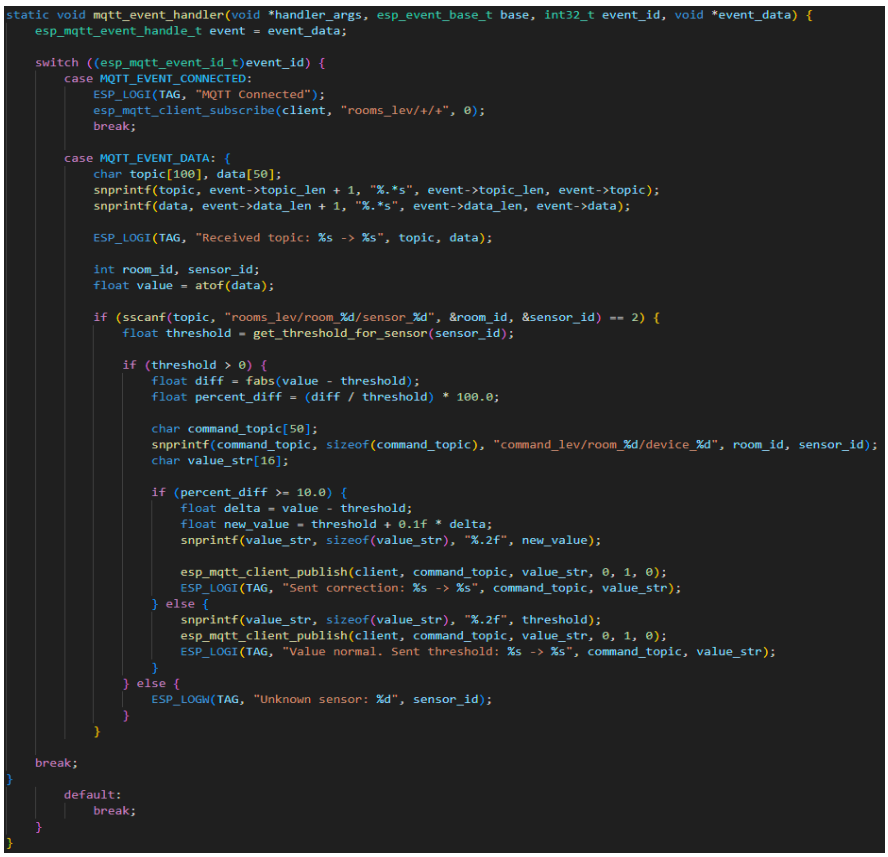
Рисунок 3.9 – Код функції get_threshold_for_sensor()

Функція mqtt_app_start() виконує ініціалізацію MQTT-клієнта та запуск зв'язку з брокером. Вона Формує конфігурацію MQTT (mqtt_cfg), вказуючи URI брокера. Клієнт ініціалізує через esp_mqtt_client_init(). Також вона реєструє обробник подій MQTT (mqtt_event_handler), та, власне, запускає MQTT-клієнт (esp_mqtt_client_start()). Код функції наведено на рисунку 3.10.

```c
// ==== Ініціалізація MQTT ====
static void mqtt_app_start(void) {
    esp_mqtt_client_config_t mqtt_cfg = {
        .broker.address.uri = MQTT_BROKER_URI,
    };

    client = esp_mqtt_client_init(&mqtt_cfg);
    esp_mqtt_client_register_event(client, ESP_EVENT_ANY_ID, mqtt_event_handler, NULL);
    esp_mqtt_client_start(client);
}
```

Рисунок 3.10 – Код функції mqtt_app_start()

Функція mqtt_event_handler(). Відповідає за логіку реакції на повідомлення, отримані з брокера. При MQTT_EVENT_CONNECTED виконує підписку на всі сенсорні повідомлення (топік rooms_lev/+/+). При MQTT_EVENT_DATA зчитує назву топіка та значення повідомлення, витягує room_id та sensor_id із назви топіка, перетворює отримане значення (data) у число, визначає поріг за sensor_id. Якщо відхилення більше ніж 10%, виконує корекцію: додає 10% до напрямку повернення до порогу. Якщо відхилення незначне — просто надсилає порогове значення назад як "норму". Далі формує нове повідомлення у топік command_lev/room_X/device_Y. Код функції наведено на рисунку 3.11.



```c
static void mqtt_event_handler(void *handler_args, esp_event_base_t base, int32_t event_id, void *event_data) {
    esp_mqtt_event_handle_t event = event_data;

    switch ((esp_mqtt_event_id_t)event_id) {
        case MQTT_EVENT_CONNECTED:
            ESP_LOGI(TAG, "MQTT Connected");
            esp_mqtt_client_subscribe(client, "rooms_lev/+/+", 0);
            break;

        case MQTT_EVENT_DATA: {
            char topic[100], data[50];
            snprintf(topic, event->topic_len + 1, "%.*s", event->topic_len, event->topic);
            snprintf(data, event->data_len + 1, "%.*s", event->data_len, event->data);

            ESP_LOGI(TAG, "Received topic: %s -> %s", topic, data);

            int room_id, sensor_id;
            float value = atof(data);

            if (sscanf(topic, "rooms_lev/room_%d/sensor_%d", &room_id, &sensor_id) == 2) {
                float threshold = get_threshold_for_sensor(sensor_id);

                if (threshold > 0) {
                    float diff = fabs(value - threshold);
                    float percent_diff = (diff / threshold) * 100.0;

                    char command_topic[50];
                    snprintf(command_topic, sizeof(command_topic), "command_lev/room_%d/device_%d", room_id, sensor_id);
                    char value_str[16];

                    if (percent_diff >= 10.0) {
                        float delta = value - threshold;
                        float new_value = threshold + 0.1f * delta;
                        snprintf(value_str, sizeof(value_str), "%.2f", new_value);

                        esp_mqtt_client_publish(client, command_topic, value_str, 0, 1, 0);
                        ESP_LOGI(TAG, "Sent correction: %s -> %s", command_topic, value_str);
                    } else {
                        snprintf(value_str, sizeof(value_str), "%.2f", threshold);
                        esp_mqtt_client_publish(client, command_topic, value_str, 0, 1, 0);
                        ESP_LOGI(TAG, "Value normal. Sent threshold: %s -> %s", command_topic, value_str);
                    }
                } else {
                    ESP_LOGW(TAG, "Unknown sensor: %d", sensor_id);
                }
            }

    break;
}
        default:
            break;
    }
}
```

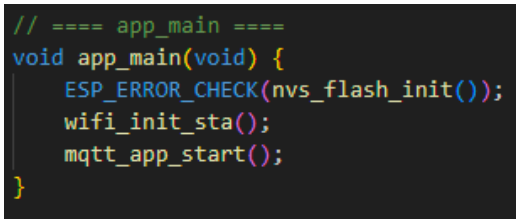
Рисунок 3.11 – Код функції mqtt_event_handler()

Функція wifi_init_sta() виконує налаштування та запуск Wi-Fi у режимі клієнта. Вона ніціалізує мережеві інтерфейси ESP-IDF, створює подієвий цикл, конфігурує параметри мережі (SSID, пароль), встановлює режим Station. реєструє обробники подій (event_handler), стартує Wi-Fi та очікує завершення з'єднання або помилку (через xEventGroupWaitBits()). Код функції наведено на рисунку 3.12.

```
// ==== Ініціалізація Wi-Fi ====
static void wifi_init_sta(void) {
    s_wifi_event_group = xEventGroupCreate();

    ESP_ERROR_CHECK(esp_netif_init());

    ESP_ERROR_CHECK(esp_event_loop_create_default());
    esp_netif_create_default_wifi_sta();

    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));

    esp_event_handler_instance_t instance_any_id;
    esp_event_handler_instance_t instance_got_ip;
    ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT,
                                                        ESP_EVENT_ANY_ID,
                                                        &event_handler,
                                                        NULL,
                                                        &instance_any_id));
    ESP_ERROR_CHECK(esp_event_handler_instance_register(IP_EVENT,
                                                        IP_EVENT_STA_GOT_IP,
                                                        &event_handler,
                                                        NULL,
                                                        &instance_got_ip));

    wifi_config_t wifi_config = {
        .sta = {
            .ssid = WIFI_SSID,
            .password = WIFI_PASS,
            .threshold.authmode = 0,
        },
    };
    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA) );
    ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_STA, &wifi_config) );
    ESP_ERROR_CHECK(esp_wifi_start() );

    ESP_LOGI(TAG, "wifi_init_sta finished.");

    EventBits_t bits = xEventGroupWaitBits(s_wifi_event_group,
            WIFI_CONNECTED_BIT | WIFI_FAIL_BIT,
            pdFALSE,
            pdFALSE,
            portMAX_DELAY);

    if (bits & WIFI_CONNECTED_BIT) {
        ESP_LOGI(TAG, "connected to ap SSID:%s password:%s",
                 WIFI_SSID, WIFI_PASS);
    } else if (bits & WIFI_FAIL_BIT) {
        ESP_LOGI(TAG, "Failed to connect to SSID:%s, password:%s",
                 WIFI_SSID, WIFI_PASS);
    } else {
        ESP_LOGE(TAG, "UNEXPECTED EVENT");
    }
}
```

Рисунок 3.12 – Код функції wifi_init_sta()

Функція app_main(). Це головна функція всієї програми, яка викликається після запуску пристрою. Вона ініціалізує flash-пам'ять (nvs_flash_init()), запускає підключення до Wi-Fi (wifi_init_sta()) та запускає MQTT-клієнт (mqtt_app_start()). Код функції наведено на рисунку 3.13.



```
// ==== app_main ====
void app_main(void) {
    ESP_ERROR_CHECK(nvs_flash_init());
    wifi_init_sta();
    mqtt_app_start();
}
```

Рисунок 3.13 – Код функції app_main()

3.4 Розробка програми симуляції датчиків

Ця програма реалізує програмну імітацію роботи сенсорів, які використовуються в системі моніторингу клімату. Мета цієї програми — регулярно

передавати випадкові значення, що імітують показники сенсорів, у систему через протокол MQTT. При цьому дані про самі сенсори — їхній тип та розташування — беруться з бази даних MySQL, що забезпечує гнучкість та масштабованість моделі.

Робота програми починається з налаштувань підключення до двох ключових систем: MQTT-брокера та бази даних MySQL. MQTT використовується як транспорт для передачі повідомлень, а база даних — як джерело інформації про доступні сенсори, зокрема їхній ідентифікатор, до якої кімнати вони прив'язані та якого вони типу. Саме від типу сенсора залежить діапазон значень, які буде імітувати програма. Наприклад, температурний сенсор може генерувати значення від 10.0 до 40.0, в той час як інші типи можуть працювати в ширшому або іншому діапазоні.

Після запуску програма встановлює підключення до бази даних і виконує SQL-запит, щоб отримати список усіх сенсорів. Потім створюється MQTT-клієнт, який підключається до брокера й починає працювати у фоновому режимі. Основна логіка роботи реалізована в нескінченному циклі: для кожного сенсора з бази програма генерує випадкове значення, формує MQTT-тему за шаблоном, який включає номер кімнати та ідентифікатор сенсора, і надсилає туди це значення як текстове повідомлення. Після публікації всіх значень програма робить паузу (15 секунд) і повторює весь процес заново.

Програма також має обробку переривання користувачем (наприклад, натискання Ctrl+C). У цьому випадку вона завершує роботу коректно: зупиняє MQTT-клієнт і розриває з'єднання.

Діаграма алгоритму програми представлена на кресленику ІК11.180БАК.006 Д2.

Дана програма використовує сторонні бібліотеки. paho-mqtt застосовується для створення клієнту MQTT, підключення до брокера та обміну даними. mysql-connector в свою чергу застосовується для встановлення з базами даних MySQL, створення курсору для виконання запитів SQL, їх виконання та отримання результатів. Обидві бібліотеки необхідно встановити окремо.

Розглянемо наявні функції програми.

Функція generate_sensor_value(sensor_type) генерує випадкове значення на основі типу сенсора. Якщо тип сенсора 1, то генерується значення з визначеного проміжку, якщо тип сенсору 2, то проміжок, з якого вибирається значення, вже є іншим, і так для кожного прописаного типу. Код функції наведено на рисунку 3.14.

```
def generate_sensor_value(sensor_type):
    if sensor_type == 1:
        return round(random.uniform(1.5, 300.0), 2)
    elif sensor_type == 2:
        return round(random.uniform(1, 800.0), 2)
    elif sensor_type == 3:
        return round(random.uniform(1.5, 800.0), 2)
    else:
        return 0
```

Рисунок 3.14 – Код функції generate_sensor_value(sensor_type)

Функція get_sensors_from_db() отримує з бази даних список сенсорів, їхні ідентифікатори, типи та кімнати. Вона підключається до бази даних з використанням mysql.connector, виконує SQL-запит, що об'єднує таблиці Sensors, Rooms і Sensor_types, отримує результат як список словників (dictionary=True). Закриває підключення та повертає список сенсорів. Код функції наведено на рисунку 3.15.

```
# ==== Отримання списку сенсорів ====
def get_sensors_from_db():
    conn = mysql.connector.connect(**MYSQL_CONFIG)
    curs = conn.cursor(dictionary=True)

    query = "SELECT s.ID_Sensor, r.ID_Room, st.ID_Sensor_Type FROM Sensors s JOIN Rooms r ON s.Room = r.ID_Room JOIN Sensor_types st ON s.Type_sensor = st.ID_Sensor_Type"

    curs.execute(query)
    print(curs.rowcount)
    sensors = curs.fetchall()

    curs.close()
    conn.close()
    return sensors
```

Рисунок 3.15 – Код функції get_sensors_from_db()

Функція on_connect(client, userdata, flags, rc) є callback-функцією для з’єднання з MQTT-брокером. Приймає параметри client, що і є MQTT-клієнтом, та userdata, flags, rc — службові параметри MQTT (код результату, статуси тощо). Код функції наведено на рисунку 3.16.

```python
# ==== MQTT Callback ====
def on_connect(client, userdata, flags, rc):
    print(f"[MQTT] Connected with result code {rc}")
```

Рисунок 3.16 – Код функції get_sensors_from_db()

Функція main() є основною функцією програми, яка ініціює симуляцію та керує циклом публікацій. Вона отримує список сенсорів з бази через get_sensors_from_db(), ініціалізує MQTT-клієнта та підключається до брокера. Далі, у нескінченному циклі, проходить по кожному сенсору, генерує значення через generate_sensor_value, формує MQTT-топік, формує MQTT-тему й чекає 15 секунд. При введенні KeyboardInterrupt — закриває з'єднання з брокером та зупиняє роботу програми. Код функції наведено на рисунку 3.17.

```python
# ==== Основна функція ====
def main():
    print("Retrieveing sensors...")
    sensors = get_sensors_from_db()

    print(sensors)

    if not sensors:
        print("No sensors added to DB")
        return

    client = mqtt.Client()
    client.on_connect = on_connect
    client.connect(MQTT_BROKER, MQTT_PORT, 60)
    client.loop_start()

    try:
        while True:
            for sensor in sensors:
                sensor_id = sensor['ID_Sensor']
                room = sensor['ID_Room']
                sensor_type = sensor['ID_Sensor_Type']

                value = generate_sensor_value(sensor_type)
                topic = f"rooms_lev/room_{room}/sensor_{sensor_id}"
                message = str(value)

                client.publish(topic, message)
                print(f"Published: {topic} -> {message}")

            time.sleep(PUBLISH_INTERVAL)

    except KeyboardInterrupt:
        print("Stopped by the user.")
    finally:
        client.loop_stop()
        client.disconnect()

if __name__ == "__main__":
    main()
```

Рисунок 3.17 – Код функції main()

Висновок до розділу 3

В даному розділі було створено архітектуру інформаційної системи клімат контролю. Вона складається з чотирьох частин: програми-контролера, MQTT-

брокера, симулятора даних з датчика та бази даних. База даних зберігає конфігурацію системи клімат контролю, симулятор підвантажує цю конфігурацію та формує список датчиків, для яких і генерує значення. MQTT-брокер отримує ці значення, оформлені в топіки. Контролер підписується на ці топіки, отримує ними дані, порівнює з вбудованими параметрами та надсилає команду через MQTT-брокера на відповідний пристрій в залежності від порівняння.

Створена база даних MySQL складається з трьох таблиць: кімант, типів датчиків та самих датчиків. Таким чином зберігається конфігурація системи.

Було реалізовано програму для контролера, яка виконує обробку кліматичних даних у режимі реального часу. Основною метою цієї програми є забезпечення стабільної роботи системи автоматизованого клімат-контролю: вона приймає вхідні дані про стан середовища, аналізує їх відповідно до заданих порогів і формує керуючі команди у разі відхилень від норми. Програма призначена для взаємодії з MQTT-брокером, звідки вона отримує повідомлення від сенсорів (реальних або віртуальних), зокрема значення температури, вологості, концентрації CO_2 та інших параметрів.

Було реалізовано програму-симулятор, що виконує функції віртуального джерела кліматичних даних. Її головною метою є створення умов для тестування та налаштування системи автоматизованого клімат-контролю без необхідності використання реальних фізичних сенсорів. Програма дозволяє моделювати показники температури, вологості, концентрації CO_2 та інших параметрів у режимі реального часу, формуючи відповідні повідомлення, що передаються до MQTT-брокера.

Для реалізації передачі даних використано бібліотеку paho-mqtt, яка забезпечує стабільне з'єднання з брокером та передачу повідомлень відповідно до MQTT-протоколу. Для взаємодії з базою даних застосовано mysql.connector, що дозволяє виконувати запити до структури системи та отримувати актуальний список сенсорів для моделювання.

4 ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

Після завершення проєктування та реалізації програмних компонентів системи автоматизованого клімат-контролю було проведено тестування, метою якого є перевірка працездатності, стабільності та відповідності функціональності заявленим вимогам. Тестування дозволяє виявити можливі помилки в логіці обробки даних, комунікації між модулями, а також перевірити реакцію системи на типові та критичні ситуації — наприклад, перевищення порогових значень температури чи вологості.

Для тестування запустимо систему всю.

В першу чергу запустимо MQTT-брокер. Для цього перейдемо в каталог, де розміщуються файли брокера. В моєму випадку це каталог: C:\Program Files\mosquitto (див. рисунок 4.1).

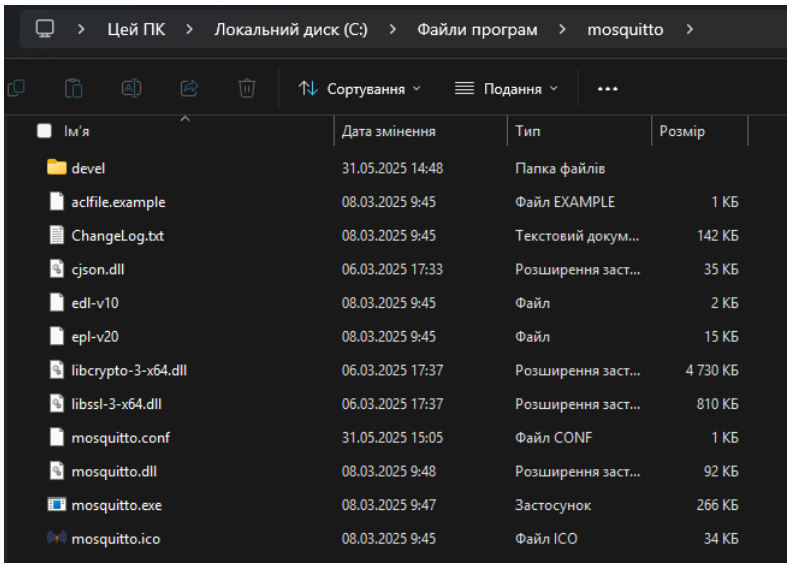


Рисунок 4.1 – Каталог mosquitto, реалізації протоколу MQTT

Якщо просто запустити mosquitto.exe, то брокер запуститься в режимі лише для цього пристрою, тому треба запустити його з конфігураційним файлом mosquitto.conf з певним змістом (див. рисунок 4.2).

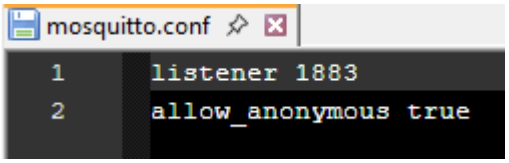


Рисунок 4.2 – Вміст файлу mosquitto.conf

Для запуску програми відкриємо cmd з правами адміністратора в цьому каталозі та вводимо наступну команду: mosquitto.exe -c "C:\Program Files\mosquitto\mosquitto.conf" -v (див. рисунок 4.3).

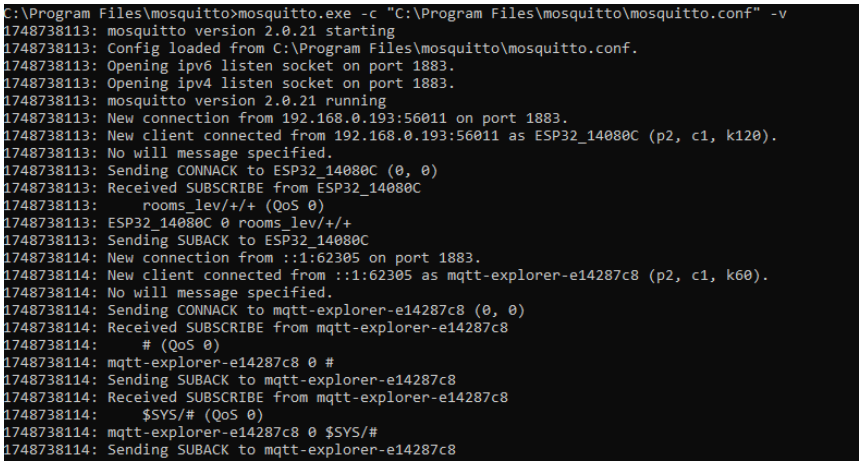


Рисунок 4.3 – Старт MQTT-брокера

Наступним чином запускаємо програму MQTT Explorer. Це MQTT клієнт, який під’єднується до брокера і відображає всі топіки. Це корисно та зручно для відслідковування всіх даних, що надходять на нього. Вводимо параметри брокера та під’єднуємось (див. рисунок 4.4).

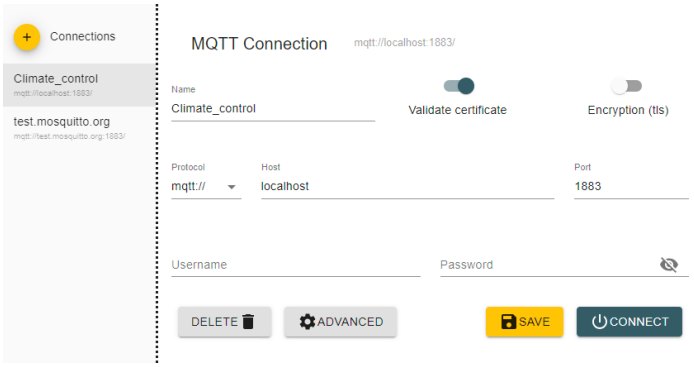


Рисунок 4.4 – Вікно під’єднання до брокера в MQTT Explorer

Під’єднавшись, маємо вигляд всіх даних, що до нього надходять. Наразі це лише системні дані, які необхідні для роботи програми (див. рисунок 4.5).

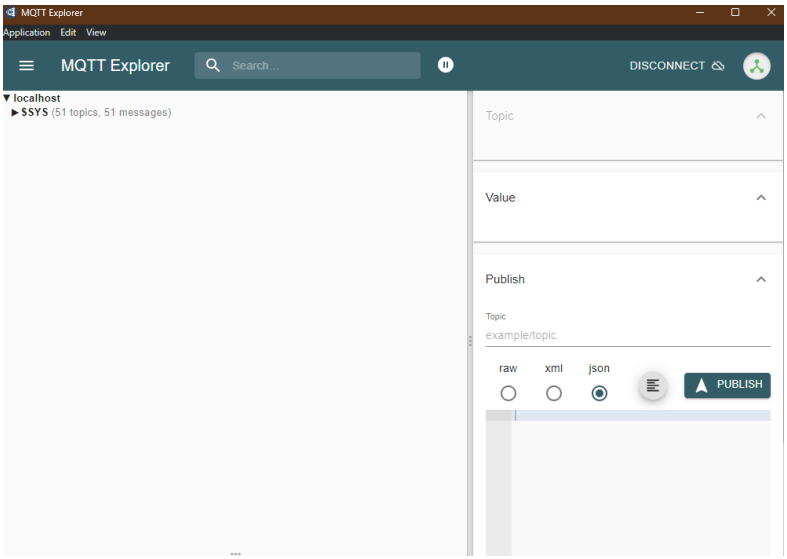


Рисунок 4.5 – Вікно брокера, до якого під’єднались

Далі, запускаємо програму симулятор. Можна побачити, що програма запустилась, встановила підключення з базою даних, отримала конфігурацію системи, підключилась до брокера та надсилає дані, про що пише в термінал (див. рисунок 4.6).

```
Retrieveing sensors...
0
[{'ID_Sensor': 1, 'ID_Room': 1, 'ID_Sensor_Type': 1}, {'ID_Sensor': 4, 'ID_Room': 2, 'ID_Sensor_Type': 1}, {'ID_Sensor': 7, 'ID_Room': 3, 'ID_Sensor_Type':
': 6, 'ID_Sensor_Type': 1}, {'ID_Sensor': 2, 'ID_Room': 1, 'ID_Sensor_Type': 2}, {'ID_Sensor': 5, 'ID_Room': 2, 'ID_Sensor_Type': 2}, {'ID_Sensor': 8, 'ID_
 {'ID_Sensor': 17, 'ID_Room': 6, 'ID_Sensor_Type': 2}, {'ID_Sensor': 3, 'ID_Room': 1, 'ID_Sensor_Type': 3}, {'ID_Sensor': 6, 'ID_Room': 2, 'ID_Sensor_Type'
': 5, 'ID_Sensor_Type': 3}, {'ID_Sensor': 18, 'ID_Room': 6, 'ID_Sensor_Type': 3}]
d:\Diploma\Utilitty\sensor_simulator.py:60: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
  client = mqtt.Client()
Published: rooms_lev/room_1/sensor_1 -> 133.05
Published: rooms_lev/room_2/sensor_4 -> 143.45
Published: rooms_lev/room_3/sensor_7 -> 64.1
Published: rooms_lev/room_4/sensor_10 -> 54.82
Published: rooms_lev/room_5/sensor_13 -> 122.39
```

Рисунок 4.6 – Робота програми симулятора

Переходимо назад до MQTT Explorer. Бачимо, що з’явився пункт з такою назвою топіка, яку надсилає програма. Розкриваємо його і бачимо номери кімнат. Розкриваємо всіх їх і бачимо показники датичків з кожної кімнати (див. рисунок 4.7).

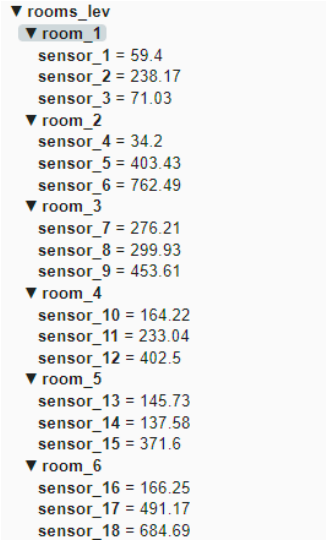


Рисунок 4.7 – Отримання даних з симулятора брокером

Запускаємо програму контролера. Бачимо, що до мережі контролер під'єднався, так само і до брокера та вже отримує і надсилає дані назад (див. рисунок 4.8).

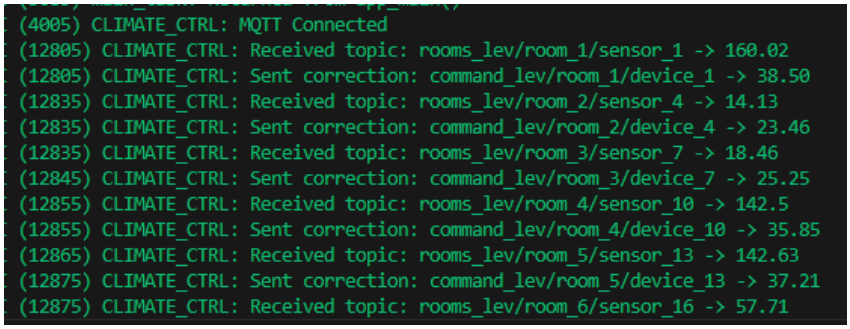


Рисунок 4.8 – Робота програми мікроконтролера

Перевіряємо, чи відображаються ці дані в MQTT Explorer (див. рисунок 4.9).

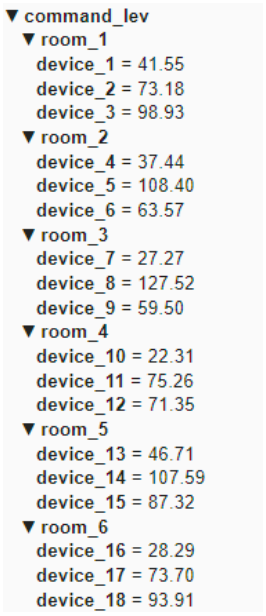


Рисунок 4.9 – отримання даних з контролера брокером.

Почекавши певний час, бачимо, що дані оновилися, а, значить, симулятор нові дані згенерував, передав їх на брокер, контролер ці дані з брокера отримав, їх проаналізував, дав свою відповідь і брокер це теж записав.

На останок, виставимо частоту в 1 секунду для перевірки роботи за високої завантаженості. Бачимо швидше оновлення даних періодом в 1 секунду, як і було вказано. Тобто система може оброблювати велику кількість оновлень (див. рисунок 4.10).

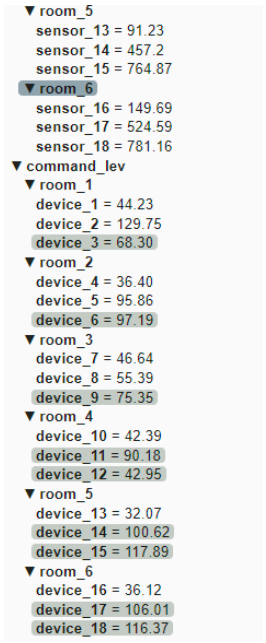


Рисунок 4.10 – оновлення даних з періодом в 1 секунду.

Штатно завершуємо роботу симулятора та брокера комбінацією клавіш Crtl + C, а контролера – Crtl +].

Результати тестування показали, що розроблена система автоматизованого клімат-контролю працює ефективно та стабільно в умовах, наближених до реального середовища. Завдяки використанню MQTT-протоколу, взаємодія між симулятором сенсорів, контролером та базою даних відбувається без затримок і з високою надійністю. Контролер своєчасно отримує дані про температуру, вологість та концентрацію CO_2, коректно аналізує їх у режимі реального часу та формує відповідні керуючі команди згідно з логікою алгоритму.

У ході випробувань система демонструвала стійку роботу навіть за умов швидкої зміни вхідних параметрів. Це підтверджує правильність реалізації логіки контролера, ефективність механізму порівняння з порогами та стабільність передачі даних через MQTT-брокер. Завдяки підключеному GUI (графічному інтерфейсу користувача), тестування стало більш наочним і зручним, що дозволило оперативно контролювати процеси в системі.

Отримані результати підтверджують життєздатність створеної системи клімат-контролю та її придатність до подальшого використання, зокрема для впровадження в офісних, навчальних або виробничих приміщеннях. Надійність обміну даними, гнучка модульна структура та можливість масштабування системи свідчать про високий потенціал її подальшого розвитку, розширення функціональності та адаптації до нових умов експлуатації.

ВИСНОВКИ

У даному дипломному проєкті було розроблено і реалізовано систему автоматизованого клімат-контролю для офісного приміщення, основною метою якої є створення надійного, адаптивного та ефективного рішення для моніторингу й регулювання параметрів мікроклімату в режимі реального часу. Така система дозволяє забезпечити комфортні умови для перебування людей, знизити витрати на енергоспоживання та підвищити загальну ефективність використання інженерного обладнання приміщення. Архітектура системи побудована за модульним принципом, що забезпечує гнучкість у її розгортанні, простоту модернізації та динамічну взаємодію між її основними компонентами — сенсорами, контролером, базою даних і MQTT-брокером.

Для досягнення поставленої мети в проєкті було використано сучасні технології обміну даними та інструменти розробки, що забезпечують масштабованість та стабільність функціонування. Передавання показників температури, вологості, концентрації CO_2 та інших параметрів реалізовано через легковаговий MQTT-протокол, який відзначається високою швидкістю передачі повідомлень, низькими затримками та надійністю при роботі в локальних і хмарних мережах. Крім того, у рамках проєкту створено віртуальний симулятор сенсорів, який дозволяє генерувати дані в режимі реального часу без потреби у фізичному підключенні датчиків. Це значно спростило процес налагодження системи, прискорило розробку та дало змогу повноцінно протестувати логіку роботи контролера на ранніх етапах.

Ключовим елементом системи є програмно-апаратний контролер, побудований на основі сучасного мікроконтролера ESP32-S3. Він відповідає за приймання, аналіз і обробку вхідних даних, порівняння їх із заздалегідь заданими пороговими значеннями та формування відповідних керуючих сигналів для виконавчих пристроїв, таких як вентилятори, нагрівачі, зволожувачі тощо. Програмна частина контролера реалізована із застосуванням фреймворку ESP-IDF, що забезпечує стабільну роботу в реальному часі завдяки використанню

операційної системи FreeRTOS. Це дало змогу ефективно розподілити обчислювальні ресурси, організувати паралельне виконання завдань та підвищити загальну продуктивність пристрою.

Особливу увагу в проєкті було приділено процесу тестування системи та візуальному моніторингу її роботи. Для цього було використано інструмент MQTT Explorer, який дозволяє наочно відслідковувати передачу повідомлень, структуру тем, значення параметрів, а також відповідність керуючих команд поточному стану середовища. Тестування охоплювало різноманітні сценарії роботи, включаючи зміну умов довкілля, перевірку реакції системи на перевищення порогів, а також стабільність зв'язку між симулятором і контролером.

Результати випробувань продемонстрували високу точність реагування системи, надійність роботи комунікаційного каналу та відповідність логіки контролера заданим технічним умовам. Система повністю задовольняє вимоги, сформульовані на етапі проєктування, і може бути з успіхом адаптована для впровадження в офісних приміщеннях, навчальних закладах, лабораторіях та інших середовищах, де необхідне постійне підтримання стабільного мікроклімату.

Отримані результати підтверджують технічну доцільність запропонованого підходу та свідчать про високий потенціал розробленої системи до подальшого вдосконалення, масштабування та інтеграції в інші інтелектуальні системи управління будівлями.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rhys Goldstein, Alex Tessier, Azam Khan. Space layout in occupant behavior simulation. 2011. 8 с.
2. Специфікація контролера Honeywell HCE80. URL: https://dhb3yazwboecu.cloudfront.net/284/hon-hbc/HCE80_Specification_sheet.pdf
3. Специфікація контролера Honeywell HCE100. URL: https://evohomeshop.hu/custom/honeywell/image/data/srattached/642553ef5b6f60b816626caeb0a397e5_Resideo_HCC100.pdf?srsltid=AfmBOorPfPDQxttPe7X-cNlgvpkRMnIHUNCAvx_xN_pDSaW09p-8SroO
4. Специфікація Honeywell WEBs-AX. URL: http://www.alliedsuccess.com/pdf/Honeywell/Webs-AX.pdf
5. Базова документація системи Siemens Climatix. URL: https://www.etsnord.fi/wp-content/uploads/2020/12/Siemens-Climatix-Basic-documentation.pdf
6. Офіційна документація системи Johnson Controls Metasys System. URL: https://docs.johnsoncontrols.com/bas/r/Metasys/en-US/Metasys-System-Configuration-Guide/14.0/Introduction
7. Специфікація системи Schneider Electronics SmartX. URL: https://iportal2.schneider-lectric.com/Contents/docs/LSS_CRS_Optimum_Datasheet_F-28082-4.pdf
8. Офіційна сторінка серії контролерів STMicroelectronics SMT32H7. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32h7-series.html
9. Espressif Systems. ESP32-S3 Series Datasheet Version 2.0. 2025. 84 с.
10. NXP Semiconductors. i.MX RT1060 Crossover Processors for Consumer Product. 2024. 119 с.
11. Raspberry Pi Ltd. RP2040 Datasheet. A microcontroller by Raspberry Pi. 2025. 642 с.
12. Офіційна сторінка документації ESP-IDF. URL: https://idf.espressif.com/

13. Офіційна сторінка документації MicroPython. URL: https://docs.micropython.org/en/latest/

14. R. Ierusalimschy, L. H. de Figueiredo, W. Celes. Lua 5.1 Reference Manual. 2006. 112 с.

15. Офіційна документація мови Rust. URL: https://doc.rust-lang.org/book/

16. Офіційна документація MS SQL Server. URL: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver17

17. Rick Silva. MySQL Crash Course: A Hands-on Introduction to Database Development. 2023. 352 с.

18. Офіційна документація SQLite. URL: https://www.sqlite.org/docs.html

19. The PostgreSQL Global Development Group. PostgreSQL 17.5 Documentation. 2025. 3044 с.

20. Sandra Sendra, Miguel Garcia, Carlos Turro, Jaime Lloret. WLAN IEEE 802.11a/b/g/n Indoor Coverage and Interference Performance Study. 2011. 15 с.

21. Muhammad Shamaas. Introduction to Bluetooth Low Energy. 2022. 7 с.

22. Офіційна документація ESP-NOW. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_now.html

23. OASIS. MQTT Version 5.0. OASIS Standart. 2019. 137 с.

24. Офіційна документація HTTP/REST. URL: https://restfulapi.net/

25. Офіційна документація WebSocket. URL: https://datatracker.ietf.org/doc/html/rfc6455