

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”

Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор

_____ Олексій СМІРНОВ

“ _____ ” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

за першим (бакалаврським) рівнем вищої освіти

на тему

**“ Програмне забезпечення системи інтелектуального керування
мікрокліматом промислових приміщень на основі аналізу
сенсорних даних та оптимізаційних алгоритмів ”**

Виконав здобувач вищої освіти

IV курсу, групи _____

ОПП «Комп’ютерні науки»

спеціальності 122 «Комп’ютерні науки»

_____ Кохан І.О.

« _____ » _____ 2026р.

Керівник проекту

кандидат технічних наук, доцент

_____ Минайленко Р.М

« _____ » _____ 2026р.

Рецензент _____

м. Кропивницький

K6ПЗ_2026

Центральноукраїнський національний технічний університет

Факультет Механіко-технологічний

Кафедра Кібербезпеки та програмного забезпечення

Рівень вищої освіти бакалавр

Галузь знань . _12_ “Інформаційні технології”

Спеціальність 122 “Комп’ютерні науки”

Освітньо-професійна (освітньо-наукова) програма “Комп’ютерні науки”

ЗАТВЕРДЖУЮ

Завідувач кафедри

д.т.н., проф.

_____Олексій СМІРНОВ

“06” лютого 2026 року

**ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ
(БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ**

Кохана Ігора Олександровича

(прізвище, ім’я, по батькові)

1.Тема роботи Програмне забезпечення системи інтелектуального керування мікрокліматом промислових приміщень на основі аналізу сенсорних даних та оптимізаційних алгоритмів

2.Керівник роботи Минайленко Роман Миколайович, канд. техн. наук, доцент

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти № 116-02 від 06.02 2026року

Строк подання роботи до захисту 22.05.2026 р.

Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи інтелектуального керування мікрокліматом промислових приміщень на основі аналізу сенсорних даних та оптимізаційних алгоритмів*

5.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання. _____
- 2.Перегляд аналогічних існуючих систем. _____
- 3.Опис і обґрунтування проектних рішень. _____
- 4.Етапи програмування системи. _____
- 5.Впровадження системи в промислову експлуатацію. _____
- 6.Висновки. _____

6.Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи 1 аркуш

Функціональна схема системи 1 аркуш

Діаграма процесів 1 аркуш

Блок-схема алгоритму роботи додатку 2 аркуша

Дата видачі завдання «06» лютого 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем керування	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.05.2026 р.	
8.	Попередній захист роботи	22.05.2026 р.	

Дата видачі завдання

«06» 02 2026р.

Підпис керівника

(прізвище та ініціали)

Завдання прийнято до виконання

«06» 02 2026р.

Підпис здобувача

(прізвище та ініціали)

K6ПЗ_2026

АНОТАЦІЯ

Кохан І.О. Програмне забезпечення системи інтелектуального керування мікрокліматом промислових приміщень на основі аналізу сенсорних даних та оптимізаційних алгоритмів. 122 Комп'ютерні Науки. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення системи інтелектуального керування мікрокліматом складських і промислових приміщень.

Метою розробки є програмне забезпечення системи автоматичного контролю та регулювання температури і відносної вологості повітря на основі сенсорних даних та ПІД-алгоритму керування.

Результат роботи - програмна реалізація трирівневої системи: вбудоване програмне забезпечення мікроконтролера ESP32 з датчиком DHT22, серверна частина на платформі ASP.NET Core 8 з базою даних SQLite та веб-інтерфейс оператора.

В процесі роботи виконано аналіз існуючих систем автоматизації мікроклімату. Описано всі компоненти розробленого програмного забезпечення. Реалізовано оптимізаційний ПІД-алгоритм керування та цифровий двійник теплового середовища.

Розроблено зручний веб-інтерфейс з відображенням даних у реальному часі через SignalR. Реалізовано систему авторизації на основі JWT-токенів та генерацію звітів у форматах PDF і CSV.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11 або Linux з встановленим .NET 8 Runtime.

Програму розроблено мовами програмування C#, C++ та JavaScript з використанням платформи ASP.NET Core 8 та середовище Arduino IDE.

Ключові слова: комп'ютерна інженерія, мікроклімат, ПІД-регулятор, ESP32, ASP.NET Core, IoT, SignalR.

ABSTRACT

Kokhan I.O. Software for an Intelligent Microclimate Control System for Industrial Premises Based on Sensor Data Analysis and Optimization Algorithms. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

This bachelor's thesis developed software for an intelligent automatic microclimate control system for warehouse and industrial premises.

The aim of the development is software for automatic control and regulation of air temperature and relative humidity based on sensor data and a PID control algorithm.

The result of the work is a software implementation of a three-tier system: embedded firmware for the ESP32 microcontroller with a DHT22 sensor, a server-side application on the ASP.NET Core 8 platform with an SQLite database, and an operator web interface.

In the course of the work, an analysis of existing microclimate automation systems was performed. All components of the developed software are described. A PID optimization control algorithm and a digital twin of the thermal environment have been implemented.

A user-friendly web interface with real-time data display via SignalR has been developed. A JWT-based authorization system and report generation in PDF and CSV formats have been implemented.

The software can be used on a PC running Windows 10/11 or Linux with .NET 8 Runtime installed.

The software was developed using the C#, C++, and JavaScript programming languages with the ASP.NET Core 8 platform and the Arduino IDE development environment.

Keywords: computer engineering, microclimate, PID controller, ESP32, ASP.NET Core, IoT, SignalR.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	2
ВСТУП	3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	5
1.1 Призначення системи	5
1.2 Область застосування	7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень	9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування.....	15
2.3 Розгорнута постановка завдання	19
3 ОПИС І ОБґРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	21
3.1 Опис функціонування системи	21
3.2 Розробка структурної схеми	24
3.3 Розробка функціональної схеми	28
3.4 Розробка діаграми процесів	31
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ	35
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	35
4.2 Захист розробленого програмного забезпечення.....	42
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	44
6 ОСНОВНІ ВИСНОВКИ	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	60

					<i>ВКРБ-122.26.0017.00.00.ПЗ</i>		
Змн.	Арк.	№ докум.	Підпис	Дата	Програмне забезпечення системи інтелектуального керування мікрокліматом промислових приміщень на основі аналізу сенсорних даних та оптимізаційних алгоритмів		
Розроб.		Кохан І.О.					
Перевір.		Минайленко Р.М					
Н. Контр.		Коваленко А.С.					
Затверд.		Смірнов О.А					
					Літ.	Арк.	Акрушів
						1	63
					<i>ЦНТУ КН-22</i>		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АСК - автоматизована система керування

БД - база даних

ПІД - пропорційно-інтегрально-диференціальний (регулятор)

ПЗ - програмне забезпечення

API - Application Programming Interface

CPU - Central Processing Unit

DHT22 - Digital Humidity and Temperature sensor, model 22

EF Core - Entity Framework Core

ESP32 - мікроконтролер серії Espressif Systems 32-bit

GPIO - General Purpose Input/Output

HTTP - HyperText Transfer Protocol

IoT - Internet of Things

JSON - JavaScript Object Notation

JWT - JSON Web Token

MCU - Microcontroller Unit

MQTT - Message Queuing Telemetry Transport

PID - Proportional-Integral-Derivative (controller)

REST - Representational State Transfer

SCADA - Supervisory Control and Data Acquisition

SDK - Software Development Kit

SignalR - бібліотека реального часу для ASP.NET

SQLite - компактна вбудована реляційна СУБД

Wi-Fi - Wireless Fidelity

WEB API - програмний інтерфейс на основі HTTP-протоколу

					<i>ВКРБ-122.26.0017.00.00.ПЗ</i>	Арк.
						2
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Сучасне промислове виробництво висуває підвищені вимоги до ефективності управління мікрокліматом у складських і виробничих приміщеннях. Відхилення температури та відносної вологості від допустимих норм призводить до псування сировини та готової продукції, прискореного зносу обладнання, а також до збільшення енергетичних витрат. За даними Міжнародної організації стандартизації (ISO), некоректне зберігання товарів у ненормованому мікрокліматі спричиняє збитки, що сягають 15-20 % від вартості продукції на рік [1]. Отже, автоматизація контролю та регулювання параметрів мікроклімату є критично важливою задачею для підприємств будь-якої галузі.

Цифровізація промисловості в рамках концепції Industry 4.0 та розвиток технологій IoT (Internet of Things) відкривають нові можливості для побудови інтелектуальних систем керування, що здатні в режимі реального часу збирати дані з сенсорів, аналізувати їх та формувати керуючі впливи на виконавчі пристрої. Застосування мікроконтролерних платформ, таких як ESP32, у поєднанні з хмарними або локальними серверними рішеннями дозволяє суттєво знизити вартість впровадження систем автоматизації порівняно з промисловими SCADA-рішеннями від таких виробників, як Siemens або Honeywell [2].

Незважаючи на широке розповсюдження готових рішень для автоматизації мікроклімату, більшість із них є або надмірно складними та дорогими для середніх підприємств, або ж спрощеними і не передбачають адаптивного управління з використанням оптимізаційних алгоритмів. Розробка гнучкого, масштабованого програмного забезпечення на базі відкритих технологій залишається актуальною задачею.

Мета й завдання дослідження. Метою роботи є розробка програмного забезпечення системи інтелектуального керування мікрокліматом промислових приміщень на основі аналізу сенсорних даних та оптимізаційних алгоритмів.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

Для досягнення поставленої мети визначена наступна програма дослідження:

- огляд та аналіз існуючих систем автоматизованого керування мікрокліматом, визначення їх переваг і недоліків;
- розробка вбудованого програмного забезпечення (firmware) для мікроконтролера ESP32 з датчиком DHT22 для збору телеметричних даних;
- проектування та реалізація серверної частини (backend) на базі ASP.NET Core Web API з використанням Entity Framework Core та SQLite;
- реалізація оптимізаційного ПІД-алгоритму керування та цифрового двійника середовища у вигляді фонових сервісів ASP.NET;
- розробка веб-інтерфейсу оператора з візуалізацією даних у реальному часі засобами SignalR та Chart.js;
- тестування системи та оцінка ефективності розроблених алгоритмів управління.

Практична цінність отриманих результатів полягає в тому, що розроблене програмне забезпечення може бути безпосередньо застосоване на складських підприємствах для автоматичного підтримання нормованих параметрів мікроклімату. Система забезпечує скорочення енерговитрат за рахунок інтелектуальних профілів режимів роботи та дозволяє персоналу дистанційно контролювати стан приміщення через веб-браузер без встановлення додаткового програмного забезпечення. Реалізований модуль генерації звітів у форматах PDF та CSV надає керівництву підприємства аналітичні дані для прийняття управлінських рішень.

Таким чином, виходячи з вищенаведеного, розробка програмного забезпечення системи інтелектуального керування мікрокліматом промислових приміщень є актуальною задачею, що потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Розроблена система призначена для автоматизованого контролю та інтелектуального регулювання параметрів мікроклімату - температури та відносної вологості повітря - у промислових і складських приміщеннях.

Система здійснює безперервний моніторинг мікроклімату за допомогою апаратного вузла на базі мікроконтролера ESP32 з підключеним цифровим датчиком DHT22, передає виміряні дані через Wi-Fi мережу на сервер, де вони обробляються оптимізаційними алгоритмами, та формує керуючі сигнали для виконавчих пристроїв (нагрівач, вентилятор, зволожувач).

Основними функціями системи є:

- безперервний збір телеметричних даних температури та вологості з дискретністю 5 секунд;
- зберігання хронологічної бази телеметрії у локальній базі даних SQLite з можливістю подальшого аналізу;
- автоматичне регулювання мікроклімату за ПІД-алгоритмом із можливістю налаштування коефіцієнтів K_p , K_i , K_d ;
- підтримка інтелектуальних профілів мікроклімату з розкладом ("Робочий", "Нічний економ", "Аварійний") із автоматичною зміною цільових значень setpoint;
- відображення поточних даних та стану виконавчих пристроїв на веб-інтерфейсі в режимі реального часу;
- розмежування доступу між ролями "Оператор" та "Інженер" за допомогою JWT-токенів;
- формування аналітичних звітів у форматах PDF та CSV за заданий часовий проміжок;
- сигналізація про аварійні ситуації (втрата зв'язку з датчиком, вихід за критичні межі параметрів) через веб-інтерфейс.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

Система функціонує в автоматичному режимі без постійної присутності обслуговуючого персоналу. Усі налаштування та моніторинг здійснюються через веб-браузер на будь-якому пристрої (ПК, планшет, смартфон) у локальній мережі підприємства. Технічну підтримку та зміну параметрів ПІД-регулятора і профілів виконує персонал з роллю "Інженер".

Система не вимагає встановлення спеціалізованого клієнтського ПЗ на робочих місцях операторів та є платформи-незалежною в частині веб-інтерфейсу. Серверна частина розгортається на будь-якій платформі, що підтримує середовище виконання .NET 8 (Windows, Linux, macOS).

Система складається з трьох взаємопов'язаних частин. Перша -апаратний вузол на базі мікроконтролера ESP32 з датчиком DHT22, прошивку якого написано мовою C++ у середовищі PlatformIO. Він виконує безпосереднє вимірювання фізичних параметрів середовища та надсилає їх на сервер через Wi-Fi. Друга -серверна частина на платформі ASP.NET Core Web API (.NET 8), яка є центром обробки: приймає телеметрію, запускає цикл ПІД-регулювання у фоновому сервісі BackgroundService, зберігає дані у базі SQLite та формує звіти. Третя -веб-інтерфейс на HTML/CSS/JavaScript з Bootstrap 5 і Chart.js, через який відбувається вся взаємодія з системою без потреби встановлювати будь-яке додаткове програмне забезпечення.

Датчик DHT22 забезпечує вимірювання температури в діапазоні від -40°C до $+80^{\circ}\text{C}$ з похибкою $\pm 0,5^{\circ}\text{C}$ та відносної вологості від 0 % до 100 % з похибкою $\pm 2-5\%$. Ці характеристики повністю задовольняють вимоги промислового моніторингу для більшості типів виробничих та складських об'єктів. Передача даних від датчика до мікроконтролера відбувається за однопровідним цифровим протоколом, що виключає аналого-цифрові похибки перетворення і спрощує апаратну реалізацію вузла.

Окремою перевагою системи є вбудований модуль цифрового двійника приміщення -клас EnvironmentSimulator. Він моделює теплову динаміку середовища залежно від потужності ввімкнених виконавчих пристроїв, що дозволяє повноцінно демонструвати і перевіряти роботу алгоритмів керування навіть за відсутності підключеного фізичного обладнання. Завдяки цьому розробка і тестування системи можуть вестися незалежно від наявності апаратного вузла.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

Функціонально систему можна розділити на три рівні: інформаційний, керувальний та аналітичний. Інформаційний рівень відповідає за відображення поточних показників температури і вологості, стану виконавчих пристроїв та активного профілю мікроклімату на дашборді. Керувальний рівень реалізує ПІД-алгоритм регулювання, перемикання між профілями за розкладом та формування керуючих сигналів. Аналітичний рівень забезпечує накопичення архіву вимірювань і генерацію звітів у форматах PDF і CSV для потреб керівництва підприємства.

1.2 Область застосування

Основною областю застосування системи є складські приміщення промислових підприємств, де зберігаються матеріали, сировина або готова продукція, якість яких залежить від параметрів навколишнього середовища. Нормативні вимоги до умов зберігання більшості промислових товарів, фармацевтичних препаратів, продуктів харчування, електронних компонентів та полімерних матеріалів регламентують допустимі діапазони температури та відносної вологості. Порушення цих діапазонів призводить до незворотного погіршення властивостей продукції та матеріальних збитків.

Розглянемо конкретний приклад застосування системи. На складі площею від 500 до 5000 м² для зберігання лакофарбових матеріалів нормативна температура зберігання становить від +5 °С до +30 °С, відносна вологість - не більше 75 %. Контроль цих параметрів вручну є трудомістким та ненадійним. Впровадження розробленої системи дозволяє:

- автоматично підтримувати задані параметри мікроклімату цілодобово без участі персоналу;
- вести журнал відхилень від норми з точним зазначенням часу та тривалості порушень;
- отримувати миттєві сповіщення при виникненні аварійних ситуацій;
- формувати звіти для служб якості та контролюючих органів;
- скоротити споживання електроенергії системами опалення та вентиляції за рахунок автоматичного перемикання у режим "Нічний економ" у неробочі години.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Крім складів, система може застосовуватись у таких сферах:

-виробничі цехи точного машинобудування та електронної промисловості, де мікроклімат безпосередньо впливає на якість виробу та роботу технологічного обладнання;

-агропромислові підприємства: зернові склади, овочесховища, теплиці - де підтримання вологості та температури є обов'язковою умовою збереження врожаю;

-серверні кімнати та технічні приміщення центрів обробки даних, де перегрів є основною причиною відмов обладнання;

-фармацевтичні склади та лабораторії, де умови зберігання препаратів суворо регламентовані стандартами GMP (Good Manufacturing Practice).

Система розрахована на обслуговування одного або декількох вимірювальних вузлів (ESP32 + DHT22), що можуть бути розміщені у різних

зонах одного приміщення або у декількох суміжних приміщеннях, підключених до однієї локальної мережі. Масштабування системи досягається додаванням нових апаратних вузлів без змін у серверному програмному забезпеченні.

Мінімальні технічні вимоги до інфраструктури для розгортання системи:

-наявність локальної Wi-Fi мережі (стандарт IEEE 802.11 b/g/n) у зоні встановлення датчиків;

-сервер або персональний комп'ютер з ОС Windows 10/11, Linux або macOS з встановленим середовищем .NET 8 Runtime;

-веб-браузер будь-якого виробника на робочому місці оператора (Google Chrome, Mozilla Firefox, Microsoft Edge);

-живлення 5V/USB або 5V DC для кожного апаратного вузла ESP32.

Подальший розвиток системи, на мою думку, може охопити значно ширший спектр застосувань. Наприклад, додавши підтримку кількох апаратних вузлів одночасно, систему можна перетворити на повноцінне рішення для великих складських комплексів або багатозонних виробництв. Також цікавим напрямком є інтеграція з зовнішніми системами автоматизації будівель через протоколи MQTT або Modbus, що відкриває можливість керування реальним промисловим обладнанням - кондиціонерами, вентиляційними установками або системами зрошення теплиць.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень

На сьогодні на ринку систем автоматизації мікроклімату представлений широкий спектр рішень - від великих промислових SCADA-платформ до DIY-проектів на основі мікроконтролерів з відкритим кодом. Кожен клас рішень має свою архітектуру, стек технологій та сферу застосування. Для обґрунтованого вибору технологій розробки власної системи необхідно провести детальний аналіз найбільш поширених аналогів та виявити їхні ключові переваги й недоліки.

Siemens SIMATIC S7 / WinCC

Siemens SIMATIC S7 - це лінійка програмованих логічних контролерів (ПЛК) виробництва компанії Siemens AG, яка є одним із світових лідерів у сфері промислової автоматизації. У поєднанні з SCADA-системою WinCC (Windows Control Center) дане рішення утворює повноцінний комплекс для моніторингу та управління технологічними процесами, у тому числі параметрами мікроклімату. Система широко застосовується на великих промислових підприємствах, у хімічній, металургійній, харчовій та енергетичній промисловості [3].

Архітектура рішення Siemens SIMATIC S7 / WinCC є класичною трирівневою: на польовому рівні розташовані датчики та виконавчі механізми, на рівні управління - ПЛК серій S7-300, S7-400 або S7-1500, на рівні диспетчеризації - SCADA WinCC, встановлена на промислових комп'ютерах під управлінням ОС Windows. Зв'язок між рівнями здійснюється по промислових мережах PROFIBUS або PROFINET. ПЛК програмуються мовами стандарту IEC 61131-3 (LAD, FBD, SCL, STL) за допомогою інтегрованого середовища розробки TIA Portal [4].

Основні переваги системи Siemens SIMATIC S7 / WinCC:

- висока надійність і відмовостійкість - компоненти розраховані на роботу в умовах підвищених температур, вібрацій та електромагнітних перешкод;

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

- розвинена екосистема компонентів і бібліотек для різних галузей промисловості;
- вбудовані апаратні ПД-регулятори з автоматичним налаштуванням (auto-tuning);
- сертифікація відповідно до міжнародних стандартів безпеки (SIL, IEC 62443);
- масштабованість від невеликих установок до розподілених систем із тисячами

точок введення-виведення.



Рисунок 2.1 - Контролер Siemens SIMATIC S7-1500

Суттєвими недоліками є:

- надзвичайно висока вартість обладнання і ліцензій на програмне забезпечення - повноцінне впровадження для одного складського приміщення потребує інвестицій від 50 000 доларів США і вище;
- необхідність залучення сертифікованих спеціалістів Siemens для проектування, монтажу та налагодження;
- замкнена архітектура та прив'язка до екосистеми одного виробника;

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		10

- надлишкова функціональність та складність для задачі моніторингу мікроклімату у невеликому складі.

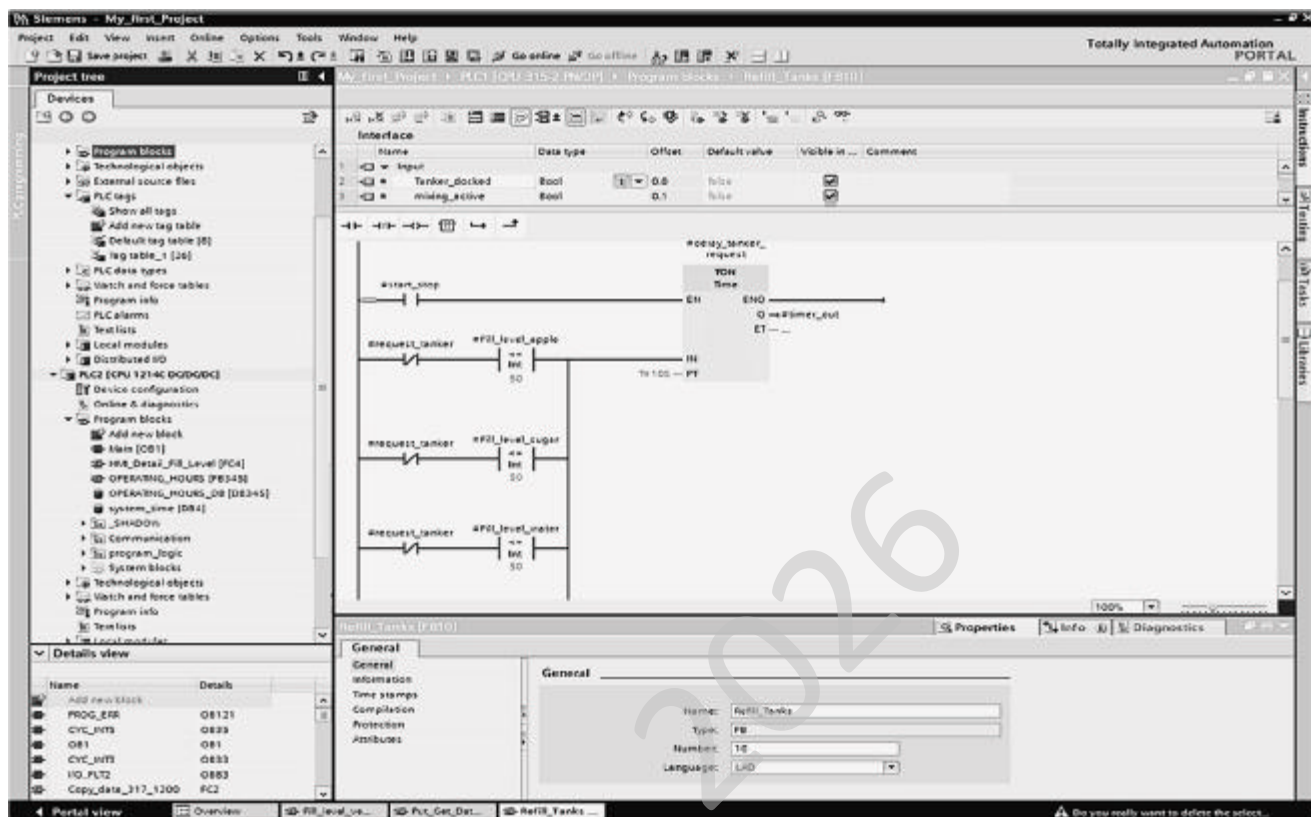


Рисунок 2.2 - Середовище розробки Siemens TIA Portal



Рисунок 2.3 - Інтерфейс SCADA-системи Siemens WinCC

Arduino-based DIY рішення (MQTT + Grafana)

З поширенням руху відкритого апаратного забезпечення та концепції IoT самостійно зібрані системи моніторингу мікроклімату на платформах Arduino та ESP32 стали доступною альтернативою комерційним рішенням. Типова архітектура будується навколо мікроконтролера з датчиками температури й вологості (DHT11, DHT22, SHT31), брокера повідомлень Mosquitto, часової бази даних InfluxDB та системи візуалізації Grafana [5].

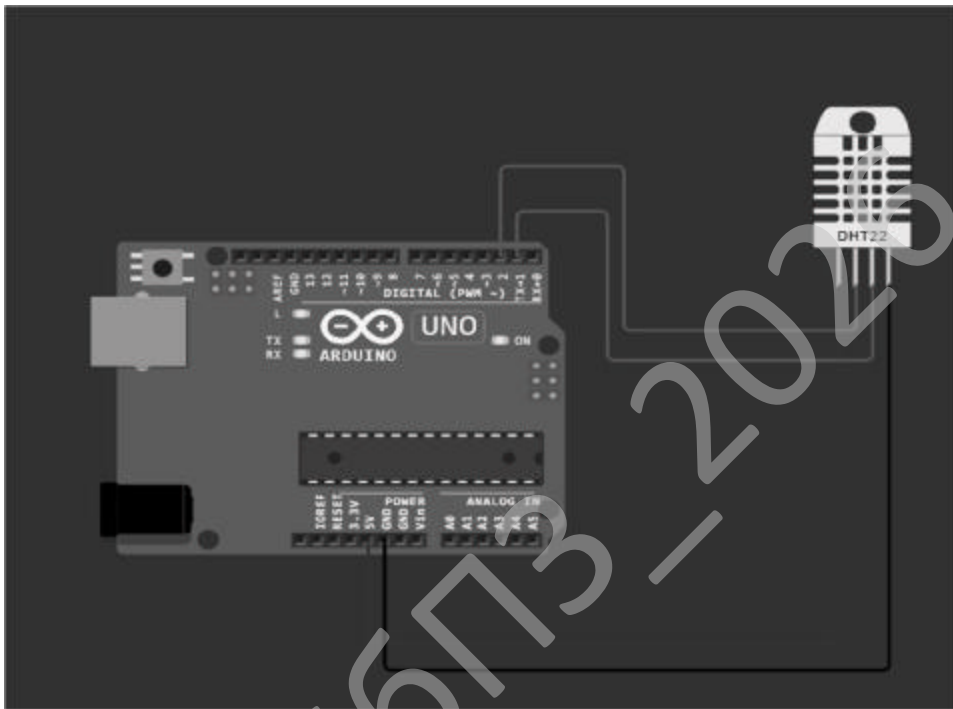


Рисунок 2.4.- Схема підключення датчика DHT22 до мікроконтролера Arduino

Протокол MQTT (Message Queuing Telemetry Transport) - легковаговий протокол обміну повідомленнями за моделлю публікація-підписка, розроблений під пристрої з обмеженими ресурсами та нестабільними каналами зв'язку [5]. Мікроконтролер публікує виміри у відповідний топік, брокер Mosquitto розподіляє їх між підписниками - сервером збору даних і системою візуалізації. Така схема добре масштабується горизонтально: кількість вузлів збільшується без зміни серверної частини.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

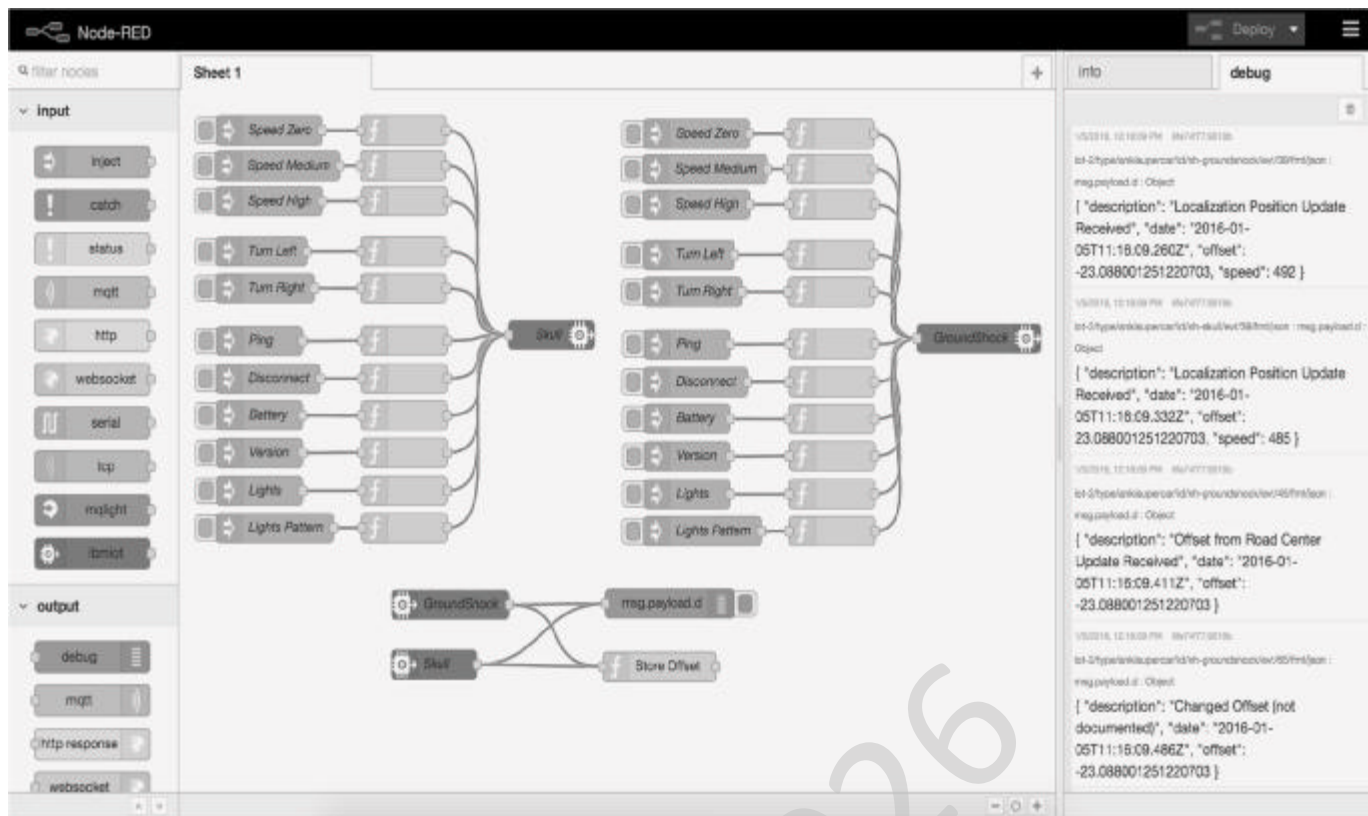


Рисунок 2.5 - Редактор потоків Node-RED з MQTT-вузлами

До практичних переваг стеку Arduino/ESP32 + MQTT + Grafana належать низька вартість апаратури - ESP32 і датчик DHT22 разом коштують менше 10 доларів США - і повністю відкритий код усіх компонентів: бібліотек Arduino, Mosquitto, InfluxDB і Grafana. Додатково варто відзначити широкий вибір датчиків і протоколів зв'язку та розвинену спільноту з великою кількістю готових прикладів реалізації.

Стек Arduino/ESP32 + MQTT + Grafana має кілька вузьких місць. Права доступу здебільшого або не реалізовані взагалі, або зведені до одного HTTP-пароля на весь інтерфейс. Регулятор із ПД-законом керування в жодному з компонентів не передбачений - розробник або пише його самостійно, або підключає зовнішню бібліотеку, що додає нові залежності до проєкту. Mosquitto, InfluxDB, Grafana і Node-RED - чотири незалежні програми від різних авторів, і первинне налаштування їх спільної роботи займає суттєвий час; при цьому оновлення навіть одного пакету нерідко порушує сумісність із рештою. Вивантаження даних у PDF чи CSV жодна з цих програм не підтримує без додаткових плагінів. Нарешті, більшість таких збірок існують без схем і технічної

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

документації, що ускладнює пошук несправностей і робить цілодобову роботу системи на виробничому об'єкті ненадійною.



Рисунок 2.6 - Дашборд Grafana для моніторингу температури та вологості

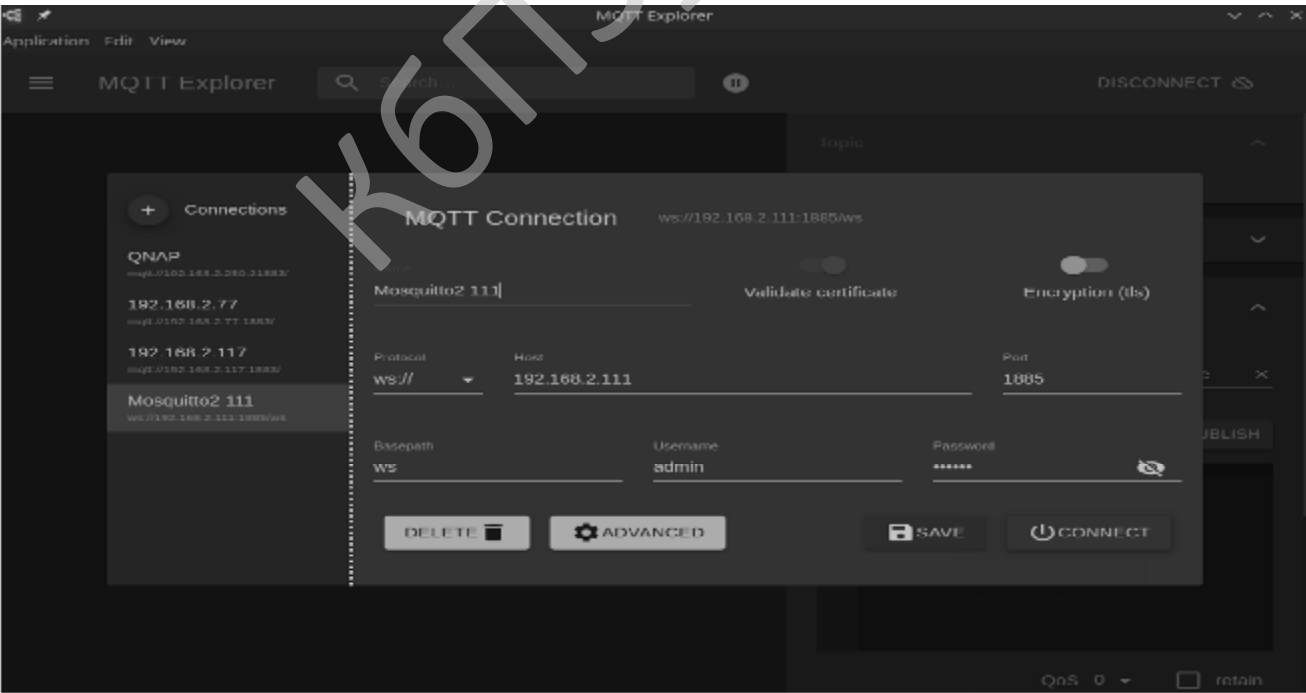


Рисунок 2.7 - Інтерфейс MQTT Explorer для підключення до брокера Mosquitto

2.2 Обґрунтування вибору засобів для побудови системи

На основі проведеного аналізу аналогів сформульовано вимоги до власної розробки та обрано технологічний стек, що відповідає цим вимогам. Порівняльну характеристику розглянутих систем та розробленого рішення наведено у таблиці 2.1.

Таблиця 2.1 - Порівняльна характеристика систем керування мікрокліматом

Характеристика	Siemens SIMATIC S7 / WinCC	Arduino DIY (MQTT + Grafana)	Розроблена система
Тип рішення	Промислова SCADA	Відкрите IoT- рішення	Локальний Web API
Мова програмування	SCL, LAD (PLC)	C++, Python, Node.js	C#, C++ (ESP32)
Протокол передачі	PROFIBUS, PROFINET	MQTT / HTTP	HTTP / SignalR
СУБД	MS SQL / Oracle	InfluxDB / MySQL	SQLite
Алгоритм управління	ПІД (апаратний)	ПІД (опційно)	ПІД (програмний)
Веб-інтерфейс	WinCC WebNavigator	Grafana Dashboard	Власна HTML/JS панель
Авторизація	SCADA-права	Базова / відсутня	JWT, ролі Operator/Engineer
Звіти PDF/CSV	Так (ліцензія)	Grafana (CSV)	Так (QuestPDF + CSV)

Продовження таблиці 2.1

Характеристика	Siemens SIMATIC S7 / WinCC	Arduino DIY (MQTT + Grafana)	Розроблена система
Апаратна платформа	Siemens S7- 300/400	Arduino / ESP32	ESP32 + DHT22
Вартість впровадження	Висока (>50 000 \$)	Низька (<200 \$)	Низька (<200 \$)
Масштабованість	Висока	Середня	Середня
Відкритість коду	Закрита	Відкрита	Відкрита

Як видно з таблиці 2.1, рішення Siemens SIMATIC S7 / WinCC забезпечує найвищу надійність та функціональність, але є економічно недоступним для малих і середніх підприємств. DIY-рішення на базі Arduino мають низьку вартість, але не забезпечують необхідного рівня безпеки, структурованості та готовності до промислового застосування. Розроблена система поєднує переваги обох підходів: низьку вартість апаратної частини та відкритість DIY-рішень з архітектурною зрілістю та функціональністю, притаманною промисловим продуктам.

Обґрунтування вибору конкретних технологій та компонентів стеку розробки наведено нижче.

Мікроконтролер ESP32 та датчик DHT22

Для апаратного вузла збору телеметрії обрано мікроконтролер ESP32 виробництва Espressif Systems. ESP32 є двоядерним 32-бітним мікроконтролером на базі архітектури Xtensa LX6 з тактовою частотою до 240 МГц, що поєднує вбудовані модулі Wi-Fi (IEEE 802.11 b/g/n) та Bluetooth, достатній обсяг Flash-пам'яті (4 МБ) та оперативної пам'яті (520 КБ SRAM). ESP32 підтримується Arduino IDE та бібліотеками Arduino-фреймворку, що значно спрощує розробку прошивки.

Вартість модуля складає менше 5 доларів США, що робить його оптимальним вибором для систем IoT-моніторингу [6].

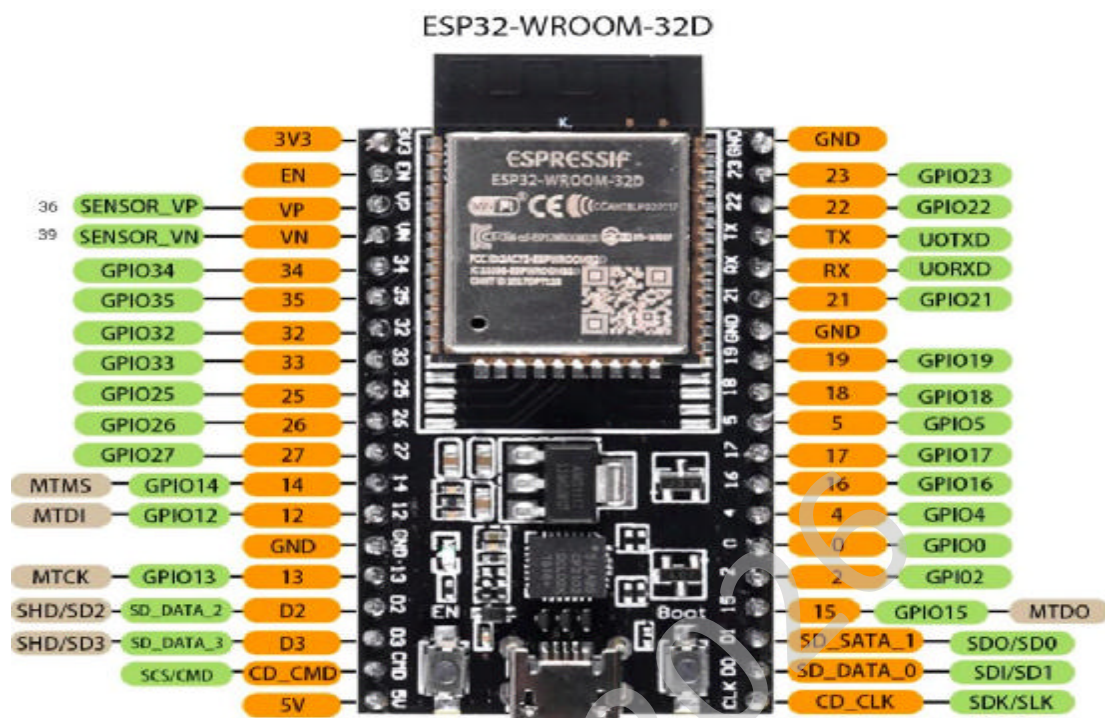


Рисунок 2.9 - Мікроконтролер ESP32-WROOM-32D та його розпіновка

Датчик DHT22 (AM2302) є цифровим давачем температури та відносної вологості з однопровідним послідовним інтерфейсом. Технічні характеристики DHT22: діапазон вимірювання температури - від мінус 40 °C до плюс 80 °C, точність $\pm 0,5$ °C; діапазон вимірювання вологості - від 0 до 100 %, точність $\pm 2-5$ %. Датчик підключається до одного цифрового GPIO-піна ESP32 та зчитується за допомогою бібліотеки DHT.h від Adafruit. Оновлення даних відбувається не частіше ніж раз на 2 секунди [7].

Серверна платформа ASP.NET Core та мова C#

Backend написаний на C# під ASP.NET Core 8 - фреймворку від Microsoft із відкритим кодом, що працює на Windows, Linux і macOS з одним і тим самим скомпільованим артефактом.

У бенчмарках TechEmpower Round 22 [8] платформа обходить Node.js/Express, Python/FastAPI та Java/Spring Boot за пропускнуою здатністю на ідентичному стенді.

DI-контейнер і IHostedService вбудовані - сторонні рішення для керування залежностями чи фоновими задачами підключати не треба.

Вибір платформи для цього проєкту обумовлений конкретними потребами реалізації. SignalR входить до складу фреймворку і самостійно обирає транспорт - WebSocket, Server-Sent Events або Long Polling - залежно від мережевого середовища клієнта. EF Core замінює ручний SQL: схема таблиць задається C#-класами, вибірки будуються через LINQ, міграції генеруються командою dotnet ef. JWT із ролевою моделлю налаштовується в межах стандартного middleware-ланцюжка без додаткових пакетів. BackgroundService дає змогу тримати ПІД-регулятор і симулятор як окремі керовані потоки всередині одного процесу хоста.

База даних SQLite та Entity Framework Core

Для зберігання телеметричних даних, профілів мікроклімату та журналу системних подій обрано вбудовану реляційну СУБД SQLite. SQLite є безсерверною, файловою базою даних, що не потребує окремого процесу сервера баз даних. Вся база даних зберігається у єдиному файлі на диску сервера, що спрощує розгортання та резервне копіювання. Для застосунків із помірним навантаженням (десятки запитів на секунду) SQLite забезпечує достатню продуктивність [9].

Взаємодія з SQLite реалізована через Entity Framework Core (EF Core) - сучасний об'єктно-реляційний маппер (ORM) для платформи .NET. EF Core дозволяє описувати структуру бази даних у вигляді класів C# (Entity Models), а міграції автоматично генерують та застосовують зміни схеми БД. Це забезпечує незалежність від конкретної СУБД: за потреби SQLite може бути замінено на PostgreSQL або MS SQL Server без змін у коді застосунку, лише шляхом зміни рядка підключення та провайдера EF Core. У розробленій системі EF Core використовується для роботи з чотирма основними сутностями: TelemetryRecord, User, Profile та SystemLog.

SignalR для передачі даних у реальному часі

Для забезпечення відображення телеметричних даних у веб-інтерфейсі оператора в режимі реального часу без перезавантаження сторінки обрано бібліотеку SignalR. SignalR - це бібліотека з відкритим кодом для платформи ASP.NET, яка спрощує додавання функціональності реального часу до веб-застосунків.

					<i>ВКРБ-122.26.0017.00.00.ПЗ</i>	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

Бібліотека автоматично обирає оптимальний транспортний механізм: WebSocket (пріоритетний), Server-Sent Events або Long Polling - залежно від підтримки браузером та мережевою інфраструктурою [10].

Фоновий сервіс ClimateControlService щосекунди обчислює керуючі впливи ПІД-алгоритму та надсилає оновлені дані через SignalR-хаб усім підключеним клієнтам (веб-браузерам операторів). Клієнтська JavaScript-бібліотека signalr.js, підключена до веб-інтерфейсу, отримує ці дані та оновлює графіки Chart.js без будь-яких додаткових HTTP-запитів з боку клієнта. Такий підхід суттєво знижує навантаження на сервер порівняно з традиційним polling-механізмом та забезпечує миттєву доставку аварійних сповіщень усім підключеним операторам одночасно.

Веб-інтерфейс: Bootstrap, Chart.js

Фронтенд-частина системи реалізована у вигляді статичних HTML/CSS/

JavaScript файлів, що роздаються серверами ASP.NET Core як статичний контент. Для адаптивної верстки та готових UI-компонентів використовується CSS-фреймворк Bootstrap 5. Bootstrap забезпечує коректне відображення інтерфейсу оператора на пристроях з різними розмірами екрану (ПК, планшет, смартфон) через систему сітки (Grid System) та адаптивні компоненти.

Для візуалізації даних телеметрії у вигляді динамічних графіків застосовано бібліотеку Chart.js - легковаговий інструмент побудови інтерактивних графіків на основі HTML5 Canvas. У системі реалізовано два типи графіків: лінійний графік поточної температури та вологості у реальному часі та графік потужності виконавчих пристроїв (нагрівач, охолоджувач) у діапазоні від 0 до 100 %. Обидва графіки оновлюються в режимі реального часу через SignalR.

2.3 Розгорнута постановка завдання

На підставі аналізу існуючих аналогів та вимог до системи сформульовано розгорнуту постановку завдання на розробку програмного забезпечення системи інтелектуального керування мікрокліматом промислових приміщень на основі аналізу сенсорних даних та оптимізаційних алгоритмів.

					<i>ВКРБ-122.26.0017.00.00.ПЗ</i>	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

Предмет розробки - програмне забезпечення, що охоплює три рівні: вбудоване програмне забезпечення мікроконтролерного вузла, серверну частину та веб-інтерфейс користувача.

В процесі виконання випускної кваліфікаційної роботи необхідно виконати наступний обсяг роботи:

- визначити призначення системи та область її застосування;
- провести аналіз існуючих систем-аналогів, виявити їх переваги та недоліки і врахувати результати аналізу при розробці власного рішення;
- обґрунтувати вибір засобів розробки: мікроконтролера ESP32, датчика DHT22, платформи ASP.NET Core, СУБД SQLite та бібліотек SignalR і QuestPDF;
- розробити програмне забезпечення системи інтелектуального керування мікрокліматом із реалізацією ПД-алгоритму, модуля симуляції та веб-інтерфейсу;
- протестувати систему за допомогою модульних тестів xUnit для виявлення та усунення можливих помилок;
- оформити пояснювальну записку відповідно до вимог стандартів, включаючи перелік умовних позначень, список літератури та анотацію українською і англійською мовами.

Таким чином, задача розробки є комплексною інженерною задачею, що охоплює програмування мікроконтролера, проєктування серверної архітектури, реалізацію математичного алгоритму автоматичного регулювання та розробку веб-інтерфейсу. Вирішення даної задачі повністю висвітлюється в наступних розділах пояснювальної записки.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розроблена система інтелектуального керування мікрокліматом є трирівневою програмно-апаратною системою, що охоплює рівень збору фізичних даних, рівень обробки та управління і рівень відображення та взаємодії з користувачем. Взаємодія між рівнями здійснюється через стандартні мережеві протоколи: HTTP/REST та WebSocket (SignalR).

На першому рівні розташований апаратний вузол на базі мікроконтролера ESP32 з підключеним цифровим датчиком температури та відносної вологості DHT22. Прошивка (firmware), написана мовою C++ у середовищі Arduino IDE, виконує циклічне зчитування показань датчика кожні 5 секунд, формує JSON-повідомлення і відправляє його на сервер через протокол HTTP POST по локальній Wi-Fi мережі. У разі тимчасової втрати з'єднання прошивка автоматично виконує повторне підключення до точки доступу.

На другому рівні функціонує серверна частина (backend), реалізована як Web API на платформі ASP.NET Core 8. Сервер виконує такі функції: прийом та збереження телеметричних даних у базі даних SQLite, автентифікацію користувачів із видачею JWT-токенів, надання REST-ендпоінтів для отримання даних і керування налаштуваннями, виконання фонових сервісів ClimateControlService, а також роздачу статичних файлів веб-інтерфейсу. Фоновий сервіс щосекунди зчитує останню телеметрію, передає температуру в ПІД-регулятор, оновлює стан цифрового симулятора середовища та транслює результати підключеним клієнтам через SignalR-хаб.

На третьому рівні знаходиться веб-інтерфейс оператора, реалізований на HTML5/CSS3/JavaScript з використанням Bootstrap 5 та Chart.js. Інтерфейс надає дві ролі: "Оператор" може переглядати поточну телеметрію, графіки та сповіщення; "Інженер" додатково має доступ до панелі налаштування коефіцієнтів ПІД-регулятора та управління профілями мікроклімату. Автентифікація виконується через форму входу з подальшим збереженням JWT-токена у localStorage браузера.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

Система передбачає два режими функціонування. Перший - штатний, при якому апаратний вузол ESP32 фізично підключений до локальної мережі та надсилає реальні показання датчика DHT22 на сервер. Другий - демонстраційний (режим симуляції), при якому сервер автоматично активує модуль EnvironmentSimulator у разі відсутності даних від апаратного вузла протягом більше 15 секунд. Обидва режими функціонування системи реалізують однакову логіку ПІД-регулювання та відображення даних у веб-інтерфейсі - оператор бачить однаковий дашборд незалежно від того, чи дані надходять від реального датчика, чи генеруються симулятором. Це дуже зручно при налагодженні та демонстрації системи без підключення фізичного обладнання.

Подібна архітектура є характерною для сучасних IoT-систем промислового класу. ESP32 широко застосовується для побудови вузлів збору даних, де мікроконтролер виступає як HTTP-клієнт, а серверна частина - як центр обробки. Такий підхід дозволяє відокремити логіку збору даних від логіки керування, що суттєво спрощує тестування та подальше масштабування системи шляхом додавання нових апаратних вузлів.

Функціонально система реалізує три групи функцій: інформаційну, керувальну та аналітичну. Інформаційна функція забезпечує відображення поточної температури та вологості, стану виконавчих пристроїв (нагрівач, вентилятор, зволожувач), активного профілю мікроклімату та системних сповіщень. Керувальна функція відповідає за роботу ПІД-регулятора, перемикання між профілями мікроклімату за розкладом та зміну параметрів системи через веб-інтерфейс. Аналітична функція забезпечує накопичення архіву телеметрії та формування звітів у форматах PDF і CSV за вказаний часовий діапазон.

Авторизація є важливим елементом безпеки системи. Вона реалізована на основі JWT-токенів: користувач вводить логін та пароль у формі входу веб-інтерфейсу, після чого сервер перевіряє облікові дані та повертає підписаний токен. У разі невідповідності даних сервер повертає код HTTP 401, а інтерфейс відображає повідомлення про помилку. Токен зберігається у localStorage браузера та автоматично додається до заголовків усіх наступних запитів. Система розмежовує доступ між двома ролями: "Оператор" має право лише переглядати дані, тоді як "Інженер" може змінювати коефіцієнти ПІД-регулятора, створювати нові профілі мікроклімату та переглядати повний журнал системних подій.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		22

Загальний алгоритм функціонування системи в штатному режимі виглядає наступним чином. ESP32 зчитує показання DHT22 і відправляє їх на сервер. Контролер TelemetryController записує отримані дані у таблицю TelemetryRecords з поточним UTC-часом. Фоновий сервіс ClimateControlService вибирає останній запис телеметрії та перевіряє час його надходження: якщо затримка перевищує 15 секунд, система переходить у режим симуляції та формує попередження у журналі SystemLogs. ПІД-регулятор обчислює керуючий вплив $U(t)$ за формулою:

$$U(t) = K_p \times e(t) + K_i \times \int e(\tau) d\tau + K_d \times de(t)/dt, \quad (3.1)$$

де $e(t) = T_{set} - T(t)$ - відхилення поточної температури від заданого значення T_{set} ;

K_p , K_i , K_d - пропорційний, інтегральний та диференціальний коефіцієнти регулятора відповідно.

Обчислений вихід $U(t)$ передається до симулятора EnvironmentSimulator, який моделює теплові процеси у приміщенні: при позитивному значенні $U(t)$ активується нагрівач, при від'ємному - охолоджувач. Оновлені значення розрахункової температури, вологості та потужності виконавчих пристроїв надсилаються через SignalR-хаб усім підключеним браузерам та зберігаються у журналі SystemLogs.

Цикл взаємодії системи в штатному режимі виглядає наступним чином:

- запуск серверної частини та ініціалізація фонового сервісу ClimateControlService;
- підключення апаратного вузла ESP32 до локальної Wi-Fi мережі та початок передачі телеметрії;
- авторизація оператора або інженера через веб-інтерфейс та отримання JWT-токена;
- відображення поточних даних телеметрії та стану виконавчих пристроїв на дашборді в режимі реального часу через SignalR;
- автоматичне перемикання профілю мікроклімату за розкладом та коригування setpoint ПІД-регулятора;

- формування керуючих сигналів та оновлення стану симулятора щосекунди;
- формування звітів PDF або CSV на запит користувача з роллю "Інженер".

Обрана архітектура системи є модульною: логіка апаратного вузла, серверна частина та веб-інтерфейс розділені на незалежні компоненти у папках /hardware, /backend та /frontend репозиторію. Такий підхід дозволяє незалежно розробляти та тестувати кожен компонент, не порушуючи роботу всієї системи. Серверна частина відповідає за збереження стану системи, виконання алгоритму регулювання та надання API, тоді як веб-інтерфейс відповідає виключно за відображення даних та взаємодію з користувачем.

Модель взаємодії побудована за принципом клієнт-сервер. У ролі клієнтів виступають як апаратний вузол ESP32 (надсилає телеметрію), так і веб-браузер оператора (отримує дані та надсилає команди керування). У ролі сервера виступає ASP.NET Core Web API. Такий підхід є типовим для промислових IoT-систем, де центральний сервер агрегує дані від кількох вузлів та надає єдиний інтерфейс керування.

3.2 Розробка структурної схеми

Структурна схема системи відображає склад основних функціональних блоків та зв'язки між ними на рівні компонентів без деталізації внутрішньої логіки кожного блоку. Структурну схему системи наведено на рисунку 3.1.

Система складається з чотирьох основних структурних блоків: апаратного вузла збору даних, серверного ядра, бази даних та клієнтського веб-інтерфейсу.

Апаратний вузол включає датчик DHT22, мікроконтролер ESP32 та модуль Wi-Fi (вбудований в ESP32). Датчик DHT22 підключається до GPIO-піна 4 мікроконтролера за однопровідним протоколом і постачає цифрові значення температури та вологості. ESP32 обробляє отримані дані та через вбудований Wi-Fi-модуль передає їх на сервер.

Серверне ядро реалізоване як єдиний процес ASP.NET Core і містить кілька логічних підсистем: маршрутизатор HTTP-запитів (Middleware), контролери REST API (TelemetryController, AuthController, ProfileController, ReportController), підсистему авторизації JWT, SignalR-хаб для двостороннього зв'язку з браузерами, фоновий сервіс ClimateControlService, ПІД-регулятор та симулятор середовища EnvironmentSimulator.

					<i>ВКРБ-122.26.0017.00.00.ПЗ</i>	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

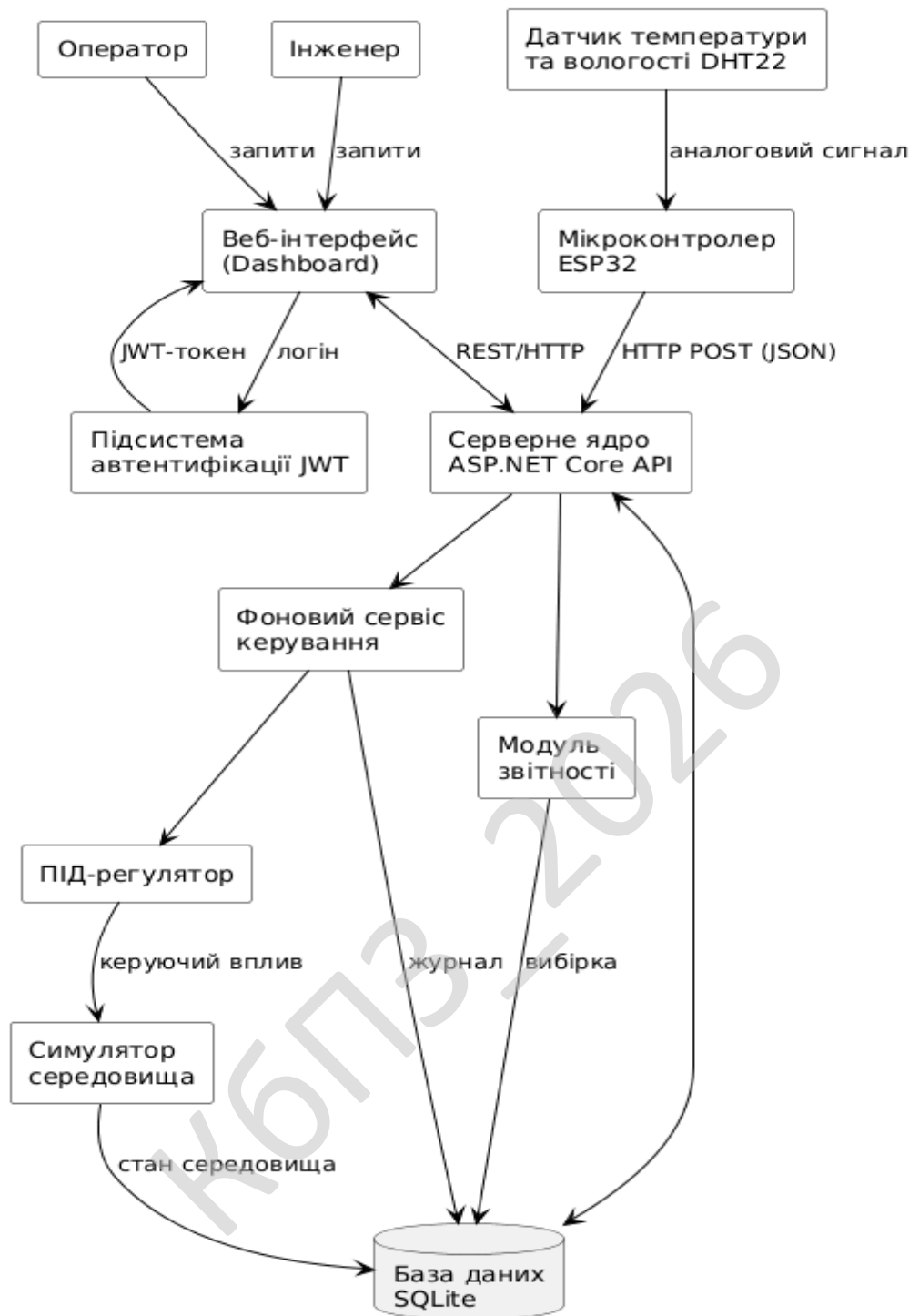


Рисунок 3.1 - Структурна схема системи

База даних SQLite містить чотири таблиці: TelemetryRecords (хронологія показань датчиків), Users (облікові записи з хешованими паролями та ролями), Profiles (налаштування профілів мікроклімату) та SystemLogs (журнал системних подій та аварійних повідомлень).

Клієнтський веб-інтерфейс - статичні файли index.html, style.css та app.js, що завантажуються з сервера і виконуються у браузері. Клієнт підтримує постійне SignalR-з'єднання для отримання даних у реальному часі та виконує REST API запити для авторизації та керування налаштуваннями.

Між структурними блоками системи існують два типи зв'язків. Суцільні стрілки позначають основні канали передачі даних: HTTP POST від апаратного вузла до серверного ядра, звернення контролерів до бази даних SQLite через Entity Framework Core та SignalR-з'єднання між серверним ядром і веб-інтерфейсом. Пунктирні стрілки позначають внутрішні виклики між підсистемами серверного ядра: фоновий сервіс ClimateControlService звертається до PidController та EnvironmentSimulator, які не мають прямого виходу назовні і є суто внутрішніми компонентами бізнес-логіки.

Перша підсистема - апаратний вузол ESP32 + DHT22 - є точкою входу фізичних даних у систему. Вона містить три блоки. Блок зчитування датчика виконує циклічне опитування DHT22 кожні 5 секунд за допомогою бібліотеки DHT.h від Adafruit та формує структуру даних з поточними значеннями температури і вологості. Блок Wi-Fi-з'єднання відповідає за підключення ESP32 до локальної мережі та автоматичне перепідключення при втраті сигналу. Блок HTTP-клієнта формує JSON-рядок вигляду {"temperature": 24.5, "humidity": 60.2} та виконує POST-запит на ендпоінт /api/telemetry серверного ядра. У разі недоступності сервера апаратний вузол продовжує роботу і повторює спроби відправки при відновленні з'єднання.

Друга підсистема - серверне ядро ASP.NET Core - є центральним вузлом обробки та містить сім логічних блоків. Middleware-маршрутизатор приймає вхідні HTTP-запити та направляє їх до відповідних контролерів залежно від шляху та методу запиту. TelemetryController обробляє POST-запити від апаратного вузла та зберігає дані у таблицю TelemetryRecords з поточним UTC-часовим штампом. AuthController реалізує ендпоінт /api/auth/login, перевіряє облікові дані користувача та повертає підписаний JWT-токен з вказаною роллю. ProfileController надає CRUD-операції для управління профілями мікроклімату, доступні лише користувачам з роллю "Інженер". ReportController формує файли PDF та CSV на основі вибірки з таблиці TelemetryRecords за вказаний часовий діапазон.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

ClimateControlService є фоновим сервісом, що запускається автоматично при старті застосунку і щосекунди виконує цикл регулювання. SignalR-хаб TelemetryHub забезпечує постійне двостороннє з'єднання з браузерами та транслює оновлені дані всім підключеним клієнтам одночасно.

Третя підсистема - модуль цифрового двійника EnvironmentSimulator - є важливим елементом архітектури, що відрізняє розроблену систему від простих систем моніторингу. Симулятор активується автоматично при відсутності реальних даних від апаратного вузла протягом більше 15 секунд. Він моделює теплову динаміку приміщення: при позитивному керуючому впливі $U(t)$ температура розрахункового середовища зростає, при від'ємному - знижується. Завдяки цьому веб-інтерфейс продовжує відображати актуальні дані навіть без підключеного фізичного датчика, що є зручним при налагодженні системи та демонстрації її роботи.

Четверта підсистема - клієнтський веб-інтерфейс - є рівнем відображення та взаємодії з оператором. Вона містить три блоки. Модуль авторизації реалізує форму входу та зберігає JWT-токен у localStorage браузера, автоматично додаючи його до заголовків усіх наступних запитів. Модуль дашборду відображає поточні значення температури та вологості, стан виконавчих пристроїв та активний профіль мікроклімату, оновлюючи дані в реальному часі через SignalR-з'єднання. Модуль інженерної панелі доступний лише користувачам з роллю "Інженер" і надає форми для зміни коефіцієнтів K_p , K_i , K_d ПІД-регулятора, створення нових профілів мікроклімату та перегляду журналу системних подій.

Компонент ClimateControlService на структурній схемі винесено окремо від контролерів, оскільки він є єдиним джерелом активної дії в системі. Всі контролери лише реагують на зовнішні запити, тоді як ClimateControlService самостійно ініціює цикли обчислення та розсилки даних. Саме через нього виконуються виклики `PidController.Compute()`, `EnvironmentSimulator.Update()` та `TelemetryHub.SendUpdate()`. Така централізація спрощує розуміння потоку даних: вся активна логіка зосереджена в одному місці.

Структурна схема демонструє, що система не є монолітною програмою, а складається з кількох чітко розмежованих рівнів.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

Перший рівень - фізичний, представлений апаратним вузлом ESP32 + DHT22, що збирає реальні показання середовища. Другий рівень - серверний, що агрегує дані, виконує алгоритм регулювання та зберігає хронологію подій. Третій рівень - клієнтський, що відображає стан системи та надає інтерфейс керування оператору і інженеру. Такий розподіл відповідає класичній трирівневій архітектурі IoT-систем і забезпечує незалежність кожного рівня від деталей реалізації інших.

3.3 Розробка функціональної схеми

Функціональна схема відображає деталізований опис взаємодії між програмними модулями системи у динаміці - яким чином інформація про стан мікроклімату трансформується з фізичного сигналу датчика до керуючого впливу на виконавчі пристрої та відображення на екрані оператора. Функціональну схему наведено на рисунку 3.2.

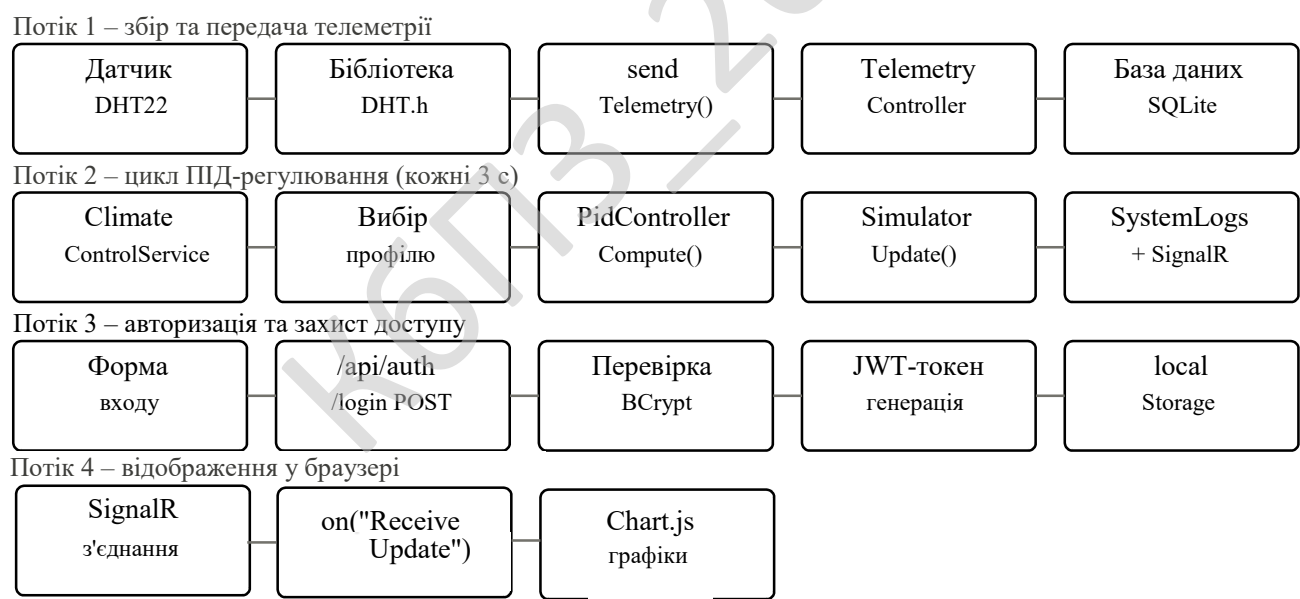


Рисунок 3.2 - Функціональна схема роботи системи

Функціонування системи на рівні модулів описується чотирма основними інформаційними потоками.

Перший функціональний блок - ініціалізація та запуск. При запуску серверної частини виконуються такі дії: реєстрація всіх сервісів у контейнері залежностей ASP.NET Core, застосування міграцій Entity Framework Core до бази даних SQLite, запуск фонових сервісів ClimateControlService у окремому потоці, ініціалізація SignalR-хабу TelemetryHub та очікування вхідних з'єднань. Паралельно апаратний вузол ESP32 виконує підключення до Wi-Fi-мережі, ініціалізацію датчика DHT22 та перехід у режим циклічної відправки телеметрії. Якщо Wi-Fi-з'єднання не вдалося встановити, ESP32 продовжує спроби підключення з інтервалом 5 секунд без переривання основного циклу роботи.

Потік 1 - збір та передача телеметрії. Датчик DHT22 генерує цифровий сигнал, який бібліотека DHT.h декодує у значення float temperature та float humidity. Функція sendTelemetry() формує JSON-рядок вигляду {"temperature": 23.5, "humidity": 45.2, "deviceId": "ESP32_01"} та виконує HTTP POST на ендпоінт /api/telemetry. Клас TelemetryController отримує об'єкт TelemetryDto, присвоює полю Timestamp поточний UTC-час і через ApplicationDbContext зберігає запис TelemetryRecord у таблиці SQLite. У разі недоступності сервера апаратний вузол продовжує роботу і повторює спроби відправки при відновленні з'єднання. Цей потік є базовим для всієї системи - без надходження телеметрії решта функцій переходять у режим симуляції.

Потік 2 – цикл ПІД-регулювання. Фоновий сервіс ClimateControlService виконується у окремому потоці з інтервалом 3 секунди. Сервіс запитує у ApplicationDbContext останній рядок таблиці TelemetryRecords та виконує вибір активного профілю мікроклімату відповідно до поточного часу. Якщо час останнього запису перевищує 15 секунд від поточного моменту, система автоматично переходить у режим симуляції та використовує розрахункову температуру симулятора Simulator.Temperature замість реальних показань датчика. Отримана температура передається до методу PidController.Compute(), що повертає значення $U(t)$ у діапазоні [мінус 100; 100] відсотків. Метод EnvironmentSimulator.Update() приймає $U(t)$ та часовий крок dt і розраховує нові значення Simulator.Temperature та Simulator.Humidity відповідно до спрощеної теплової моделі приміщення. Результати зберігаються у таблиці SystemLogs та надсилаються клієнтам через SignalR-хаб.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		29

Потік 3 - авторизація та захист доступу. Користувач вводить логін і пароль у форму входу веб-інтерфейсу. JavaScript-код виконує HTTP POST /api/auth/login. Клас AuthService перевіряє хеш пароля за алгоритмом BCrypt та видає підписаний JWT-токен зі строком дії 8 годин і полем role (Operator або Engineer). У разі невідповідності облікових даних сервер повертає статус HTTP 401, а веб-інтерфейс відображає повідомлення про помилку. У разі успіху токен зберігається у localStorage браузера та автоматично додається до заголовка Authorization: Bearer <token> кожного наступного запиту. Атрибут [Authorize(Roles = "Engineer")] на відповідних ендпоінтах захищає налаштування ПД-регулятора та профілів мікроклімату від несанкціонованого доступу.

Потік 4 - відображення у браузері. Після завантаження сторінки клієнтський JavaScript встановлює SignalR-з'єднання з хабом /climateHub. Фоновий сервіс щосекунди викликає метод Clients.All.SendAsync(). Обробник on("ReceiveUpdate") у браузері отримує об'єкт з полями temperature, humidity, heaterPower, coolerPower та оновлює точки даних на графіках Chart.js без перезавантаження сторінки. При втраті SignalR-з'єднання клієнт автоматично виконує повторне підключення та відображає відповідне попередження в інтерфейсі.

Функціональна схема показує не лише перелік можливостей системи, а й логіку доступу до них. Усі функції можна умовно поділити на інформаційні та керувальні. Інформаційні функції доступні будь-якому авторизованому користувачу і включають перегляд поточної телеметрії, графіків та журналу системних подій. Керувальні функції змінюють параметри роботи системи - коефіцієнти ПД-регулятора, профілі мікроклімату, налаштування setpoint - і тому захищені роллю "Інженер". Таке розділення відповідальностей є типовим підходом для промислових систем моніторингу, де оператор має право лише спостерігати за процесом, тоді як зміна алгоритмів регулювання залишається виключною компетенцією технічного персоналу.

Паралельно з обробкою HTTP-запитів фоновий сервіс ClimateControlService виконує безперервний цикл регулювання незалежно від активності користувачів у веб-інтерфейсі. Це означає, що система підтримує задані параметри мікроклімату навіть якщо жоден оператор не підключений до дашборду в даний момент - саме така поведінка є ключовою вимогою для промислового застосування.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		30

Цілодобова робота без постійної присутності персоналу є нормою, а не виключенням. При цьому всі події регулювання фіксуються у таблиці SystemLogs, що дозволяє інженеру переглянути повну хронологію роботи системи у будь-який зручний момент.

3.4 Розробка діаграми процесів

Діаграма процесів системи відображає послідовність станів та переходів основного циклу керування відповідно до вимог ДСТУ ISO 5807:2016. Діаграму процесів наведено на рисунку 3.3.

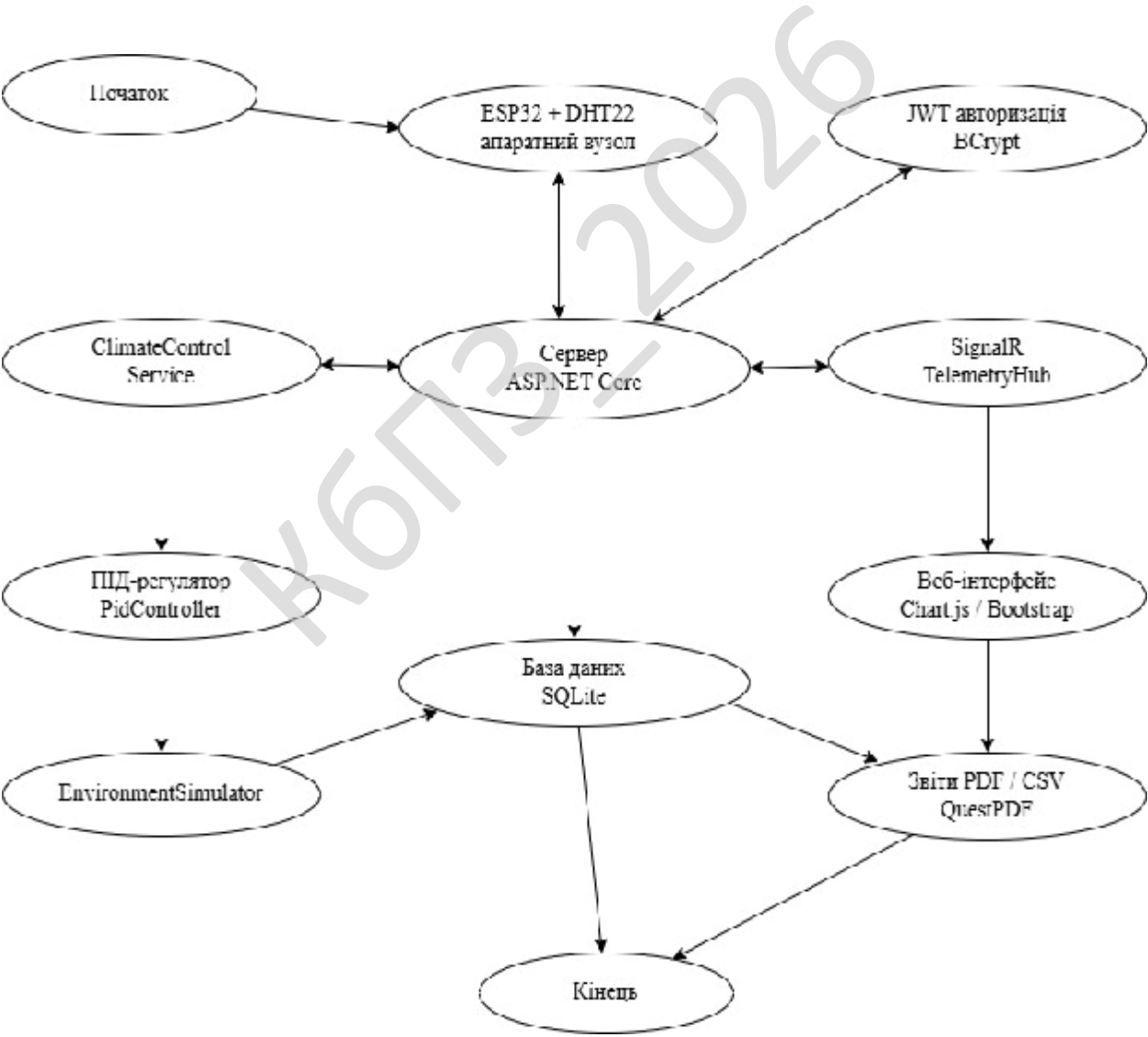


Рисунок 3.3 - Діаграма взаємодії процесів

Діаграма побудована навколо центрального вузла - серверної частини ASP.NET Core, яка є точкою взаємодії між усіма підсистемами. Від центрального вузла відходять три гілки: ліва відповідає за цикл регулювання (ClimateControlService , ПІД-регулятор, EnvironmentSimulator), права - за взаємодію з користувачем (SignalR, TelemetryHub, Веб-інтерфейс Звіти PDF/CSV), а нижня - за збереження даних (База даних SQLite). Апаратний вузол ESP32 + DHT22 та модуль JWT авторизації підключені до центрального сервера зверху. Така архітектура відповідає принципу найменших привілеїв і є типовою для промислових IoT-систем, де центральний сервер агрегує дані від апаратних вузлів та надає єдиний захищений інтерфейс керування.

Цикл взаємодії починається із запуску системи. Від блоку "Початок" керування передається до апаратного вузла ESP32 + DHT22, який після ініціалізації Wi-Fi та датчика DHT22 починає циклічно кожні 5 секунд зчитувати показання температури та вологості і надсилати їх на сервер через HTTP POST. Сервер ASP.NET Core приймає телеметрію та передає її одночасно у два напрями: до бази даних SQLite для збереження та до ClimateControlService для обробки. До авторизації користувач не має доступу до керувальних функцій - доступний лише перегляд публічних ендпоінтів.

Перший сценарій - збір та передача телеметрії. Апаратний вузол ESP32 + DHT22 є базовим джерелом фізичних даних для всієї системи. Після ініціалізації ESP32 виконує нескінченний цикл: перевірка стану Wi-Fi (якщо з'єднання розірвано - повторне підключення з інтервалом 5 секунд); перевірка інтервалу часу (якщо від попередньої відправки минуло менше 5 секунд - повернення на початок циклу); зчитування DHT22 (якщо повернуто NaN - запис помилки та повернення на початок); формування JSON-рядка та виконання HTTP POST на сервер. Без роботи апаратного вузла решта процесів автоматично переходять у режим симуляції через EnvironmentSimulator.

Другий сценарій - цикл регулювання. ClimateControlService запускається при старті ASP.NET Core та ініціалізує PidController з коефіцієнтами $K_p = 2.0$, $K_i = 0.5$, $K_d = 1.0$ і цільовою температурою Setpoint = 22.0 °C, а також EnvironmentSimulator з початковою температурою 20 °C. Сервіс виконує нескінченний цикл з затримкою 3 секунди: перевірка часу останньої телеметрії - якщо затримка перевищує 15 секунд, система переходить у режим симуляції та записує попередження у SystemLogs; перевірка поточного часу для

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		32

вибору активного профілю та оновлення `Pid.Setpoint;` виклик `Pid.Compute(temperature);` виклик `Simulator.Update(pidOutput, dt);` надсилання оновлених даних через SignalR-хаб; збереження запису у базі даних. Такий уніфікований процес обробки зручний для тестування - кожен етап можна перевірити окремо, що спрощує локалізацію помилок.

Третій сценарій - авторизація та захист доступу. Блок JWT авторизації `BCrypt` підключений безпосередньо до центрального сервера. Перевірка облікових даних відбувається виключно на стороні сервера - веб-клієнт лише збирає введені дані та передає їх серверу через `POST /api/auth/login`. Якщо дані правильні, сервер видає підписаний JWT-токен з роллю `Operator` або `Engineer`, який зберігається у `localStorage` браузера і автоматично додається до заголовків усіх наступних запитів. Якщо дані невірні - стан системи не змінюється і повертається `HTTP 401`. Така послідовність дозволяє уникнути ситуації, коли клієнт самовільно надає собі доступ до захищених функцій. Спроба виконати захищену дію без достатніх прав повертає `HTTP 403`, некоректний формат параметрів - `HTTP 400`. Використання різних `HTTP`-статусів дозволяє веб-клієнту точніше реагувати на помилки та відображати зрозуміле повідомлення замість загального повідомлення про збій.

Четвертий сценарій - відображення даних у реальному часі. `SignalR TelemetryHub` забезпечує постійне двостороннє з'єднання між сервером та браузером. При відкритті сторінки веб-клієнт встановлює `SignalR`-з'єднання з хабом `/climateHub` та підписується на подію `ReceiveUpdate`. Фоновий сервіс щосекунди викликає `Clients.All.SendAsync()` і передає оновлені значення температури, вологості та потужності виконавчих пристроїв. `Chart.js` оновлює графіки без перезавантаження сторінки. Зміни профілю або коефіцієнтів ПІД-регулятора, виконані інженером, через короткий час стають видимими у всіх підключених браузерах одночасно - це наочно демонструє, що всі клієнти працюють з одним спільним станом системи. При втраті зв'язку з апаратним вузлом відображається червоний банер попередження з прапором `isDeviceOffline`.

П'ятий сценарій - формування звітів. З веб-інтерфейсу користувач з роллю "Інженер" може сформувати звіти у форматах `PDF` або `CSV` через бібліотеку `QuestPDF`. Дані вибираються з бази даних `SQLite` за вказаний часовий діапазон через `ReportController`.

					<i>ВКРБ-122.26.0017.00.00.ПЗ</i>	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		33

PDF-звіт містить таблицю вимірювань та статистику (мінімальне, максимальне та середнє значення температури і вологості), CSV - серіалізовані дані з роздільником крапка з комою.

З точки зору тестування кожний блок схеми можна перевірити окремо. Спочатку перевіряється надходження телеметрії від ESP32 та її збереження у базі даних, потім правильна та неправильна авторизація, далі робота ПІД-регулятора при різних значеннях температури, після чого перевіряється коректність відображення даних у веб-інтерфейсі через SignalR. Такий підхід дозволяє швидко знайти місце помилки: в апаратному вузлі, у серверній логіці або у клієнтській частині. Обмеження поточної реалізації полягають у тому, що система розрахована на одну локальну мережу, а база даних SQLite не підходить для великих обсягів даних. У подальшому систему можна розширити: додати підтримку кількох апаратних вузлів одночасно, замінити SQLite на PostgreSQL та додати шифрування трафіку через HTTPS.

Таким чином, у третьому розділі сформовано повний опис проектних рішень системи: описано логіку функціонування на рівні компонентів, розроблено структурну та функціональну схеми, побудовано діаграму процесів. Матеріали розділу є основою для реалізації системи, описаної у розділі 4.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

У даному підрозділі наведено опис ключових алгоритмів, реалізованих у програмному забезпеченні системи. Для основних процесів функціонування розроблено блок-схеми відповідно до вимог ДСТУ ISO 5807:2016, а допоміжні алгоритми наведено у вигляді текстового опису. Розглядаються алгоритм основної програми мікроконтролера ESP32, алгоритм фонового сервісу керування ClimateControlService, а також алгоритми ПІД-регулятора та симулятора середовища.

Алгоритм основної прошивки

На рисунку 4.1 наведено блок-схему алгоритму роботи прошивки мікроконтролера ESP32, призначеної для збору даних із датчика температури та вологості DHT22 і передачі їх на сервер через мережу Wi-Fi.

Робота алгоритму починається з етапу ініціалізації системи. Після запуску мікроконтролера послідовно виконуються три кроки: налаштування серійного порту для налагодження, ініціалізація датчика температури та вологості DHT22, а також підключення до бездротової мережі Wi-Fi. Після завершення ініціалізації керування передається до з'єднувача M1, з якого починається основний цикл роботи програми.

На початку кожної ітерації циклу виконується перевірка наявності активного підключення до мережі Wi-Fi. Якщо з'єднання відсутнє, виконується процедура підключення до мережі Wi-Fi, після чого керування повертається до перевірки з'єднувача M1 і перевірка повторюється. За наявності активного підключення виконується перевірка часу, що минув з моменту останньої передачі даних - чи минуло 5 секунд від останньої передачі. Якщо зазначений інтервал ще не завершився, керування повертається до початку циклу через з'єднувач M1. Якщо інтервал завершився, виконується зчитування даних з датчика вологості та температури, після чого керування передається до з'єднувача M2.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		35

Від з'єднувача M2 виконується перевірка коректності зчитування даних. Якщо під час отримання показників виявлено помилку - дані не зчитано коректно - виконується реєстрація події в журналі налагодження, після чого алгоритм завершує поточну ітерацію. У разі успішного зчитування виконується формування телеметричного повідомлення у форматі JSON-рядка, яке містить актуальні значення температури та вологості. Сформоване повідомлення передається на сервер за допомогою HTTP POST-запиту, після чого оновлюється час останньої передачі даних, який використовується для контролю інтервалу наступного надсилання телеметрії. Після оновлення часу алгоритм завершує виконання поточної ітерації та повертається до з'єднувача M1 для початку наступного циклу.

Описаний алгоритм забезпечує автономну роботу вузла збору телеметрії та стійкість до короткочасних порушень мережевого з'єднання. Завдяки періодичному контролю стану Wi-Fi та автоматичному повторному підключенню підтримується надійний обмін даними між мікроконтролером і серверною частиною системи. Використання фіксованого інтервалу передачі у 5 секунд дозволяє забезпечити рівномірне надходження телеметричної інформації, необхідної для подальшої обробки, візуалізації та роботи алгоритмів автоматичного керування мікрокліматом. Прошивка написана мовою C++ у середовищі PlatformIO з використанням бібліотеки DHT.h від Adafruit та стандартної бібліотеки HTTPClient, що забезпечує її сумісність з усіма ESP32-сумісними платформами без суттєвих змін у логіці алгоритму. Окрему роль відіграє реєстрація помилок зчитування у серійному журналі - це дозволяє оперативно виявляти несправності датчика DHT22 безпосередньо на місці встановлення вузла без необхідності підключення до серверної частини. Алгоритм свідомо позбавлений надлишкової логіки: мікроконтролер виконує рівно одну функцію - збір і передачу телеметрії, що відповідає принципу єдиної відповідальності та мінімізує ймовірність збоїв у довгостроковій роботі. За необхідності інтервал передачі може бути змінений шляхом редагування константи у прошивці без будь-яких змін у серверній або клієнтській частинах системи, що підкреслює модульність розробленої архітектури.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

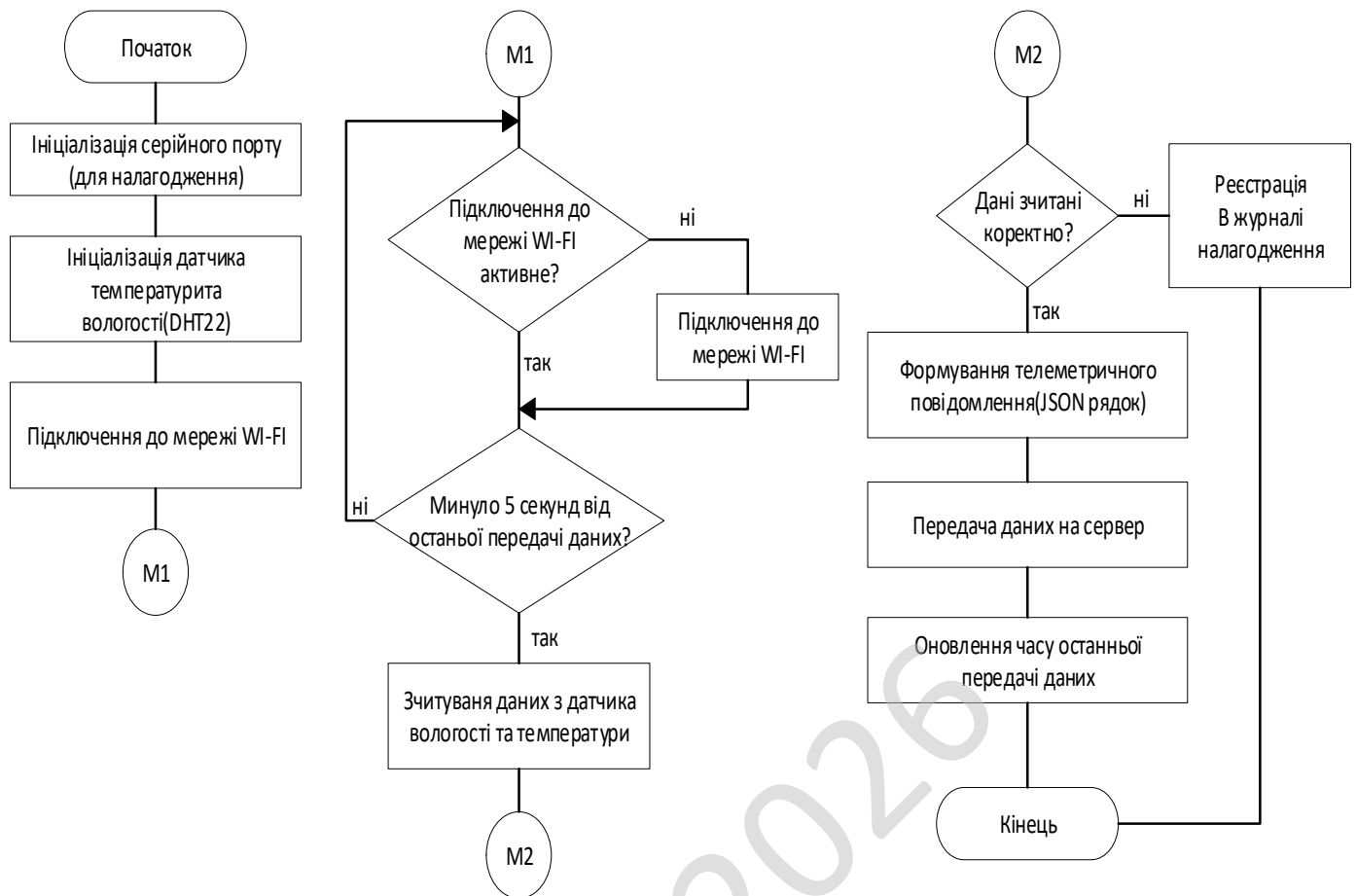


Рисунок 4.1 - Блок-схема алгоритму основної програми

Алгоритм фонового сервісу ClimateControlService

Алгоритм роботи сервісу керування мікрокліматом ClimateControlService наведено на рисунку 4.2. Після запуску виконується ініціалізація сервісу керування мікрокліматом, після чого процес переходить у циклічний режим роботи. На початку кожної ітерації виконується затримка тривалістю 3 секунди, що визначає період оновлення системи. Після затримки керування передається до з'єднувача M3.

Від з'єднувача M3 виконується перевірка токена скасування роботи сервісу. Якщо отримано сигнал скасування - керування передається до з'єднувача M4, який завершує виконання алгоритму. У протилежному випадку перевіряється наявність активного профілю керування за розкладом. Якщо такий профіль активний, встановлюється відповідне значення цільової температури.

Після встановлення цільової температури або якщо активного профілю немає - керування передається до з'єднувача M1.

Від з'єднувача M1 виконується перевірка актуальності даних від ESP32. Якщо дані з ESP32 актуальні - виконується отримання даних безпосередньо від апаратного вузла ESP32. Якщо дані застарілі або відсутні - виконується отримання даних із симулятора та формується запис попередження до системного журналу. В обох випадках після отримання даних виконується оновлення поточних параметрів мікроклімату - температури та вологості - після чого керування передається до з'єднувача M2.

Від з'єднувача M2 послідовно виконуються чотири кроки. Спочатку виконується обчислення керуючого впливу ПІД-регулятора на основі поточної температури та встановленого значення Setpoint. Далі виконується оновлення параметрів мікроклімату шляхом моделювання роботи системи через EnvironmentSimulator. Після цього формується системний журнал, до якого записуються поточна температура, цільова температура, значення керуючого впливу та стан нагрівальних і охолоджувальних пристроїв. Завершальним кроком є збереження розрахованих та отриманих даних у базі даних SQLite.

Після збереження даних керування повертається до з'єднувача M3, де повторно перевіряється токен скасування. Якщо сигнал скасування отримано - алгоритм завершує виконання через з'єднувач M4. Якщо ні - виконується наступна ітерація починаючи із затримки 3 секунди, що забезпечує безперервний цикл керування мікрокліматом протягом усього часу роботи серверного застосунку.

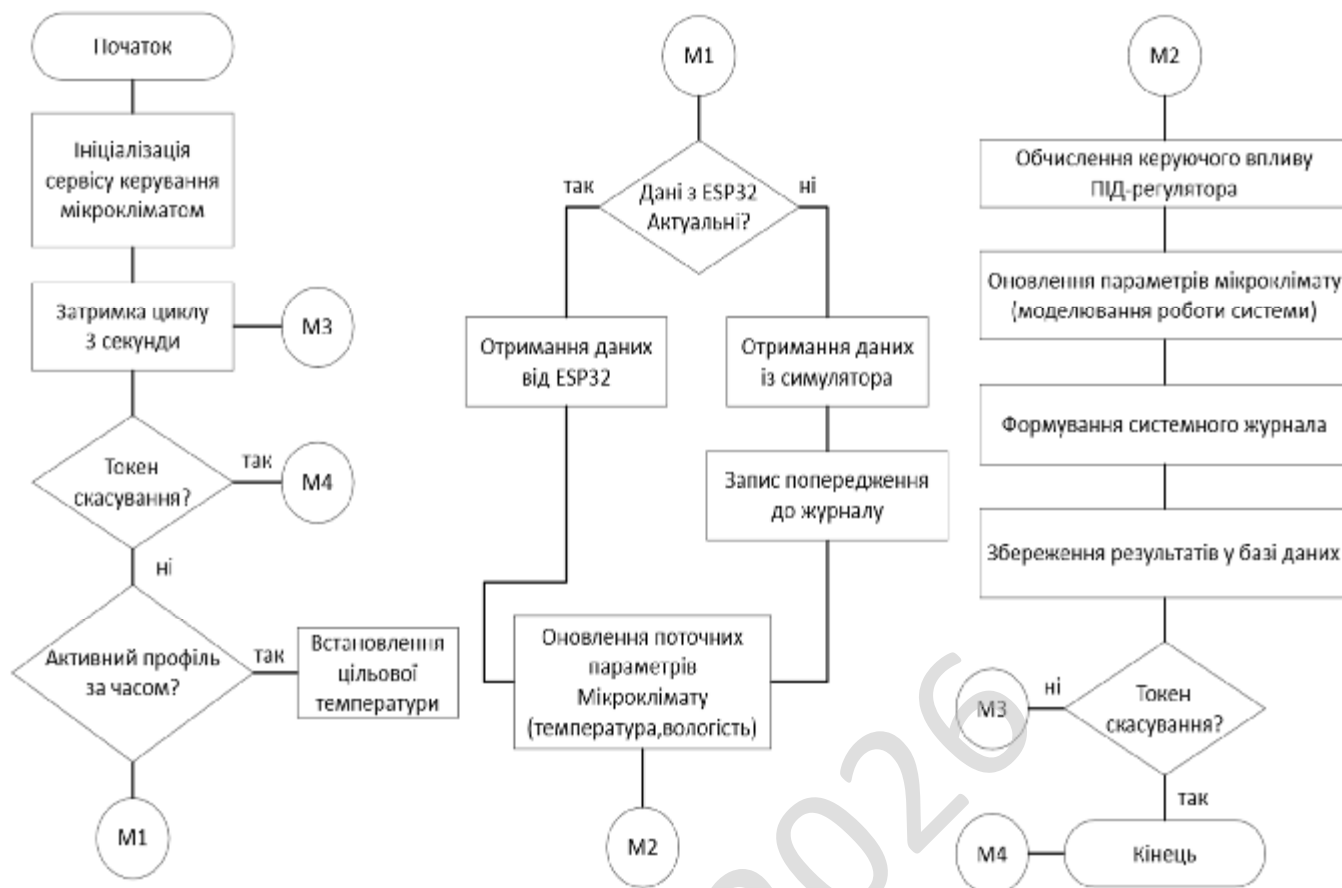


Рисунок 4.2. Блок-схема алгоритму фоновому сервісу ClimateControlService

Алгоритм роботи симулятора середовища EnvironmentSimulator

Клас EnvironmentSimulator призначений для програмного моделювання зміни температури та вологості у складському приміщенні. Він використовується як допоміжний компонент системи керування мікрокліматом і дозволяє підтримувати роботу програмного забезпечення навіть у випадку тимчасової втрати зв'язку з апаратним вузлом ESP32.

Основою роботи симулятора є спрощена модель теплового балансу. Зміна температури визначається дією нагрівача, охолоджувача та природним теплообміном із зовнішнім середовищем. Вхідним параметром методу Update(double pidOutput, double deltaSeconds) є керуючий сигнал ПІД-регулятора та часовий інтервал між двома обчисленнями.

На початку роботи методу значення `pidOutput` розподіляється між виконавчими пристроями. Якщо керуючий сигнал є додатним, активується нагрівач, а його потужність записується у змінну `HeaterPower`. Якщо сигнал є від'ємним, активується охолоджувач, а його потужність визначається за модулем і записується у змінну `CoolerPower`. Якщо значення `pidOutput` дорівнює нулю, потужність нагрівача та охолоджувача обнуляється.

Після цього виконується розрахунок впливу нагрівання на температуру приміщення. Значення `heatDelta` обчислюється за формулою $\text{HeaterPower} \times 0,0005 \times \text{deltaSeconds}$. Коефіцієнт 0,0005 визначає швидкість зростання температури при роботі нагрівача. Аналогічно обчислюється вплив охолодження: $\text{coolDelta} = \text{CoolerPower} \times 0,0004 \times \text{deltaSeconds}$. Для врахування природного теплообміну із зовнішнім середовищем використовується складова $\text{ambientDelta} = (18,0 - \text{Temperature}) \times 0,001 \times \text{deltaSeconds}$, де 18,0 °C прийнято як умовну температуру навколишнього середовища.

Підсумкове значення температури оновлюється шляхом додавання теплового приросту, віднімання охолоджувальної дії та врахування природного теплообміну. Оновлення виконується за формулою: $\text{Temperature} = \text{Temperature} + \text{heatDelta} - \text{coolDelta} + \text{ambientDelta}$. Таким чином, модель відображає поступову зміну температури без різких стрибків.

Вологість у симуляторі також перераховується після кожного циклу оновлення. Її зміна залежить від нагрівання та охолодження: нагрівання умовно збільшує розрахункову зміну вологості, а охолодження її зменшує. Щоб значення залишалося у фізично допустимих межах, використовується обмеження діапазону від 20 до 95 %. Це реалізовано за допомогою методу `Math.Clamp`.

Окремо у класі передбачено метод `SetRealData(double temperature, double humidity)`, який синхронізує внутрішній стан симулятора з фактичними показаннями датчика. Після відновлення зв'язку з ESP32 симулятор приймає реальні значення температури та вологості як початкові для подальших розрахунків. Це дозволяє уникнути різких змін на графіках і забезпечує плавний перехід між режимом симуляції та роботою з реальними сенсорними даними.

Отже, `EnvironmentSimulator` виконує роль цифрової моделі середовища, яка забезпечує безперервність роботи системи керування мікрокліматом, підтримує

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

тестування ПІД-регулятора та дозволяє відображати розрахунковий стан приміщення у веб-інтерфейсі навіть за відсутності актуальної телеметрії з датчика.

Алгоритм ПІД-регулятора

ПІД-регулятор (пропорційно-інтегрально-диференціальний) є основним алгоритмом автоматичного керування системою. Клас `PidController` реалізований мовою програмування C#. Поточні значення коефіцієнтів за замовчуванням становлять: $K_p = 2,0$, $K_i = 0,5$, $K_d = 1,0$, а цільова температура $Setpoint = 22,0$ °C. Усі параметри реалізовані як публічні властивості та можуть змінюватися через веб-інтерфейс інженера без перезавантаження сервера.

Метод `Compute()` реалізує алгоритм обчислення керуючого впливу на основі поточного значення температури. На першому етапі визначається часовий крок між двома послідовними викликами алгоритму:

$$dt = t_{current} - t_{previous} \quad (4.1)$$

Для запобігання помилкам обчислення при нульовому або від'ємному значенні часового кроку використовується його примусове встановлення рівним 1 с.

Далі визначається поточна похибка регулювання як різниця між заданою та фактичною температурою:

$$e(t) = T_{set} - T(t) \quad (4.2)$$

Після цього виконується накопичення інтегральної складової регулятора. Для запобігання надмірному накопиченню інтегральної помилки її значення обмежується діапазоном від -100 до 100 .

Диференціальна складова визначається за швидкістю зміни похибки:

$$D(t) = \frac{e(t) - e(t-1)}{dt} \quad (4.3)$$

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

Підсумковий керуючий вплив обчислюється за класичним законом ПД-регулювання:

$$U(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt} \quad (4.4)$$

Отримане значення виходу обмежується діапазоном від -100% до $+100\%$. Після завершення обчислень поточні значення похибки та часу зберігаються для використання під час наступного циклу роботи регулятора. Фізичний зміст виходу регулятора полягає в тому, що додатне значення $U(t)$ означає необхідність нагріву приміщення, оскільки поточна температура нижча за задану. Від'ємне значення $U(t)$ свідчить про необхідність охолодження. Абсолютна величина вихідного сигналу визначає відносну потужність виконавчого пристрою в діапазоні від 0 до 100 %.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення від несанкціонованого доступу та неправомірного використання забезпечується на кількох рівнях: на рівні мережевого протоколу, на рівні автентифікації та авторизації, а також на рівні організаційно-правових заходів.

Система автентифікації та авторизації на основі JWT

Серверні ресурси захищені через JSON Web Token (JWT) - стандарт RFC 7519 із компактним форматом передачі даних у вигляді JSON-об'єкта з цифровим підписом HMAC-SHA256.

Вхід починається з POST `/api/auth/login` із тілом `{"username": "...", "password": "..."}.` AuthService шукає запис у таблиці Users за іменем і звіряє пароль із BCrypt-хешем. При збігу збирається токен: sub - ім'я користувача, role - Operator або Engineer, exp - поточний час плюс 8 годин. Підпис ставиться ключем із `Jwt:SecretKey` у `appsettings.json`.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42

Розмежування прав на ендпоінтах - через атрибути. [Authorize] не пропускає запит без токена і повертає HTTP 401. [Authorize(Roles = "Engineer")] стоїть на ендпоінтах зміни ПД-коефіцієнтів і керування профілями - токен із роллю Operator там отримує HTTP 403.

На стороні браузера токен осідає в localStorage і кожен API-запит отримує його в заголовок Authorization: Bearer <token> через getAuthHeaders(). Під час відкриття будь-якої сторінки скрипт перевіряє токен - немає або прострочений, користувач летить на сторінку входу.

Захист паролів методом BCrypt

Паролі в базі не зберігаються у відкритому вигляді - лише BCrypt-хеш. BCrypt побудований на шифрі Blowfish і має параметр cost factor: чим він вищий, тим більше обчислень потрібно на один хеш, що прямо впливає на вартість перебору при зростанні потужності заліза.

В системі cost factor = 11 - час хешування на сучасному процесорі виходить близько 100 мс, що робить масовий перебір нежиттєздатним.

При вході викликається BCrypt.Verify(enteredPassword, storedHash): з хешу витягується оригінальний salt, пароль хешується з тими ж параметрами і результат порівнюється зі збереженим. Якщо файл бази SQLite потрапить стороннім - вони побачать лише рядки хешів, відновити з них паролі прямим читанням неможливо.

Таким чином, захист паролів у розробленій системі реалізовано на достатньому для виробничого застосування рівні. Поєднання BCrypt-хешування з індивідуальним salt для кожного пароля та JWT-авторизації з розмежуванням ролей забезпечує надійний захист як облікових даних користувачів, так і серверних ресурсів від несанкціонованого доступу. На мою думку, обраний підхід є обґрунтованим вибором для системи такого класу - він не ускладнює архітектуру надлишковими механізмами безпеки, але водночас відповідає рекомендаціям OWASP щодо мінімально прийнятного рівня захисту веб-застосунків промислового призначення.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Результати тестування системи

Для підтвердження коректності розроблених алгоритмів були написані модульні тести (unit tests) з використанням фреймворку xUnit. Проєкт тестів розташований у папці /backend/ClimateControl.Tests.

Тест 1 - перевірка позитивного виходу ПІД при температурі нижчій за цільову. Встановлюється Setpoint = 22,0 °C та подається поточна температура 18,0 °C. Очікується, що метод Compute() поверне позитивне значення (команда на нагрів). Результат: тест пройдено, output = 8,0 (значення пропорційної складової $K_p \times e = 2,0 \times 4,0$ при першому виклику з $dt = 1,0$ с).

Тест 2 - перевірка від'ємного виходу ПІД при температурі вищій за цільову. Setpoint = 22,0 °C, поточна температура = 26,0 °C. Очікується від'ємне значення виходу (команда на охолодження).

Результат: тест пройдено, output = -8,0.

Тест 3 - перевірка зростання температури в симуляторі при 100 % нагріві. Задається початкова температура симулятора 20,0 °C. Викликається Update(100.0, 10.0) (100 % потужність нагрівача, 10 секунд). Очікується, що Temperature після виклику буде більшою за 20,0 °C. Результат: тест пройдено, Temperature = 20,50 °C (приріст: $100 \times 0,0005 \times 10 = 0,5$ °C).

Тест 4 - перевірка зниження температури в симуляторі при 100 % охолодженні. Початкова температура 25,0 °C, Update(-100.0, 10.0). Очікується Temperature < 25,0 °C. Результат: тест пройдено, Temperature = 24,62 °C (зниження: $100 \times 0,0004 \times 10 = 0,4$ °C мінус природний приріст від ambient).

Результати виконання тестів у Visual Studio наведено на рисунку 5.1

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

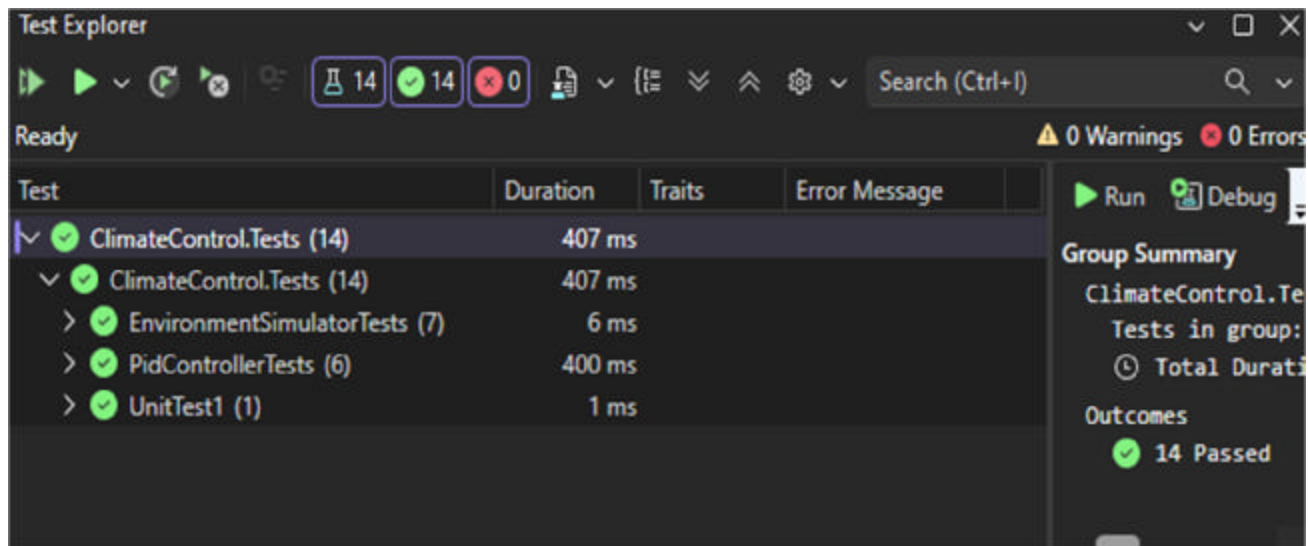


Рисунок 5.1 - Результати виконання модульних тестів у Visual Studio TestExplorer

Демонстрація роботи веб-інтерфейсу

На рисунку 5.2 наведено головний екран веб-інтерфейсу оператора у режимі реального часу. На дашборді відображаються поточні значення температури та відносної вологості, лінійний графік зміни температури за останні хвилини, графік потужності нагрівача та охолоджувача, а також індикатор активного профілю мікроклімату. Оновлення показників виконується автоматично через технологію SignalR без перезавантаження сторінки, що забезпечує оперативний моніторинг стану системи. Крім того, інтерфейс надає користувачу можливість швидко оцінити динаміку зміни параметрів мікроклімату та ефективність роботи алгоритму керування за допомогою наочних графічних елементів. У разі втрати зв'язку з апаратним вузлом ESP32 на дашборді автоматично з'являється червоний банер з попередженням, що одразу привертає увагу оператора до проблеми. Веб-інтерфейс реалізовано з урахуванням адаптивного дизайну на основі Bootstrap 5, тому він однаково коректно відображається як на персональному комп'ютері, так і на планшеті або смартфоні у локальній мережі підприємства.

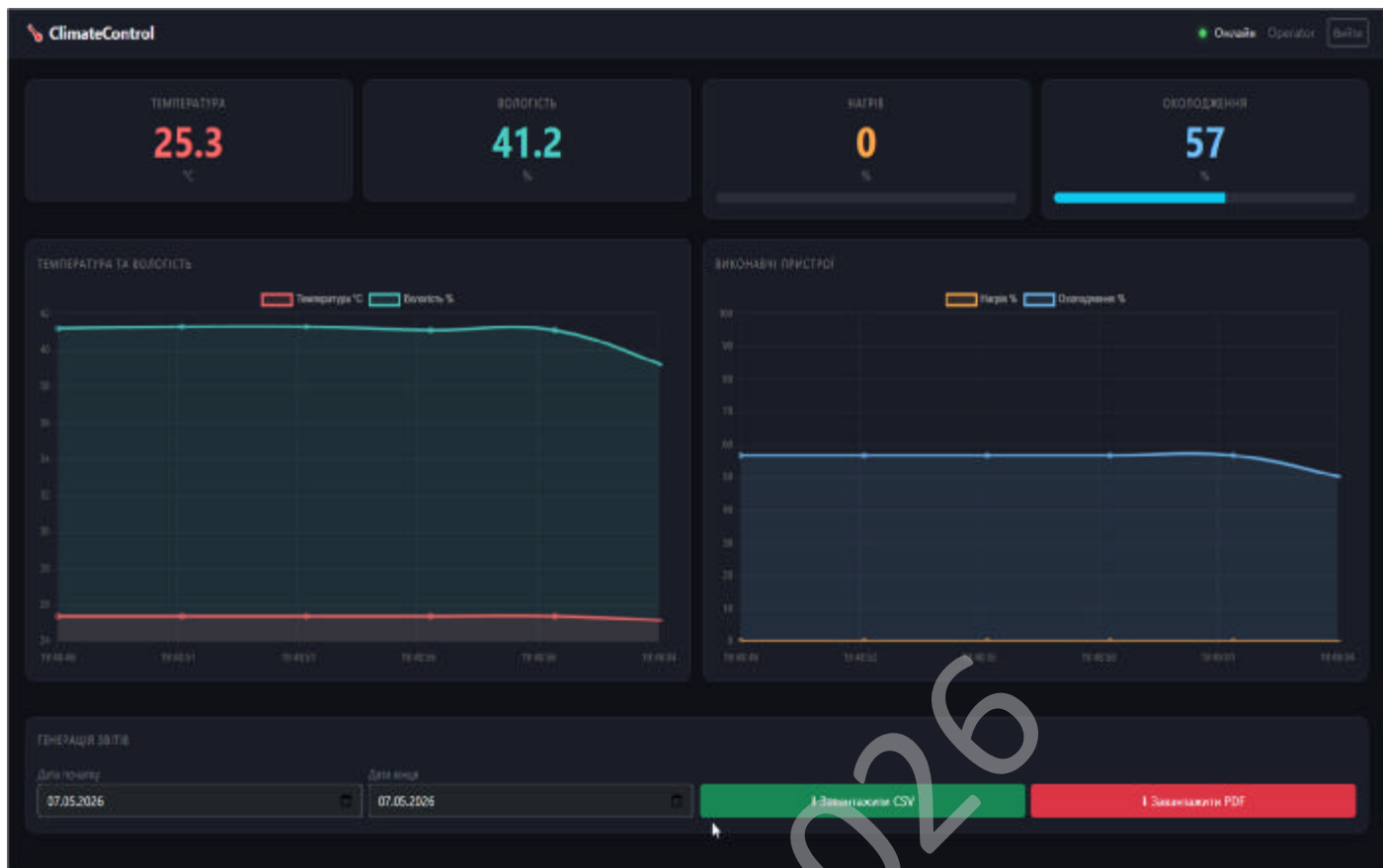


Рисунок 5.2 - Головний екран веб-інтерфейсу оператора

На рисунку показано панель інженера з формами для налаштування коефіцієнтів ПД-регулятора (K_p , K_i , K_d) та цільової температури *setpoint*, а також форму для створення нового профілю мікроклімату із зазначенням назви, цільової температури та часового діапазону дії. Використання даної панелі дозволяє оперативно змінювати параметри керування системою відповідно до технологічних вимог конкретного приміщення без необхідності втручання у програмний код. Налаштовані параметри та створені профілі зберігаються у базі даних і автоматично враховуються під час роботи алгоритму керування, що забезпечує гнучке налаштування режимів функціонування системи та адаптацію до змін умов експлуатації.

Завдяки цьому інженер може швидко коригувати параметри регулювання та розклад роботи профілів, забезпечуючи підтримання необхідних параметрів мікроклімату в різних режимах експлуатації об'єкта.

ГЕНЕРАЦІЯ ЗВІТІВ

Дата початку

07.05.2026

Дата кінця

07.05.2026

Завантажити CSV

Завантажити PDF

НАЛАШТУВАННЯ ПІД

Кр

2,0

Ки

0,5

Кд

1,0

Цільова температура (Setpoint °C)

22,0

Застосувати налаштування

НОВИЙ ПРОФІЛЬ КЛІМАТУ

Створити профіль

Рисунок 5.3 - Панель налаштувань інженера: коефіцієнти ПІД та профілі клімату

На рисунку 5.4 наведено приклад відображення аварійного банера при відключенні ESP32: у верхній частині інтерфейсу з'являється червоний банер з текстом попередження про втрату зв'язку з пристроєм та інформацією про перехід системи у режим симуляції.

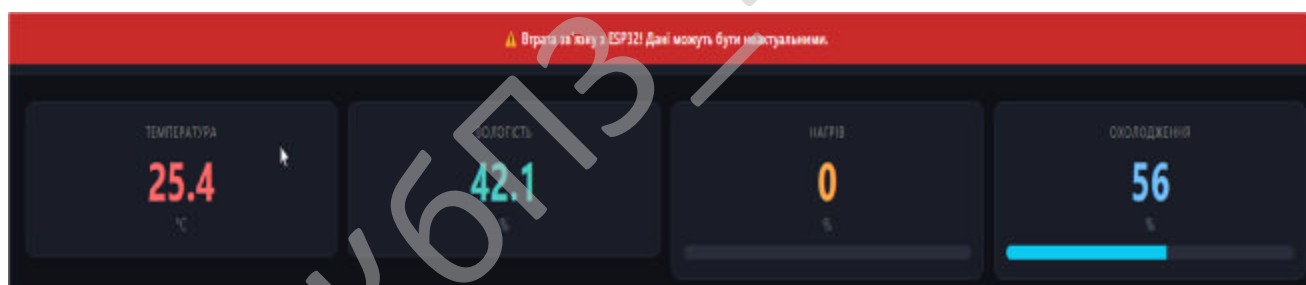


Рисунок 5.4 - Відображення аварійного банера при втраті зв'язку з ESP32

Коли пристрій втрачає зв'язок із сервером, користувач негайно про це дізнається завдяки яскравому червоному банеру, який з'являється у верхній частині екрана.

Цей банер займає всю ширину вікна браузера і містить чітке повідомлення про те, що пристрій офлайн, тому поточні вимірювання могли бути недостовірними.

На практиці це запобігає ситуаціям, коли оператор спирається на застарілі дані, думаючи, що система працює в режимі реального часу.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		47

Та сама інформація про статус підключення передається через WebSocket у кожному пакеті телеметрії, тому навіть якщо банер з якихось причин не видно, дані в логах однозначно вказуватимуть на розрив зв'язку. Банер автоматично зникає, як тільки ESP32 відновить з'єднання і першого разу відправить оновлення.

Формування звітів PDF та CSV

На рисунку 5.5 показано форму вибору параметрів звіту. Інженер вказує початкову і кінцеву дату, обирає формат файлу та відмічає потрібні показники - температуру, вологість, стан нагрівача й охолоджувача. Після підтвердження параметрів система виконує вибірку даних із бази даних за заданий період та формує звіт у вибраному форматі для подальшого завантаження користувачем.

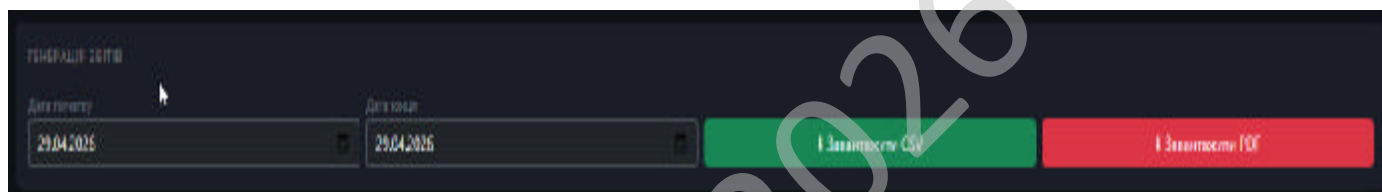


Рисунок 5.5 - Форма вибору параметрів перед генерацією звіту

На рисунку 5.6 видно випадаюче меню вибору дат. Два поля - початок і кінець періоду - після заповнення запускають генерацію. Завантаження починається автоматично.

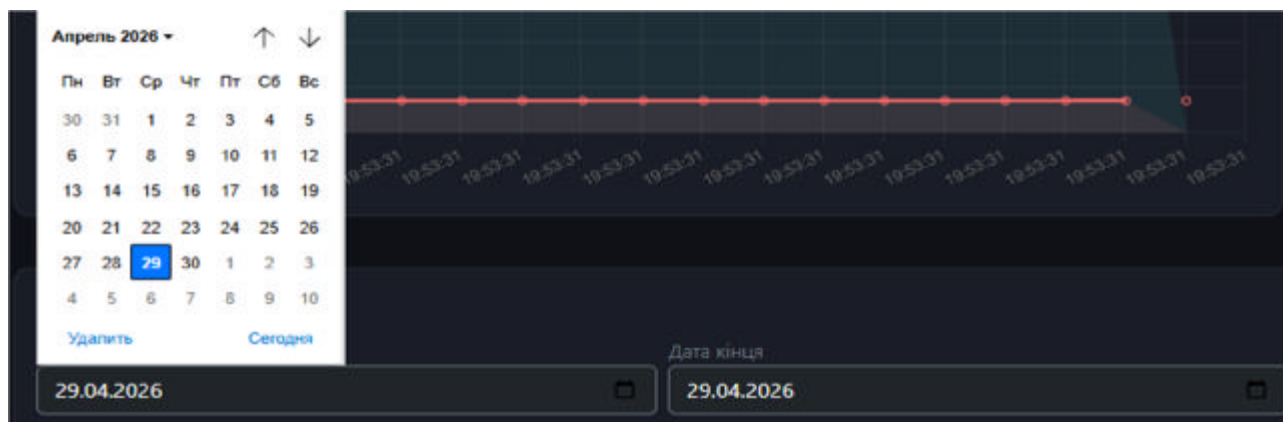


Рисунок 5.6 - Вікно вибору часового діапазону

У згенерованому PDF-звіті першою відображається службова частина документа, яка містить назву системи та часовий діапазон, за який сформовано звіт. Після заголовка розміщується таблиця зі зведеними статистичними показниками, розрахованими на основі телеметричних даних, що зберігаються у базі даних.

У таблиці відображаються кількість отриманих записів телеметрії, середнє, мінімальне та максимальне значення температури, а також середнє значення відносної вологості повітря за вибраний період. Розрахунок зазначених показників виконується на серверній стороні шляхом аналізу вибірки даних, отриманої з бази даних SQLite.

Наявність зведеної статистики дозволяє оперативно оцінити параметри мікроклімату за визначений проміжок часу, виявити можливі відхилення від заданих значень та проаналізувати ефективність роботи системи автоматичного керування. Сформований PDF-документ може використовуватися для архівування інформації, контролю технологічних процесів та підготовки звітної документації.

Звіт системи керування мікрокліматом

Період: 29.04.2026 — 29.04.2026

Зведена статистика

Кількість записів	220
Середня температура	24,2 °C
Мін. температура	23,7 °C
Макс. температура	25,4 °C
Середня вологість	54,9 %

Детальні дані телеметрії

№	Час	Темп. °C	Волог. %
1	29.04.2026 14:56:03	25,4	53,6
2	29.04.2026 14:56:05	25,4	53,7
3	29.04.2026 14:56:11	25,3	54,0
4	29.04.2026 14:56:15	25,4	53,9
5	29.04.2026 14:56:21	25,3	54,0
6	29.04.2026 14:56:25	25,3	54,0
7	29.04.2026 14:56:30	25,3	54,0
8	29.04.2026 14:56:35	25,3	54,1
9	29.04.2026 14:56:40	25,3	54,2
10	29.04.2026 14:56:45	25,2	54,2
11	29.04.2026 14:56:50	25,2	54,2
12	29.04.2026 14:56:56	25,2	54,2
13	29.04.2026 14:57:00	25,2	54,4
14	29.04.2026 14:57:05	25,2	54,4

Рисунок 5.7 - Приклад вигляду PDF-звіту

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

Експорт у формат CSV формує текстовий файл табличної структури, у якому кожен рядок відповідає окремому запису телеметрії, а стовпці містять часову мітку, значення температури та вологості, стан виконавчих пристроїв і назву активного профілю мікроклімату. Такий формат забезпечує сумісність із Microsoft Excel та іншими системами аналізу даних, що спрощує подальшу обробку, фільтрацію та статистичний аналіз отриманої інформації.

Обидва формати збираються на сервері. Бекенд тягне потрібні записи з бази, рахує агрегати і віддає файл із заголовком Content-Disposition - браузер сам відкриває діалог збереження. CSV (Comma-Separated Values) є текстовим форматом, який легко імпортується в табличні редактори та спеціалізоване ПЗ для аналізу даних, тоді як PDF забезпечує форматовану презентацію результатів з графіками та підсумками, придатну для офіційної документації та звітування.

Генерування звітів відбувається динамічно на основі вибраного користувачем часового діапазону, що дозволяє гнучко отримувати дані за будь-який період роботи системи. Потік завантаження файлу зберігається у пам'яті сервера і відправляється клієнту без попереднього збереження на диск, що зменшує навантаження на файлову систему і прискорює доставку файлу до користувача.

Фрагмент сформованого CSV-файлу наведено на рисунку 5.8

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

	A	B	C	D	E
1	Id,DeviceId,Temperature,Humidity,Timestamp				
2	97,ESP32_01,25,40,53,60,2026-04-29 14:56:03				
3	98,ESP32_01,25,40,53,70,2026-04-29 14:56:05				
4	99,ESP32_01,25,30,54,00,2026-04-29 14:56:11				
5	100,ESP32_01,25,40,53,90,2026-04-29 14:56:15				
6	101,ESP32_01,25,30,54,00,2026-04-29 14:56:21				
7	102,ESP32_01,25,30,54,00,2026-04-29 14:56:25				
8	103,ESP32_01,25,30,54,00,2026-04-29 14:56:30				
9	104,ESP32_01,25,30,54,10,2026-04-29 14:56:35				
10	105,ESP32_01,25,30,54,20,2026-04-29 14:56:40				
11	106,ESP32_01,25,20,54,20,2026-04-29 14:56:45				
12	107,ESP32_01,25,20,54,20,2026-04-29 14:56:50				
13	108,ESP32_01,25,20,54,20,2026-04-29 14:56:56				
14	109,ESP32_01,25,20,54,40,2026-04-29 14:57:00				
15	110,ESP32_01,25,20,54,40,2026-04-29 14:57:05				
16	111,ESP32_01,25,10,54,30,2026-04-29 14:57:10				
17	112,ESP32_01,25,10,54,30,2026-04-29 14:57:15				
18	113,ESP32_01,25,10,54,40,2026-04-29 14:57:20				
19	114,ESP32_01,25,10,54,50,2026-04-29 14:57:25				
20	115,ESP32_01,25,10,54,40,2026-04-29 14:57:30				
21	116,ESP32_01,25,00,54,50,2026-04-29 14:57:35				
22	117,ESP32_01,25,00,54,50,2026-04-29 14:57:40				
23	118,ESP32_01,24,90,54,50,2026-04-29 14:57:45				
24	119,ESP32_01,24,90,54,50,2026-04-29 14:57:50				
25	120,ESP32_01,24,90,54,70,2026-04-29 14:57:55				
26	121,ESP32_01,24,90,54,70,2026-04-29 14:58:00				
27	122,ESP32_01,24,90,54,80,2026-04-29 14:58:06				
28	123,ESP32_01,24,90,54,80,2026-04-29 14:58:11				
29	124,ESP32_01,24,80,55,00,2026-04-29 14:58:15				
30	125,ESP32_01,24,80,54,90,2026-04-29 14:58:20				
31	126,ESP32_01,24,80,54,90,2026-04-29 14:58:26				
32	127,ESP32_01,24,80,55,00,2026-04-29 14:58:31				
33	128,ESP32_01,24,80,55,10,2026-04-29 14:58:36				
34	129,ESP32_01,24,80,55,00,2026-04-29 14:58:41				

Рисунок 5.8 Приклад вигляду CSV даних

Таким чином, у четвертому розділі детально описано реалізацію всіх ключових алгоритмів системи: прошивки ESP32, ПІД-регулятора, фоновому сервісу та симулятора середовища. Розглянуто механізми захисту програмного забезпечення: JWT-автентифікацію з розмежуванням ролей та BCrypt-хешування паролів. Наведено результати модульного тестування, демонстрацію роботи веб-інтерфейсу та формування звітів PDF/CSV. Описані рішення забезпечують надійність, безпеку та зручність експлуатації системи як у лабораторних умовах, так і при розгортанні в реальному виробничому середовищі.

Вимоги до інфраструктури та підготовка до розгортання

Перед розгортанням системи на об'єкті необхідно впевнитись у наявності відповідної технічної інфраструктури. Мінімальні вимоги до серверного обладнання для обслуговування одного складського приміщення є невисокими: будь-який комп'ютер або одноплатний мікрокомп'ютер (наприклад, Raspberry Pi 4) з процесором з тактовою частотою від 1 ГГц, оперативною пам'яттю від 512 МБ та дисковим простором від 1 ГБ. Операційна система: Windows 10/11, Ubuntu 20.04 LTS або будь-який дистрибутив Linux з підтримкою .NET 8 Runtime. На практиці навіть старіші комп'ютери часто справляються з навантаженням однієї системи моніторингу, адже обсяг даних, які потрібно обробляти, невеликий.

Необхідне програмне забезпечення на сервері обмежується лише середовищем виконання .NET 8 Runtime, яке безкоштовно розповсюджується компанією Microsoft та встановлюється одним пакетом без додаткових залежностей. Окремий сервер баз даних не потрібен - SQLite функціонує як вбудована бібліотека в межах процесу ASP.NET Core. Це суттєво спрощує розгортання і зменшує вимоги до адміністративної підтримки системи.

Вимоги до мережевої інфраструктури передбачають наявність Wi-Fi точки доступу стандарту IEEE 802.11 b/g/n або новішої версії у зоні розміщення апаратних вузлів ESP32 зі стабільним сигналом (RSSI не гірше мінус 70 дБм). Сервер повинен бути доступний в тій же локальній мережі (LAN/WLAN), до якої підключаються вузли ESP32, або у мережі з прямою маршрутизацією до них. Для доступу оператора з корпоративної мережі потрібне налаштування мережевого маршруту або VPN-підключення до серверного сегмента, що забезпечує безпеку передачі даних. Важливо налаштувати статичну IP-адресу або DNS для серверного хоста, щоб уникнути необхідності перепрошивки ESP32 при зміні адреси сервера.

Процедура розгортання програмного забезпечення

Розгортання системи виконується у чітко визначеній послідовності. На першому кроці на сервері встановлюється .NET 8 Runtime за допомогою менеджера пакетів або офіційного інсталятора. Файли backend-застосунку (скомпільований publish-пакет) копіюються до цільової директорії, наприклад /opt/climatecontrol/. У файлі appsettings.json налаштовуються параметри: рядок підключення до SQLite (шлях до файлу .db), секретний

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		52

ключ JWT (не менше 32 символів), дозволені CORS-джерела та URL-адреса прослуховування сервера (наприклад, <http://0.0.0.0:5000>).

На другому кроці виконується перша міграція бази даних: команда `dotnet ClimateControl.API.dll --migrate` або через Entity Framework Tools: `dotnet ef database update` створює файл `climatecontrol.db` зі схемою таблиць `TelemetryRecords`, `Users`, `Profiles`, `SystemLogs`. До таблиці `Users` вноситься початковий обліковий запис інженера з паролем, захищеним BCrypt. Цей крок критично важливий, адже без правильної структури бази даних система не зможе функціонувати.

На третьому кроці налаштовується автоматичний запуск сервісу. На Linux системах це реалізується через `systemd`: створюється файл `/etc/systemd/system/climatecontrol.service` з параметрами `Restart=always` та `WorkingDirectory`, після чого виконуються команди `systemctl enable climatecontrol` та `systemctl start climatecontrol`. На Windows-системах застосунок реєструється як Windows Service за допомогою вбудованих засобів .NET Worker Service. Обидва варіанти забезпечують автоматичний перезапуск сервісу у разі збою або перезавантаження операційної системи, що гарантує надійність системи у 24/7 режимі роботи.

На четвертому кроці прошивається кожен апаратний вузол ESP32.

У файлі `firmware.ino` змінюються три критичні константи: `WIFI_SSID`, `WIFI_PASSWORD` та `SERVER_URL` (наприклад, [http://192.168.1.100:5000 /api/telemetry](http://192.168.1.100:5000/api/telemetry)). Скомпільований бінарний файл завантажується на ESP32 через Arduino IDE або утиліту `esptool.py`. Після завантаження у Serial Monitor перевіряється успішне підключення до Wi-Fi та отримання відповіді HTTP 200 від сервера. Цей крок потребує особливої уваги, адже помилки у конфігурації можуть привести до того, що вузол не зможе передавати дані протягом тривалого часу.

Налаштування та калібрування системи

Після першого запуску системи інженер виконує початкове налаштування через веб-інтерфейс за адресою <http://<IP-сервера>:5000>. Вхід здійснюється з обліковим записом Engineer. На панелі налаштувань визначаються профілі мікроклімату відповідно до технологічного регламенту зберігання продукції на конкретному складі.

					<i>ВКРБ-122.26.0017.00.00.ПЗ</i>	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		53

Цей етап передбачає детальне вивчення вимог до мікроклімату для вашого типу продукції та узгодження їх з технічними можливостями системи.

Для складів лакофарбових матеріалів рекомендується така конфігурація профілів: робочий режим з 08:00 до 18:00 у робочі дні з цільовою температурою 20 °C та допустимим діапазоном 15-25 °C; нічний економ режим з 18:00 до 08:00 з цільовою температурою 15 °C для мінімального енергоспоживання при збереженні умов зберігання; аварійний режим, який активується вручну або за критичним сигналом, з цільовою температурою 10 °C та максимальною потужністю охолоджувача для ліквідації критичних ситуацій.

Налаштування коефіцієнтів ПІД-регулятора виконується методом Зіглера-Нікольса або практичним підбором залежно від характеристик приміщення. Рекомендований початковий набір коефіцієнтів: $K_p = 2,0$; $K_i = 0,5$; $K_d = 1,0$. Інженер спостерігає за графіком температури у реальному часі та коригує коефіцієнти до досягнення прийнятного часу встановлення і мінімального перерегулювання. Якщо спостерігається значне перерегулювання (температура суттєво перевищує setpoint) - зменшується K_i . Якщо реакція системи надто повільна - збільшується K_p . Процес налаштування можна вважати завершеним, коли температура стабільно тримається в межах допустимого діапазону без різких коливань протягом декількох днів. Варто також враховувати сезонні особливості вашого регіону, адже влітку та взимку система може потребувати різних налаштувань.

Технічне обслуговування та супровід

Система не вимагає постійного технічного обслуговування після успішного запуску. Рекомендований регламент обслуговування включає регулярні перевірки та профілактичні заходи для забезпечення довгострокової надійності.

Щотижневе обслуговування передбачає перегляд журналу SystemLogs через веб-інтерфейс для виявлення повторюваних попереджень, які можуть вказати на потенційні проблеми; перевірка розміру файлу бази даних та при обсягу понад 500 МБ рекомендується архівування старих записів для оптимізації продуктивності. Крім того, варто перевірити доступність веб-інтерфейсу та переконатися, що всі ESP32 активно надсилають дані. Якщо деякі вузли перестали передавати інформацію, це може свідчити про проблеми з Wi-Fi зв'язком або розряд батарей.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк. 54
Змн.	Арк.	№ докум.	Підпис	Дата		

Щомісячне обслуговування включає перевірку актуальності налаштувань профілів відповідно до сезонних змін умов зберігання та технологічних вимог виробництва; генерація PDF-звіту за місяць та передача до служби якості підприємства для аналізу; перевірка фізичного стану апаратних вузлів ESP32 (надійність контактів, відсутність конденсату на платі, цілісність кабелів).

На цьому етапі також варто переглянути статистику аварійних сигналізацій та визначити, чи не потрібне додаткове налаштування порогів сповіщень. Рекомендується також очистити вентиляційні отвори мікроконтролерів від пилу та бруду, адже накопичення матеріалу може негативно вплинути на теплорозсіювання та стабільність роботи.

Щорічне обслуговування передбачає калібрування датчиків DHT22 шляхом порівняння їх показань з атестованим еталонним термогігрометром для забезпечення точності вимірювань. Датчики вологості можуть з часом втрачати точність, тому калібрування варто проводити систематично. Крім того, необхідне оновлення секретного ключа JWT з метою підвищення безпеки системи та запобігання несанкціонованому доступу; резервне копіювання файлу бази даних для захисту від втрати даних у разі апаратного відмови. Такі роботи доцільно планувати на період низької активності виробництва, коли простій системи не вплине на виробничий процес. На практиці підприємства часто здійснюють такі операції на вихідних або під час планового технічного обслуговування інших систем.

При проведенні капітального ремонту приміщення або заміни інженерних систем рекомендується виконати повне тестування системи моніторингу для визначення можливих проблем з сигналом Wi-Fi або механічними пошкодженнями обладнання. Якщо система працює в агресивному середовищі (високої вологості, температури або з хімічними парами), варто розглянути питання про більш частіше профілактичного обслуговування.

У разі виходу з ладу апаратного вузла ESP32 його заміна не потребує складних змін у серверному програмному забезпеченні: достатньо прошити новий мікроконтролер з тими самими константами WIFI_SSID, WIFI_PASSWORD та SERVER_URL.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		55

Ідентифікатор пристрою deviceId у прошивці рекомендується зберігати незмінним для збереження безперервності хронологічних даних у БД та правильної роботи аналітичних звітів. Запасні мікроконтролери та датчики варто тримати під рукою для мінімізації часу простою системи та забезпечення бездоганної роботи складського господарства. Спеціалісти рекомендують мати в наявності принаймні один резервний набір обладнання для кожного приміщення під моніторингом.

Моніторинг та аналіз найпоширеніших проблем показує, що в більшості випадків відмови пов'язані з проблемами Wi-Fi зв'язку, а не з дефектами самого датчика. У таких ситуаціях достатньо перезагрузити роутер або переставити точку доступу на більш вдале місце. Для складських приміщень великої площі рекомендується встановити додаткові точки доступу для забезпечення стійкого сигналу у віддалених куточках.

Економічна ефективність впровадження

Порівняно з аналогічними промисловими рішеннями класу Siemens, розроблена система забезпечує суттєво нижчу вартість впровадження.

Повна вартість апаратної частини одного вузла моніторингу (ESP32 + DHT22 + корпус + блок живлення) на українському ринку складає близько 400-500 гривень. Серверне програмне забезпечення розповсюджується безкоштовно разом з вихідним кодом, що дозволяє підприємствам модифікувати його під свої потреби без додаткових ліцензійних платежів. За умови наявності власного сервера або одноплатного комп'ютера (Raspberry Pi 4 коштує близько 2500-3000 гривень на вітчизняному ринку) загальна вартість системи для одного приміщення з трьома точками вимірювання складає менше 3500 гривень.

Таке рішення доступне навіть для малих та середніх підприємств, які раніше не могли дозволити собі системи централізованого моніторингу мікроклімату через високу вартість промислових аналогів, ціни на які сягають десятків тисяч гривень.

Система забезпечує скорочення витрат на електроенергію за рахунок автоматичного переходу у "Нічний економ" режим у неробочі години. За розрахунками, зниження цільової температури з 20 °C до 15 °C у неробочі 14 годин на добу при потужності системи опалення 2 кВт дозволяє заощадити до 30 % електроенергії на опалення складу у холодну пору року.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

При середній вартості електроенергії для підприємств у розмірі 6-8 гривень за кВт·год, економія на одному складському приміщенні може сягати 15-20 тисяч гривень щорічно. У теплу пору року система також знижує витрати на кондиціювання повітря за рахунок точного контролю та попередження перегріву приміщень. При правильному налаштуванні ПД-регулятора система працює в режимі, близькому до оптимального енергоспоживання, без необхідності частих включень та вимикань обладнання, що також продовжує його термін служби.

Окупність системи у виробничих умовах зазвичай досягається протягом першого року експлуатації за рахунок економії на енергоносіях та запобігання пошкодження продукції через недотримання умов зберігання, що могла б привести до значних матеріальних збитків. Для складу лакофарбових матеріалів втрата партії товару через неправильне зберігання може обійтись у десятки тисяч гривень, тоді як вартість системи становить менше 3500 гривень. Крім того, система генерує дані для аналізу, які допомагають виявити неефективність використання енергоресурсів та визначити додаткові резерви для економії. Багато українських підприємств відмічають, що після впровадження системи моніторингу вони змогли скоротити витрати на енергію на 20-35 % у першому році роботи, що при масштабах великого складу дорівнює десяткам тисяч гривень щорічно. Кожне додаткове приміщення під моніторингом вимагає мінімальних капітальних вкладень (всього 1500-2000 гривень на вузол без сервера), що робить розширення системи економічно вигідним навіть для невеликих підприємств.

Крім прямої економії на енергії, система забезпечує непрямі вигоди у вигляді покращення якості продукції, зменшення страхових виплат за браковану продукцію та підвищення репутації підприємства на ринку за рахунок документованого дотримання умов зберігання. Наявність детальних логів дозволяє компанії виконувати правові вимоги щодо документування умов зберігання й транспортування лікарських препаратів, харчових продуктів та інших чутливих до клімату товарів, що особливо важливо для компаній, які експортують продукцію та потребують сертифікацій ISO або інших міжнародних стандартів.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		57

6 ОСНОВНІ ВИСНОВКИ

У процесі виконання бакалаврської роботи було розроблено програмне забезпечення системи інтелектуального керування мікрокліматом промислових приміщень на основі аналізу сенсорних даних та оптимізаційних алгоритмів. Результати роботи дозволяють зробити такі висновки.

1. Проведено огляд та порівняльний аналіз існуючих систем автоматизації мікроклімату: промислової SCADA-платформи Siemens SIMATIC S7/WinCC та DIY-рішень на базі Arduino/ESP32 з протоколом MQTT і системою візуалізації Grafana. Встановлено, що перший клас рішень є економічно недоступним для малих і середніх підприємств (вартість понад 50 000 доларів США), а другий не забезпечує необхідного рівня безпеки, структурованості та готовності до промислового застосування. Обґрунтовано вибір технологічного стеку розробки: ASP.NET Core 8, C#, SQLite, Entity Framework Core, SignalR, Bootstrap 5, Chart.js, ESP32, DHT22.

2. Розроблено вбудоване програмне забезпечення (firmware) мікроконтролера ESP32 мовою C++ з використанням фреймворку Arduino. Прошивка забезпечує циклічне зчитування температури та відносної вологості з датчика DHT22 з інтервалом 5 секунд, автоматичне відновлення Wi-Fi-з'єднання при його втраті та передачу телеметрії на сервер у форматі JSON через протокол HTTP POST.

3. Спроектовано та реалізовано серверну частину системи на платформі ASP.NET Core 8. Розроблено модель даних з чотирма сутностями (TelemetryRecord, User, Profile, SystemLog) та виконано міграцію бази даних SQLite через Entity Framework Core. Реалізовано REST API для прийому телеметрії, автентифікації та управління налаштуваннями. Впроваджено систему авторизації на основі JWT-токенів з розмежуванням ролей Operator та Engineer та захистом паролів алгоритмом BCrypt.

4. Розроблено математичний алгоритм ПІД-регулювання у вигляді класу PidController з реалізацією захисту від integral windup та налаштовуваними коефіцієнтами K_p , K_i , K_d .

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

Реалізовано цифровий двійник середовища - клас EnvironmentSimulator, що моделює теплові процеси у приміщенні на основі спрощеної математичної моделі теплового балансу. Розроблено фоновий сервіс ClimateControlService, що інтегрує ПІД-регулятор, симулятор та логіку інтелектуальних профілів мікроклімату в єдиний керуючий цикл.

5. Розроблено веб-інтерфейс оператора з візуалізацією даних у режимі реального часу. Реалізовано передачу телеметрії через SignalR-хаб із затримкою не більше 3 секунд. Побудовано динамічні графіки температури, вологості та стану виконавчих пристроїв засобами Chart.js. Реалізовано панель інженера для налаштування параметрів ПІД-регулятора та профілів мікроклімату, а також механізм відображення аварійних сповіщень при втраті зв'язку з ESP32.

6. Реалізовано модуль генерації аналітичних звітів у форматах PDF (з використанням бібліотеки QuestPDF) та CSV для надання даних за довільний часовий діапазон. Кнопки завантаження звітів інтегровано у веб-інтерфейс у вигляді модального вікна вибору дат.

7. Проведено модульне тестування критичних компонентів системи з використанням фреймворку xUnit. Усі чотири тести пройдено успішно, що підтверджує коректність реалізації ПІД-алгоритму та теплової моделі симулятора.

Практична цінність розробки полягає у тому, що система може бути впроваджена на реальному складському підприємстві при загальній вартості апаратної частини менше 100 доларів США. Система забезпечує автономний цілодобовий контроль мікроклімату, скорочення енерговитрат до 30 % у неробочі години, формування звітності для служб якості та захищений багатокористувацький доступ через стандартний веб-браузер без встановлення додаткового ПЗ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO 31000:2018. Risk management - Guidelines. International Organization for Standardization, 2018. 16 p.
2. Industry 4.0 / Bundesministerium für Bildung und Forschung. Berlin, 2019. URL: <https://www.bmbf.de/bmbf/de/forschung/digitale-wirtschaft-und-gesellschaft/industrie-40> (дата звернення: 10.03.2026).
3. Siemens AG. SIMATIC S7-1500 System Manual. Munich: Siemens AG, 2022. 512 p.
4. Siemens AG. WinCC V7.5 SP2 Configuration Manual. Munich: Siemens AG, 2021. 648 p.
5. Хайне Д. MQTT: The Standard for IoT Messaging. HiveMQ, 2021. URL: <https://www.hivemq.com/mqtt/> (дата звернення: 12.03.2026).
6. Espressif Systems. ESP32 Technical Reference Manual. Version 5.1. Shanghai, 2023. 1134 p.
7. Aosong Electronics. AM2302 / DHT22 Product Manual. Guangzhou, 2022. 14 p.
8. TechEmpower Web Framework Benchmarks. Round 22. URL: <https://www.techempower.com/benchmarks/> (дата звернення: 15.03.2026).
9. SQLite Documentation. About SQLite. URL: <https://www.sqlite.org/about.html> (дата звернення: 14.03.2026).
10. Microsoft. ASP.NET Core SignalR documentation. URL: <https://learn.microsoft.com/aspnet/core/signalr/introduction> (дата звернення: 16.03.2026).
11. Microsoft. .NET 8 - What's new. URL: <https://learn.microsoft.com/dotnet/core/whats-new/dotnet-8/overview> (дата звернення: 17.03.2026).
12. Microsoft. Entity Framework Core documentation. URL: <https://learn.microsoft.com/ef/core/> (дата звернення: 18.03.2026).
13. Smith J. Entity Framework Core in Action. 2nd ed. Manning Publications, 2021. 600 p.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

14. Microsoft. JSON Web Tokens in ASP.NET Core. URL: <https://learn.microsoft.com/aspnet/core/security/authentication/jwt-authn> (дата звернення: 19.03.2026).

15. Jones M., Bradley J., Sakimura N. RFC 7519: JSON Web Token (JWT). Internet Engineering Task Force, 2015. 30 p.

16. Provos N., Mazières D. A Future-Adaptable Password Scheme. USENIX Annual Technical Conference, 1999. P. 81-91.

17. Microsoft. Background tasks with hosted services in ASP.NET Core. URL: <https://learn.microsoft.com/aspnet/core/fundamentals/host/hosted-services> (дата звернення: 20.03.2026).

18. Microsoft. Dependency injection in ASP.NET Core. URL: <https://learn.microsoft.com/aspnet/core/fundamentals/dependency-injection> (дата звернення: 20.03.2026).

19. Seemann M. Dependency Injection Principles, Practices, and Patterns. Manning Publications, 2019. 584 p.

20. xUnit.net Documentation. URL: <https://xunit.net/docs/getting-started/netcore/cmdline> (дата звернення: 21.03.2026).

21. Åström K. J., Hägglund T. PID Controllers: Theory, Design, and Tuning. 2nd ed. ISA, 1995. 354 p.

22. Ziegler J. G., Nichols N. B. Optimum Settings for Automatic Controllers. Trans. ASME, 1942. Vol. 64. P. 759-768.

23. Ogata K. Modern Control Engineering. 5th ed. Prentice Hall, 2010. 912 p.

24. Бесекерський В. А., Попов Є. П. Теорія систем автоматичного регулювання. 4-те вид. СПб.: Професія, 2003. 752 с.

25. Ротач В. Я. Теорія автоматичного управління. М.: MEI, 2008. 400 с.

26. Normey-Rico J. E., Camacho E. F. Control of Dead-time Processes. Springer, 2007. 462 p.

27. Skogestad S. Simple analytic rules for model reduction and PID controller tuning. Journal of Process Control. 2003. Vol. 13(4). P. 291-309.

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		61

28. Вітенько Т. М. Теорія автоматичного керування. Тернопіль: ТНТУ, 2015. 248 с.
29. Ладанюк А. П. Основи автоматики і автоматизація виробничих процесів. К.: Аграрна освіта, 2001. 224 с.
30. Клименко В. А. Мікропроцесорні системи керування. Харків: НТУ «ХПІ», 2017. 316 с.
31. Kolban N. Kolban's Book on ESP32. Texas, 2020. 508 p. URL: <https://leanpub.com/kolban-ESP32> (дата звернення: 22.03.2026).
32. Adafruit Industries. DHT Sensor Library. GitHub. URL: <https://github.com/adafruit/DHT-sensor-library> (дата звернення: 22.03.2026).
33. Arduino LLC. Arduino Reference. URL: <https://www.arduino.cc/reference/en/> (дата звернення: 23.03.2026).
34. Schwartz M. Internet of Things with ESP8266. Packt Publishing, 2016. 190 p.
35. Raj P., Raman A. The Internet of Things: Enabling Technologies, Platforms, and Use Cases. CRC Press, 2017. 388 p.
36. Lee E. A., Seshia S. A. Introduction to Embedded Systems: A Cyber-Physical Systems Approach. 2nd ed. MIT Press, 2017. 590 p.
37. White E. Making Embedded Systems: Design Patterns for Great Software. O'Reilly Media, 2011. 330 p.
38. Цимбалюк В. І. Мікроконтролерні системи в задачах автоматизації. К.: КПІ, 2019. 280 с.
39. Мелехін В. Ф. Мікропроцесорна техніка. М.: Академія, 2013. 464 с.
40. Бойченко С. В. Вбудовані системи реального часу на базі RTOS. Харків: НТМТ, 2018. 192 с.
41. Frisbie M. Bootstrap 5 Quick Start. San Bernardino, 2022. 302 p.
42. Chart.js Documentation. Getting Started. URL: <https://www.chartjs.org/docs/latest/getting-started/> (дата звернення: 24.03.2026).
43. QuestPDF Documentation. URL: <https://www.questpdf.com/documentation/> (дата звернення: 24.03.2026).

44. OWASP Foundation. OWASP Top Ten 2021. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 25.03.2026).
45. RFC 6749: The OAuth 2.0 Authorization Framework / Ed. Hardt D. IETF, 2012. 76 р.
46. ДСТУ ISO/IEC 27001:2015. Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. К.: ДП «УкрНДНЦ», 2016. 42 с.
47. ДСТУ ISO 5807:2016. Технології інформаційні. Схеми алгоритмів, програм, даних та систем. К.: ДП «УкрНДНЦ», 2016. 24 с.
48. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. К.: ДП «УкрНДНЦ», 2016. 17 с.
49. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures. PhD thesis. UC Irvine, 2000. 162 p.
50. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. O'Reilly Media, 2021. 616 p.

КБПЗ-2026

					ВКРБ-122.26.0017.00.00.ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		