

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

ФАКУЛЬТЕТ ПРОГРАМНИХ СИСТЕМ ТА ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___» червня 2026 р.

**Дипломний проєкт
на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Системне програмування та спеціалізовані комп'ютерні системи»**

по спеціальності **123 «Комп'ютерна інженерія»**
на тему: «Інформаційна система моніторингу та прогнозування температури
середовища».

Виконав:

Студент IV курсу, групи КВ-23

Литвин Максим Ігорович _____

Керівник:

доцент кафедри СПіСКС, к.т.н., доцент

Мартінова Оксана Петрівна _____

Консультант з нормоконтролю:

доцент кафедри СПіСКС, к.т.н. доцент

Клятченко Ярослав Михайлович _____

Рецензент:

Ст. викладач кафедри ПЗКС, к.т.н.

Хічко Яна Володимирівна _____

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

Студент _____

Київ - 2026 року

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

ФАКУЛЬТЕТ ПРОГРАМНИХ СИСТЕМ ТА ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Освітньо-професійна програма

«Системне програмування та спеціалізовані
комп'ютерні системи»

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

" ____ " _____ 2026 р.

ЗАВДАННЯ

на дипломний проєкт студента

Литвина Максима Ігоровича

1. Тема проєкту «Інформаційна система моніторингу та прогнозування температури середовища», керівник проєкту Мартинова Оксана Петрівна, доцент кафедри СПіСКС, к.т.н., доцент, затверджені наказом по університету від «28» травня 2026 р. № 1984-с.
2. Термін подання студентом проєкту «10» червня 2026 р.
3. Вихідні дані до проєкту: див. Технічне завдання.
4. Зміст пояснювальної записки:
 - 1) Огляд існуючих рішень моніторингу температури та обґрунтування актуальності розробки;

- 2) Аналіз і вибір засобів реалізації – мови програмування, бібліотек, апаратних компонентів;
- 3) Опис розробленої інформаційної системи: апаратної частини, програмного забезпечення, алгоритмів прогнозування та виявлення аномалій;
- 4) Експериментальне дослідження точності прогнозу та аналіз отриманих результатів.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій, тощо:

- схема структурна системи;
- схема структурна програмного забезпечення;
- схема алгоритму виявлення аномалій;
- схема електрична принципова;
- перелік елементів.

6. Консультанти розділів проекту:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Клятченко Я.М., доцент		

7. Дата видачі завдання «06» лютого 2026 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	07.02.2026
2.	Розроблення та узгодження технічного завдання	10.01.2026
3.	Аналіз існуючих рішень	05.02.2026
4.	Підготовка матеріалів першого розділу дипломного проекту	15.02.2026
5.	Підготовка матеріалів другого розділу дипломного проекту	12.03.2026

6.	Підготовка матеріалів третього розділу дипломного проекту	14.04.2026
7.	Підготовка матеріалів четвертого розділу дипломного проекту	05.05.2026
8.	Підготовка графічної частини дипломного проекту	17.05.2026
9.	Оформлення документації дипломного проекту	22.05.2026
10.	Попередній огляд матеріалів диплому на кафедрі	01.06.2026

Студент

Максим ЛИТВИН

(підпис)

Керівник проекту

Оксана МАРТИНОВА

(підпис)

АНОТАЦІЯ

Бакалаврський дипломний проєкт включає пояснювальну записку (85 с., 20 рис., 2 табл., 4 додатків, ** слайдів презентації).

Мета розробки – створити інформаційну систему, що поєднує апаратний модуль збору температурних даних і програмне забезпечення для їх зберігання, короткострокового прогнозування та автоматичного виявлення аномальних відхилень з оперативним інформуванням користувача.

Об'єкт розробки – інформаційна система моніторингу та прогнозування температури середовища, яка дозволяє вести безперервний збір, зберігання, аналіз та прогнозування температурних даних.

Інформаційна система дозволяє: здійснювати безперервний збір даних з температурного датчика; передавати дані дротовим та бездротовим каналами зв'язку; зберігати історію вимірювань у реляційній базі даних; виконувати короткостроковий прогноз температури; автоматично виявляти аномальні зміни показників; інформувати користувача про аномалії через месенджер Telegram. Передбачена можливість розширення функціональності завдяки відкритій архітектурі на основі шаблону проєктування Strategy. У процесі розробки були використані мова програмування Java, платформа JavaFX, мікроконтролерна платформа Arduino UNO з датчиком DHT11, технологія бездротового зв'язку Bluetooth. В якості бази даних використовувалась PostgreSQL.

В ході розробки:

- проведено аналіз існуючих систем моніторингу температури середовища;
- сформульовані вимоги до інформаційної системи моніторингу та прогнозування температури середовища;
- здійснено вибір програмних та апаратних засобів реалізації;
- розроблена структура інформаційної системи моніторингу та прогнозування температури середовища;

- розроблено користувацький додаток для візуалізації та аналізу температурних даних;
- розроблено алгоритми короткострокового прогнозування температури та виявлення аномальних значень;
- розроблено програмне забезпечення мікроконтролера для роботи з температурним датчиком та модулем бездротового зв'язку.

Використання цієї системи дозволить забезпечити доступний контроль температурного режиму середовища та своєчасне виявлення аномальних змін у побутових і навчальних умовах.

Ключові слова: МОНІТОРИНГ, ТЕМПЕРАТУРА, ПРОГНОЗУВАННЯ, АНОМАЛІЇ, ARDUINO, JAVA, POSTGRESQL, TELEGRAM, STRATEGY

ABSTRACT

The bachelor's thesis includes an explanatory note (85 pages, 20 figures, 2 tables, 4 appendices, ** presentation slides).

The goal of the project is to create an information system that combines a hardware module for collecting temperature data with software for storing that data, short-term forecasting, and automatic detection of abnormal deviations with real-time user notification.

The subject of the development is an information system for monitoring and forecasting ambient temperature, which enables continuous collection, storage, analysis, and forecasting of temperature data.

The information system enables: continuous data collection from a temperature sensor; data transmission via wired and wireless communication channels; storage of measurement history in a relational database; short-term temperature forecasting; automatic detection of abnormal changes in readings; and notification of the user about anomalies via the Telegram messenger. The system is designed to allow for functionality expansion thanks to its open architecture based on the Strategy design pattern. The development process utilized the Java programming language, the JavaFX platform, the Arduino UNO microcontroller platform with a DHT11 sensor, and Bluetooth wireless technology. PostgreSQL was used as the database.

During the development process:

- an analysis of existing ambient temperature monitoring systems was conducted;
- requirements for the ambient temperature monitoring and forecasting information system were formulated;
- software and hardware for implementation were selected;
- the structure of the ambient temperature monitoring and forecasting information system was developed;

- a user application for visualizing and analyzing temperature data was developed;
- algorithms for short-term temperature forecasting and detection of abnormal values were developed;
- microcontroller software for working with a temperature sensor and a wireless communication module was developed.

The use of this system will ensure accessible monitoring of ambient temperature conditions and timely detection of abnormal changes in residential and educational settings.

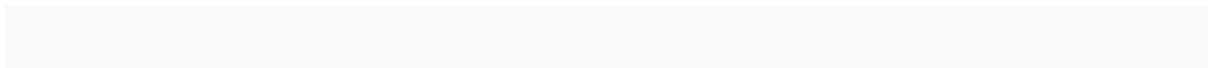
Keywords: MONITORING, TEMPERATURE, FORECASTING, ANOMALIES, ARDUINO, JAVA, POSTGRESQL, TELEGRAM, STRATEGY

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кіл-ть аркушів	№ прим.	Примітки
	A4	ІАЛЦ.467200.002 ТЗ	Інформаційна система моніторингу та прогнозування середовища Технічне завдання	4		
	A4	ІАЛЦ.467200.003 ТП	Інформаційна система моніторингу та прогнозування середовища Відомість технічного проєкту	2		
	A4	ІАЛЦ.467200.004 ПЗ	Інформаційна система моніторингу та прогнозування середовища Пояснювальна записка	85		
			ІАЛЦ.467200.001 ОА			
Змін.	Арх.	№ докум.	Підпис			
Розробив	Литвин М. І.			Інформаційна система моніторингу та прогнозування температури середовища		
Перевірив	Мартинова О.П.					
Консуьт.						
Н. контроль	Клятченко Я.М.					
Зав. каф.	Романкевич В.О.					
				Літ.	Аркуш	Аркушів
					1	2
				КПП ім.Ігоря Сікорського, ФПСІМ, КВ-23		
				Опис альбому		

[illegible]

ЗМІСТ

1.	НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ _____	2
2.	ПІДСТАВА ДЛЯ РОЗРОБКИ _____	2
3.	ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ _____	2
4.	ДЖЕРЕЛА РОЗРОБКИ _____	2
5.	ТЕХНІЧНІ ВИМОГИ _____	2
5.1.	Вимоги до програмного продукту, що розробляється _____	3
5.2.	Вимоги до апаратного забезпечення _____	3
5.3.	Вимоги до програмного та апаратного забезпечення користувача _____	3
6.	ЕТАПИ РОЗРОБКИ _____	4



					ІАЛЦ. 467200.002 ТЗ			
Зм	Лист	№ докум.	Підп.	Дата				
Розроб.		Литвин М.І.			Інформаційна система моніторингу та прогнозування температури середовища Технічне завдання	Лім.	Лист	Листів
Перев.		Мартінова О.П.					1	4
						НТУУ «КПІ ім. Ігоря Сікорського», ФПСМ, КВ-23		
Н. контр.		Клятченко						
Затв.		Романкевич						

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Інформаційна система моніторингу та прогнозування температури середовища».

Галузь застосування: безперервний моніторинг, зберігання, аналіз та короткострокове прогнозування температури середовища у побутових, навчальних та виробничих умовах.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення програмно-апаратного комплексу для безперервного збору, зберігання, прогнозування та аналізу температурних даних із автоматичним сповіщенням користувача про аномальні зміни.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

					ІАЛЦ.467200.002 ТЗ	Лист
Зм	Лист	№ докум.	Підп.	Дата		2

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- збір даних з температурного датчика в реальному часі;
- підтримка дротового та бездротового каналів передачі даних;
- зберігання історії вимірювань у реляційній базі даних;
- візуалізація температурних даних у вигляді графіка;
- короткострокове прогнозування температури;
- автоматичне виявлення аномальних змін показників;
- сповіщення користувача про аномалії через месенджер Telegram;
- експорт зібраних даних у формат CSV;
- можливість розширення функціональності завдяки відкритій архітектурі.

5.2 Вимоги до апаратного забезпечення

- мікроконтролер: Arduino UNO (ATmega328P);
- температурний датчик: DHT11;
- модуль бездротового зв'язку: Bluetooth HC-05;
- наявність персонального комп'ютера для роботи серверної частини.

5.3 Вимоги до програмного та апаратного забезпечення користувача

- операційна система Windows 10 або новіша;
- встановлене середовище виконання Java (JDK 21 або новіше);
- система керування базами даних PostgreSQL;
- наявність USB-порту або модуля Bluetooth;
- оперативна пам'ять: не менше 4 Гб.

6 ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	07.02.2026
2.	Розроблення та узгодження технічного завдання	10.01.2026
3.	Аналіз існуючих рішень	05.02.2026
4.	Підготовка матеріалів першого розділу дипломного проекту	15.02.2026
5.	Підготовка матеріалів другого розділу дипломного проекту	12.03.2026
6.	Підготовка матеріалів третього розділу дипломного проекту	14.04.2026
7.	Підготовка матеріалів четвертого розділу дипломного проекту	05.05.2026
8.	Підготовка графічної частини дипломного проекту	17.05.2026
9.	Оформлення документації дипломного проекту	22.05.2026
10.	Попередній огляд матеріалів диплому на кафедрі	01.06.2026

[illegible]

[illegible]

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ	3
ВСТУП	4
1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ	6
1.1 Обґрунтування актуальності	6
1.2 Огляд сучасних систем моніторингу температури	8
1.3 Переваги та недоліки існуючих рішень	15
1.4 Обґрунтування теми дипломного проєкту	18
1.5 Висновки до розділу	19
2. АНАЛІЗ ТА ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ	20
2.1 Обґрунтування вибору архітектури системи	20
2.2 Обґрунтування вибору мови програмування	23
2.3 Опис програмного інструментарію	25
2.4 Опис апаратного інструментарію	28
2.5 Аналіз використаних бібліотек	34
2.6 Зовнішні сервіси: Telegram Bot API	39
2.7 Висновок до розділу 2	41
3. ОПИС РОЗРОБЛЕНОЇ СИСТЕМИ	43
3.1 Загальна архітектура системи	43
3.2 Апаратна частина та схема підключення	45
3.3 Структура програмного забезпечення	48

					ІАЛЦ. 467200.004 ПЗ			
Зм	Лист	№ докум.	Підп.	Дата	Інформаційна система моніторингу та прогнозування температури середовища Пояснювальна записка	Лім.	Лист	Листів
Розроб.		Литвин М.І.						
Перев.		Мартінова О.П.					1	85
Н. контр.		Клятченко				НТУУ «КПІ ім. Ігоря Сікорського», ФПСМ, КВ-23		
Затв.		Романкевич						

3.4 Алгоритми прогнозування температури _____	63
3.5 Конфігурація системи _____	70
3.6 Висновок _____	71
4. ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ СИСТЕМИ _____	73
ВИСНОВКИ _____	81
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ _____	84

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

- ІАЛЦ. 467200.005 Д1. Загальна система проєкту. Структурна схема.
- ІАЛЦ. 467200.006 Е3. Загальна структура електронної схеми.
Електронна принципова схема.
- ІАЛЦ. 467200.007 Д2. Загальна структура програмного забезпечення.
Структура схема.
- ІАЛЦ. 467200.008 Д3. Виявлення аномалій. Схема алгоритму.

Додаток 2. Презентація

Додаток 3. Список публікацій за темою дипломного проєкту

					ІАЛЦ.467200.004 ПЗ	Лист
						2
Зм	Лист	№ докум.	Підп.	Дата		

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

СКБД – система керування базами даних

API (Application Programming Interface) – програмний інтерфейс застосунку

DIY (Do It Yourself) – категорія пристроїв для самостійного складання

GPT (Generative Pre-trained Transformer) – генеративна попередньо натренована модель

HACCP (Hazard Analysis and Critical Control Points) – система аналізу ризиків та критичних контрольних точок

I2C (Inter-Integrated Circuit) – послідовний інтерфейс міжмікросхемного зв'язку

IoT (Internet of Things) – Інтернет речей

LCD (Liquid Crystal Display) – рідкокристалічний дисплей

LLM (Large Language Model) – велика мовна модель

MAE (Mean Absolute Error) – середня абсолютна помилка

RMSE (Root Mean Square Error) – корінь з середньоквадратичної помилки

UART (Universal Asynchronous Receiver-Transmitter) – універсальний асинхронний приймач-передавач

ВСТУП

Контроль температури середовища належить до тих задач, що супроводжують людську діяльність у найрізноманітніших сферах – від побутового комфорту до підтримки роботи дата-центрів і дотримання санітарних норм у харчовій промисловості. На перший погляд, задача виглядає простою: достатньо знати поточне значення, щоб ухвалити рішення про увімкнення обігрівача, відкриття вікна чи виклик технічної служби. На практиці ж саме «поточне значення» є лише вершиною айсберга – за ним стоять питання про те, як швидко температура змінюється, чи відповідає її поведінка очікуваній динаміці, чи не наближається середовище до критичних меж. Сама по собі цифра на дисплеї відповіді на ці питання не дає.

Сучасний ринок засобів моніторингу температури характеризується вираженою поляризацією. На одному полюсі представлені комерційні рішення з широким функціоналом, але закритою архітектурою та високою вартістю – користувач отримує готовий продукт, але втрачає можливість впливати на його роботу. На іншому полюсі – бюджетні побутові пристрої, що обмежуються відображенням поточного значення, без зберігання історії, прогнозу чи сповіщень. Між цими крайнощами існує категорія потужних відкритих платформ, які теоретично можуть закрити всі потреби користувача, але вимагають таких витрат часу і технічних знань, що залишаються доступними лише ентузіастам автоматизації. Незаповненою залишається ніша рішення, що поєднує доступну ціну, розширений функціонал та відкритість архітектури.

Метою даного дипломного проєкту є розробка інформаційної системи моніторингу та прогнозування температури середовища, що відповідає вимогам цієї незаповненої ніші. Розроблений комплекс об'єднує мікроконтролерну платформу Arduino з температурним датчиком, реляційну базу даних для зберігання історії вимірювань, десктопний додаток для візуалізації та аналізу даних, а також канал сповіщень через месенджер

					ІАЛЦ.467200.004 ПЗ	Лист 4
Зм	Лист	№ докум.	Підп.	Дата		

Telegram. Передбачені алгоритми короткострокового прогнозу температури за двома методами – ковзним середнім та лінійною регресією – і автоматичне виявлення аномальних значень на основі статистичного z-критерію. Орієнтовна вартість апаратної частини не перевищує тридцяти доларів США, що робить рішення доступним для широкого кола користувачів.

Особливістю розробленої системи є застосування шаблону проєктування Strategy для основних компонентів – джерел даних, алгоритмів прогнозування та каналів сповіщень. Такий підхід дозволяє користувачам з технічними навичками самостійно розширювати функціональність системи – підключати інші типи датчиків, додавати нові алгоритми прогнозу або інтегрувати додаткові канали взаємодії – без необхідності модифікації основного коду. Це принципово відрізняє розроблене рішення від закритих комерційних аналогів.

Об'єктом дослідження виступають процеси збору, обробки та аналізу температурних даних. Предметом – методи побудови програмно-апаратного комплексу для безперервного моніторингу температури з функціями прогнозування та виявлення аномалій. Експериментальна перевірка точності прогнозу проведена методом walk-forward валідації з обчисленням метрик MAE та RMSE на даних, зібраних з реального датчика впродовж тривалого спостереження.

Робота складається з чотирьох розділів. У першому проведено огляд існуючих рішень та обґрунтовано актуальність розробки. У другому здійснено аналіз і вибір програмних та апаратних засобів реалізації. У третьому описано проєктування та архітектуру системи. У четвертому наведено результати експериментального дослідження точності прогнозування.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1 Обґрунтування актуальності

У сучасних умовах задача безперервного моніторингу температури середовища набуває особливого значення. Температура є одним із ключових фізичних параметрів, що впливає на комфорт людини, безпеку експлуатації обладнання, ефективність технологічних процесів та збереження продукції. Зростання вимог до якості мікрокліматичного контролю, посилення регуляторних норм у харчовій та фармацевтичній галузях, а також поширення концепції «розумного будинку» формують стійкий попит на доступні та надійні системи температурного моніторингу.

Серед основних сфер, де контроль температури є критично важливим, можна виділити такі:

Побут та домашній комфорт. У сучасному житті температурний режим безпосередньо впливає на самопочуття мешканців, ефективність роботи систем опалення та кондиціонування, енергоспоживання. Несвоєчасне виявлення відключення опалення в холодну пору року може призвести до промерзання комунікацій, а перегрів приміщення влітку – до погіршення стану здоров'я людей, особливо дітей та осіб похилого віку.

Зберігання харчових продуктів. Згідно з міжнародною системою НАССР (Hazard Analysis and Critical Control Points), яка є обов'язковою для підприємств харчової промисловості країн Європейського Союзу та України, безперервний контроль температури в холодильниках і морозильних камерах є ключовою умовою безпечного зберігання продуктів. Порушення температурного режиму навіть на короткий час може призвести до псування продукції та становити загрозу здоров'ю споживачів.

					ІАЛЦ.467200.004 ІІЗ	Лист
Зм	Лист	№ докум.	Підп.	Дата		6

Сільське господарство. У тепличних господарствах температура є визначальним фактором для росту й розвитку рослин. Відхилення від заданих режимів – особливо при відмові систем обігріву в зимовий період – може призвести до значних економічних втрат через загибель врожаю. Своєчасне отримання сповіщень про критичні зміни дозволяє уникнути таких втрат.

Технічна безпека приміщень з електрообладнанням. У серверних кімнатах, електрощитових та приміщеннях зі складною електронікою перегрів є однією з основних причин аварій та пожеж. Локальне підвищення температури часто сигналізує про несправності задовго до виникнення критичної ситуації, що робить температурний моніторинг важливим інструментом превентивної безпеки.

Дата-центри та обчислювальна інфраструктура. У серверних залах температура безпосередньо впливає на надійність роботи обладнання та термін його експлуатації. За даними галузевих досліджень, кожне підвищення температури на 10 °C понад рекомендований діапазон скорочує термін служби електронних компонентів приблизно вдвічі. Тому моніторинг температури в дата-центрах є невід'ємною частиною їхньої експлуатації.

Незважаючи на значну кількість представлених на ринку рішень, переважна більшість доступних систем має суттєві обмеження. Комерційні рішення, як правило, мають високу вартість, працюють у закритих екосистемах виробника та не передбачають можливості розширення функціональності користувачем. Дешеві побутові пристрої, зі свого боку, обмежуються лише відображенням поточного значення та не забезпечують зберігання історичних даних, прогнозування або сповіщень. Окремою категорією є професійні платформи на кшталт Home Assistant, які надають широкі можливості, але потребують значних технічних знань та окремої інфраструктури для розгортання.

Тому, актуальною є задача створення доступної інформаційної системи моніторингу температури, яка б поєднувала функціональність комерційних рішень (зберігання історії, прогнозування, віддалені сповіщення) із відкритістю архітектури та можливістю розширення під потреби конкретного користувача.

1.2 Огляд сучасних систем моніторингу температури

На ринку температурного моніторингу представлено значну кількість рішень, що відрізняються між собою функціональними можливостями, ціновою категорією, рівнем відкритості та сферою застосування. Для аналізу обрано чотири представники, що репрезентують основні сегменти ринку: преміальні комерційні рішення, відкриті програмні платформи, побутові метеостанції класичного формату та компактні IoT-датчики бюджетного класу.

1.2.1 Netatmo Weather Station

Netatmo Weather Station – комерційна метеостанція французької компанії Netatmo (заснована у 2011 році, з 2018 року входить до складу групи Legrand), орієнтована на користувачів сегменту «розумного дому». Пристрій складається з основного внутрішнього модуля та виносного зовнішнього датчика, з можливістю розширення додатковими модулями вимірювання вологості, рівня CO₂, шуму та опадів.

Основні характеристики Netatmo Weather Station:

- вимірювання температури з точністю $\pm 0,3$ °C у діапазоні від -40 °C до +65 °C;

					ІАЛЦ.467200.004 ІЗ	Лист
Зм	Лист	№ докум.	Підп.	Дата		8

- одночасний моніторинг кількох параметрів мікроклімату: температури, відносної вологості, атмосферного тиску, концентрації CO₂ у приміщенні;
- передача даних через Wi-Fi до власної хмарної платформи виробника;
- візуалізація даних через офіційний мобільний додаток для iOS та Android з функцією перегляду історичних значень;
- сповіщення про відхилення показників від заданих користувачем порогів;
- підтримка інтеграції зі сторонніми сервісами (Apple HomeKit, IFTTT, Amazon Alexa);
- автономна робота виносного модуля від двох батарейок типу AAA впродовж приблизно 18 місяців.

На рисунку 1.1 наведено зовнішній вигляд метеостанції Netatmo Weather Station.

					ІАЛЦ.467200.004 ІЗ	Лист
						9
Зм	Лист	№ докум.	Підп.	Дата		



Рисунок 1.1 – Зовнішній вигляд метеостанції Netatmo Weather Station

Орієнтовна вартість пристрою у роздрібному продажу становить близько 170 доларів США. До переваг рішення слід віднести високу точність вимірювань, привабливий дизайн, наявність зрілого мобільного додатку та можливість інтеграції з популярними платформами «розумного дому». Водночас система є повністю закритою: користувач не має доступу до сирих даних поза межами офіційного додатку, не може змінювати алгоритми обробки інформації та повністю залежить від хмарної інфраструктури виробника. У разі припинення підтримки сервісу з боку компанії функціональність пристрою істотно обмежується.

					ІАЛЦ.467200.004 ІЗ	Лист
						10
Зм	Лист	№ докум.	Підп.	Дата		

1.2.2 Home Assistant з модулями ESPHome

Home Assistant – відкрита програмна платформа для автоматизації «розумного будинку», розроблена міжнародною спільнотою з 2013 року. Платформа поширюється за ліцензією Apache 2.0 та підтримує понад 2000 інтеграцій з різноманітним обладнанням, у тому числі з самостійно зібраними температурними датчиками на базі мікроконтролерів ESP8266 та ESP32 через додаткове програмне забезпечення ESPHome.

У типовій конфігурації для моніторингу температури користувач самостійно збирає апаратну частину (мікроконтролер ESP32 та сенсор DHT22, BME280 або DS18B20), завантажує на нього прошивку ESPHome, після чого пристрій інтегрується до основного інстансу Home Assistant, розгорнутого на окремому одноплатному комп'ютері Raspberry Pi або домашньому сервері.

Основні можливості рішення:

- безкоштовне використання програмного забезпечення;
- широкі можливості налаштування – від вибору датчика та частоти опитування до складних сценаріїв автоматизації;
- зберігання історичних даних у вбудованій базі даних та візуалізація через веб-інтерфейс і мобільний додаток;
- підтримка великої кількості протоколів та інтеграцій (MQTT, Z-Wave, Zigbee, Modbus тощо);
- активна спільнота користувачів та розробників, що забезпечує постійний розвиток платформи.

Приклад інтерфейсу платформи Home Assistant з налаштованим моніторингом температури наведено на рисунку 1.2.

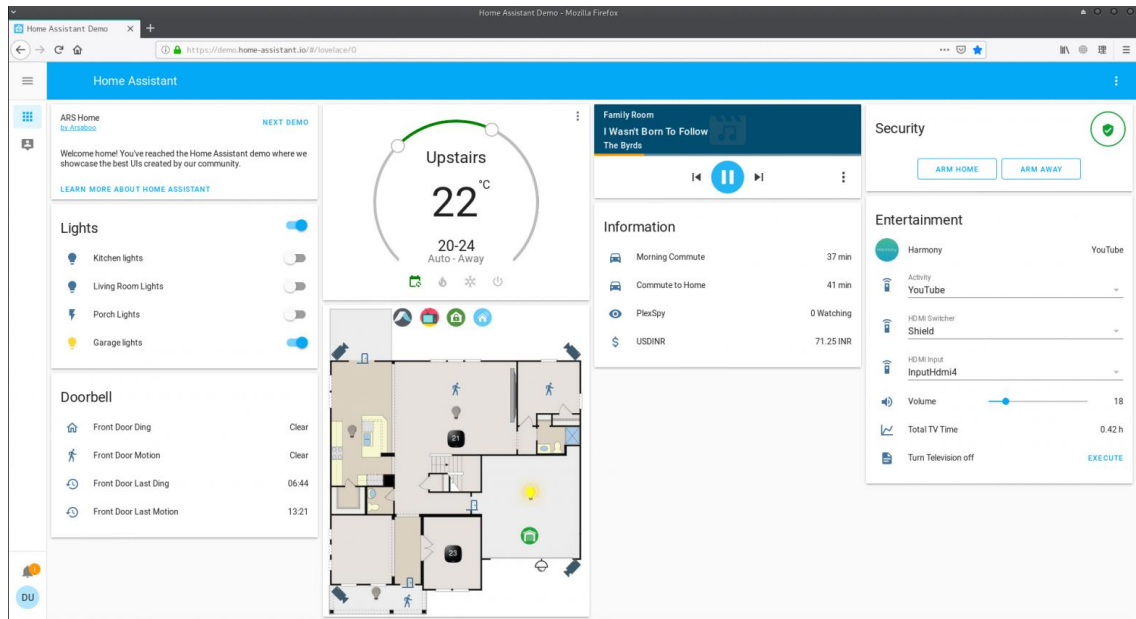


Рисунок 1.2 – Інтерфейс платформи Home Assistant

Незважаючи на безкоштовність програмної частини, повноцінне використання Home Assistant потребує значних початкових інвестицій часу та технічних навичок. Користувачеві необхідно самостійно обрати та зібрати апаратну платформу (вартість Raspberry Pi 4 з аксесуарами становить близько 100 доларів США), налаштувати операційну систему, ознайомитися з форматом конфігураційних файлів YAML, скласти власні сценарії автоматизації. Такий підхід орієнтований переважно на технічно підготовлених користувачів та ентузіастів автоматизації, але є малопридатним для широкої аудиторії, яка очікує отримати готове до використання рішення.

1.2.3 La Crosse Technology

La Crosse Technology – американський виробник побутових метеостанцій та приладів вимірювання параметрів навколишнього середовища, що працює на ринку з 1985 року. Компанія представляє широку лінійку моделей класичного формату – від простих кімнатних термогігрометрів до багатоканальних метеостанцій з прогнозом погоди.

Типова метеостанція цієї категорії (наприклад, модель La Crosse C85845 або S88907) містить:

- основний блок з рідкокристалічним дисплеєм для відображення показників;
- один або декілька виносних датчиків температури та вологості, що з'єднуються з основним блоком радіоканалом на частоті 433 МГц;
- вбудований барометр та алгоритм прогнозування погоди на найближчі 12–24 години на основі змін атмосферного тиску;
- годинник з функцією синхронізації за радіосигналом точного часу;
- сповіщення про вихід температури за межі встановлених порогів через звуковий сигнал.

Зовнішній вигляд типової метеостанції La Crosse Technology наведено на рисунку 1.3.



Рисунок 1.3 – Зовнішній вигляд метеостанції La Crosse Technology

Орієнтовна вартість моделей знаходиться у діапазоні 30–40 доларів США. До переваг цього типу рішень належать тривалий час автономної роботи від батарей, простота використання, інтуїтивно зрозумілий інтерфейс та відсутність необхідності додаткового налаштування. Водночас функціональність обмежується локальним відображенням показників: відсутня можливість віддаленого доступу до даних, історія вимірювань або не зберігається взагалі, або зберігається у вкрай обмеженому обсязі без можливості експорту, відсутні мобільні додатки, інтеграція з зовнішніми системами та можливість програмного розширення функціоналу.

1.2.4 Xiaomi Mijia Bluetooth Thermometer

Xiaomi Mijia Bluetooth Thermometer (модель LYWSD03MMC) – компактний термогігрометр китайського виробника Xiaomi, представлений у 2019 році в рамках екосистеми «розумного дому» Mi Home. Пристрій представляє собою невеликий квадратний модуль розміром приблизно $4,3 \times 4,3$ см з електронним чорнильним дисплеєм (e-ink), що відображає поточне значення температури та вологості.

Технічні характеристики:

- вимірювання температури у діапазоні від 0 °C до +60 °C з точністю $\pm 0,1$ °C;
- вимірювання відносної вологості у діапазоні 0–99 % з точністю ± 1 %;
- передача даних через Bluetooth Low Energy 4.2 до офіційного мобільного додатку Mi Home або сторонніх рішень;
- електронний чорнильний дисплей з низьким енергоспоживанням;
- автономна робота від однієї батарейки CR2032 впродовж приблизно одного року.

Зовнішній вигляд термогігрометра Xiaomi Mijia Bluetooth Thermometer наведено на рисунку 1.4.



Рисунок 1.4 – Зовнішній вигляд термогігрометра Xiaomi Mijia Bluetooth Thermometer

Вартість пристрою у роздрібному продажі складає 5–10 доларів США, що робить його одним із найдоступніших рішень на ринку. До переваг належать мала ціна, компактний розмір, низьке енергоспоживання та можливість використання значною кількістю одночасно у різних приміщеннях. Завдяки активній спільноті користувачів існують альтернативні прошивки (наприклад, ATC_MiThermometer), що розширюють базову функціональність та дозволяють інтегрувати пристрій з Home Assistant. Однак у штатній комплектації функціонал обмежується відображенням поточних значень на дисплеї та передачею даних до додатку Mi Home без серйозних можливостей аналітики, прогнозування чи налаштування сповіщень.

1.3 Переваги та недоліки існуючих рішень

Аналіз розглянутих систем дозволяє виділити ряд спільних переваг та обмежень для кожної з категорій представлених рішень.

Комерційні рішення преміального сегменту, представлені продуктом Netatmo Weather Station, забезпечують високу точність вимірювань, наявність

зрілих мобільних додатків та можливість інтеграції з популярними платформами «розумного дому». Водночас їм притаманні суттєві обмеження: висока вартість, повна залежність від хмарної інфраструктури виробника, неможливість самостійного розширення функціоналу та ризик втрати працездатності у випадку припинення підтримки сервісу.

Відкриті програмні платформи, такі як Home Assistant, надають практично необмежені можливості налаштування та інтеграції за рахунок відкритого коду та активної спільноти. Проте поріг входу для звичайного користувача залишається високим: необхідні навички адміністрування Linux-систем, розуміння принципів роботи мереж, знайомство з форматом конфігураційних файлів. Окрім того, повноцінне використання потребує окремої інфраструктури – одноплатного комп'ютера або домашнього сервера, що збільшує загальну вартість впровадження.

Побутові метеостанції класичного формату (La Crosse Technology та аналогічні) відрізняються простотою використання, тривалим часом автономної роботи та доступною ціною. Однак їх функціонал обмежується локальним відображенням показників: відсутні можливості віддаленого моніторингу, систематичного зберігання історії вимірювань, експорту даних або інтеграції з зовнішніми системами.

Компактні IoT-датчики бюджетного сегменту, такі як Xiaomi Miija Bluetooth Thermometer, поєднують доступну ціну з можливістю передачі даних на мобільний пристрій. У той же час у штатній конфігурації вони пропонують мінімальний функціонал, обмежений переглядом поточних значень, без серйозних можливостей аналітики, прогнозування чи гнучкого налаштування сповіщень.

Для систематизації результатів порівняння у таблиці 1.1 наведено зведену характеристику розглянутих систем за ключовими критеріями.

Таблиця 1.1 – Характеристика зведених систем за ключовими критеріями

	Netatmo	Home Assistant + ESPHome	La Crosse	Xiaomi Mijia
Орієнтована вартість	за фіксованим порогом	гнучко налаштовується	звуковий сигнал	відсутнє
Точність вимірювання	+0.3	залежність від датчика	+1.0	+0.1
Зберігання історії	хмара виробника	локально, без обмежень	відсутнє/обмежене	у мобільному додатку
Прогнозування	відсутнє	за допомогою інтеграції	примітивне (за тиском)	відсутнє
Виявлення аномалій	за фіксованим порогом	гнучко налаштовується	звуковий сигнал	відсутнє
Віддалені сповіщення	мобільний додаток, IFTTT	повна підтримка	відсутні	мобільний додаток
Відкритість коду	закрите пз	відкритий код	закрите пз	закрите пз
Можливість розширення	обмежена IFTTT	необмежена	відсутня	через сторонні прошивки
Складність налаштування	низька	висока	дуже низька	низька
Залежність від хмари	повна	відсутня	відсутня	часткова

Як видно з наведеної таблиці, на ринку відсутнє рішення, що одночасно поєднує доступну ціну, розширений функціонал (зберігання історії, прогнозування, віддалені сповіщення) та відкритість архітектури з можливістю самостійного розширення користувачем.

1.4 Обґрунтування теми дипломного проєкту

Проведений огляд показав, що сучасний ринок систем моніторингу температури характеризується вираженою поляризацією. З одного боку представлені комерційні рішення з широким функціоналом, але високою вартістю та закритою архітектурою. З іншого – дешеві побутові пристрої з обмеженими можливостями. Окрему категорію формують професійні відкриті платформи, які вимагають значних технічних знань для впровадження.

Таким чином, на ринку існує незаповнена ніша – доступне за ціною рішення, яке поєднує функціональність комерційних систем (зберігання історії, короткостроковий прогноз, виявлення аномалій, віддалені сповіщення) з відкритістю архітектури, властивою професійним платформам, та водночас не потребує від користувача значних технічних знань для базового використання.

Метою даного дипломного проєкту є розробка інформаційної системи моніторингу та прогнозування температури середовища, що відповідає наведеним вище вимогам. Орієнтовна вартість апаратної частини розробленого комплексу не перевищує 30 доларів США, що робить його доступним альтернативним рішенням порівняно з комерційними аналогами при збереженні розширеного функціоналу.

Ключовою архітектурною перевагою розробленої системи є застосування шаблону проєктування Strategy для основних компонентів – джерел даних, алгоритмів прогнозування та каналів сповіщень. Такий підхід дозволяє користувачам з технічними навичками самостійно розширювати функціональність системи: підключати інші типи температурних датчиків, додавати нові алгоритми прогнозу або інтегрувати додаткові канали взаємодії – без необхідності модифікації основного коду. Це принципово відрізняє розроблене рішення від закритих комерційних аналогів, де функціональність обмежена попередньо передбаченими виробником можливостями.

Розроблений комплекс орієнтований на широке коло користувачів: від побутового застосування (контроль температури в житлових приміщеннях, теплих, зонах зберігання продуктів) до використання як навчального стенда у закладах вищої освіти технічного профілю для демонстрації архітектурних рішень, алгоритмів обробки даних часових рядів та принципів побудови програмно-апаратних комплексів.

1.5 Висновки до розділу

У першому розділі проведено огляд та аналіз сучасних систем моніторингу температури середовища, представлених на ринку. Розглянуто чотири характерні рішення з різних сегментів: преміальну комерційну метеостанцію Netatmo Weather Station, відкриту програмну платформу Home Assistant з модулями ESPHome, побутову метеостанцію La Crosse Technology та компактний IoT-датчик Xiaomi Mijia Bluetooth Thermometer.

За результатами порівняльного аналізу встановлено, що ринок температурного моніторингу характеризується поляризацією між дорогими комерційними рішеннями з широким, але закритим функціоналом та дешевими побутовими пристроями з вкрай обмеженими можливостями. Незаповненою залишається ніша доступного відкритого рішення, що поєднує розширений функціонал (історія, прогноз, сповіщення) з можливістю самостійного розширення користувачем.

Обґрунтовано актуальність розробки інформаційної системи моніторингу та прогнозування температури середовища, орієнтовна вартість апаратної частини якої не перевищує 30 доларів США, з відкритою архітектурою на основі шаблону проєктування Strategy, що забезпечує природне розширення функціональності без модифікації основного коду. Сформульовано мету дипломного проєкту та визначено цільову аудиторію розроблюваного рішення.

					ІАЛЦ.467200.004 ПЗ	Лист 19
Зм	Лист	№ докум.	Підп.	Дата		

2. АНАЛІЗ ТА ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Обґрунтування вибору архітектури системи

При проектуванні програмних систем застосовуються різні архітектурні підходи, кожен з яких має власні переваги та обмеження. Для розроблюваної інформаційної системи моніторингу температури необхідно обрати архітектуру, що забезпечує надійну роботу в режимі реального часу, можливість зберігання та аналізу історичних даних, а також гнучкість для подальшого розширення функціональності.

Розглянемо основні архітектурні підходи, що могли б бути застосовані для побудови системи такого типу.

Монолітна архітектура. У межах монолітного підходу вся функціональність системи реалізується як єдина програма без чіткого розподілу на логічні шари. Перевагами такого підходу є простота реалізації та розгортання, а також відсутність накладних витрат на взаємодію між компонентами. Водночас монолітна архітектура має істотні недоліки: будь-яка зміна одного компонента вимагає перебудови всієї системи, а додавання нової функціональності (наприклад, нового датчика або алгоритму прогнозування) призводить до зростання складності коду та збільшення ризику внесення помилок. Для системи з декількома самостійними джерелами даних та обчислювальними алгоритмами такий підхід є малопридатним.

Клієнт-серверна архітектура. Цей підхід передбачає розподіл системи на серверну частину, що відповідає за обробку даних і доступ до бази даних, та клієнтську частину, яка надає користувацький інтерфейс. Клієнт-серверна архітектура добре підходить для систем з багатьма користувачами або при необхідності доступу до даних з різних пристроїв. Однак для розроблюваної системи, що використовується одним користувачем на персональному комп'ютері, такий підхід є надлишковим: він вимагає підтримки мережевої

взаємодії, окремого розгортання серверної частини та збільшує загальну складність системи без отримання очевидних переваг.

Мікросервісна архітектура. У межах мікросервісного підходу система складається з набору незалежних сервісів, кожен з яких відповідає за окрему функцію та взаємодіє з іншими через мережеві протоколи. Цей підхід забезпечує високу гнучкість та масштабованість, проте є невиправдано складним для невеликих систем – він вимагає окремої інфраструктури, оркестрації сервісів, моніторингу та зрілих практик розгортання. Для дипломного проекту такий підхід є явним архітектурним переускладненням.

Багатошарова архітектура. Цей підхід передбачає поділ програми на логічні шари, кожен з яких відповідає за окремий аспект функціональності та взаємодіє лише з сусідніми шарами через чітко визначені інтерфейси. Зазвичай виділяють три шари: шар представлення (користувацький інтерфейс), шар бізнес-логіки (обробка даних та алгоритми) та шар доступу до даних (взаємодія з базою даних та зовнішніми джерелами). Багатошарова архітектура поєднує переваги монолітного підходу (простота розгортання) з гнучкістю розподілених систем (можливість незалежно змінювати та тестувати окремі шари). [1]

Для розроблюваної системи обрано саме багатошарову архітектуру. Така архітектура забезпечує чіткий поділ відповідальності між компонентами: апаратні джерела даних та симулятор реалізовані в окремому шарі та можуть взаємно замінюватись без змін в інших частинах системи; алгоритми прогнозування ізольовані в окремих класах, що дозволяє додавати нові методи без модифікації існуючого коду; робота з базою даних винесена в окремий шар, що спрощує можливу заміну системи зберігання у майбутньому.

Додатково в системі застосовано шаблони об'єктно-орієнтованого проектування, які поглиблюють переваги багатошарового підходу. Шаблон Strategy використано для абстрагування джерел даних та алгоритмів прогнозування, що дозволяє користувачу обирати реалізацію під час роботи

програми. Шаблон Repository застосовано для роботи з базою даних – він приховує деталі формування SQL-запитів та надає бізнес-логіці зручний об'єктний інтерфейс для зберігання та отримання вимірювань. Шаблон Singleton використано для управління єдиним з'єднанням з базою даних, що запобігає створенню зайвих з'єднань та оптимізує використання ресурсів. [2]

Представлений рисунок 2.1 демонструє архітектуру багат шарової системи, схеми трьох шарів з позначенням потоків даних.

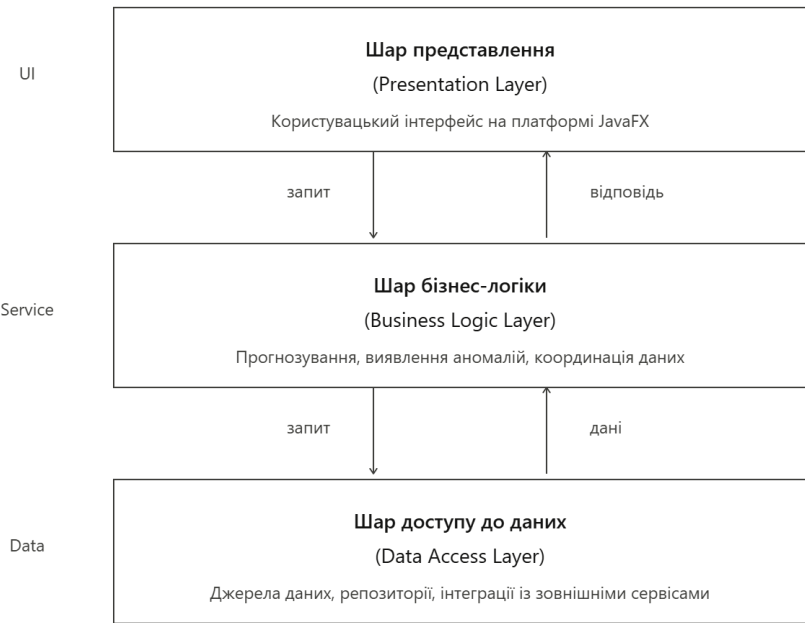


Рисунок 2.1 – Багат шарова архітектура системи: схема трьох шарів (UI / Service / Data) з позначенням потоків даних

Загальна структура розробленої системи відповідає класичному трьохшаровому розподілу: шар представлення реалізує користувацький інтерфейс на базі JavaFX і відповідає за візуалізацію температурних даних та взаємодію з користувачем; шар бізнес-логіки містить сервіси координації потоків даних, алгоритми прогнозування та виявлення аномалій; шар доступу

до даних об'єднує модулі взаємодії з мікроконтролером, реляційною базою даних PostgreSQL та зовнішніми сервісами (Telegram Bot API).

2.2 Обґрунтування вибору мови програмування

Вибір мови програмування є одним з ключових рішень при розробці програмного забезпечення, оскільки впливає на швидкість розробки, надійність роботи системи, можливості подальшого розвитку та сумісність з обраними технологіями. Для розроблюваної інформаційної системи розглянуто кілька альтернативних мов програмування, що широко застосовуються при створенні десктопних додатків з обробкою даних.

Python. Python є популярною високорівневою мовою програмування з простим синтаксисом, широкою стандартною бібліотекою та розвинутою екосистемою наукових обчислень. До переваг Python належать простота навчання, велика кількість бібліотек для аналізу даних (NumPy, Pandas, scikit-learn) та короткий час розробки прототипів. Водночас Python має й істотні недоліки для систем, що працюють у режимі реального часу: інтерпретована природа мови призводить до меншої швидкодії, відсутність статичної типізації ускладнює виявлення помилок на етапі компіляції, а стандартні засоби побудови десктопних інтерфейсів (Tkinter, PyQt) поступаються Java-аналогам за зрілістю та документацією.

С#. Мова С# від компанії Microsoft є потужним інструментом для розробки десктопних додатків під операційну систему Windows із застосуванням платформ WPF або WinForms. С# забезпечує високу швидкодію, статичну типізацію та зрілий інструментарій. Однак прив'язка до екосистеми Microsoft обмежує платформонезалежність системи: хоча існує .NET Core, повноцінна підтримка десктопних інтерфейсів на інших операційних системах залишається обмеженою. Враховуючи відкритий

характер розроблюваної системи та бажану можливість запуску на різних платформах, ця обмеженість є суттєвою.

C/C++. Мови C та C++ забезпечують максимальну продуктивність і прямий доступ до апаратних ресурсів, однак вимагають значно більших витрат часу на розробку та супровід порівняно з вищерівневими мовами. Ручне керування пам'яттю, відсутність вбудованих засобів роботи з мережевими ресурсами та необхідність використання сторонніх бібліотек для побудови графічного інтерфейсу робить ці мови менш придатними для розробки прикладного програмного забезпечення такого класу. C/C++ доцільніше застосовувати на стороні мікроконтролера, що й зроблено в межах цього проекту.

Java. Мова Java є об'єктно-орієнтованою, статично типізованою, платформонезалежною мовою із зрілою екосистемою, що активно розвивається понад тридцять років. До основних переваг Java належать: платформонезалежність (одна й та сама програма може виконуватись на Windows, Linux та macOS без перекомпіляції); потужна об'єктно-орієнтована модель з підтримкою всіх класичних шаблонів проектування; статична типізація, що дозволяє виявляти значну частину помилок на етапі компіляції; розвинена стандартна бібліотека, включаючи засоби роботи з мережею, файлами, потоками виконання та базами даних через стандарт JDBC; вбудована платформа JavaFX для побудови сучасних графічних інтерфейсів; зрілі засоби збору сміття, що звільняють розробника від ручного керування пам'яттю. [3]

Окремою перевагою Java для дипломного проекту є те, що ця мова входить до основної програми підготовки фахівців у галузі комп'ютерної інженерії, широко використовується в індустрії та має значну кількість навчальних матеріалів і документації українською й англійською мовами.

З урахуванням переваг, описаних вище, та з огляду на специфіку розроблюваної системи (десктопний додаток з обробкою даних у режимі

					ІАЛЦ.467200.004 ПЗ	Лист
Зм	Лист	№ докум.	Підп.	Дата		24

реального часу, взаємодією з апаратним обладнанням через послідовний порт, базою даних та зовнішніми вебсервісами) для реалізації обрано саме мову програмування Java версії 21 LTS. Версія Java 21 є актуальною версією з довгостроковою підтримкою (Long-Term Support) на момент розробки, що гарантує стабільність та доступність оновлень безпеки впродовж тривалого часу.

2.3 Опис програмного інструментарію

Розроблювана інформаційна система побудована на основі сучасного програмного інструментарію, що забезпечує надійну роботу всіх її компонентів. У цьому підрозділі розглянуто ключові програмні засоби, обрані для реалізації системи.

2.3.1 Платформа JavaFX

JavaFX – це сучасна програмна платформа для розробки графічних інтерфейсів користувача та мультимедійних додатків мовою Java. Платформа була представлена компанією Sun Microsystems у 2008 році та згодом стала офіційною заміною застарілої бібліотеки Swing для побудови десктопних додатків. З 2018 року JavaFX розвивається у складі окремого проекту OpenJFX під керівництвом організації Gluon у співпраці з Oracle. [4]

Основні можливості JavaFX:

- декларативний опис інтерфейсу за допомогою мови розмітки FXML, що дозволяє відокремити структуру вікон від програмної логіки;
- стилізація елементів інтерфейсу засобами CSS, аналогічно вебдодаткам;
- широкий набір вбудованих компонентів – таблиці, дерева, графіки, медіапрогравачі;

- розвинений модуль побудови графіків Charts, що включає лінійні графіки, гістограми, кругові діаграми та інші типи візуалізації;
- апаратно-прискорений рендерінг через інтерфейс OpenGL/DirectX;
- підтримка анімацій та ефектів;
- архітектурний шаблон Model-View-Controller (MVC), реалізований через зв'язку FXML-файлів з класами-контролерами.

Для розроблюваної системи JavaFX обрано з таких міркувань. По-перше, платформа є стандартом де-факто для побудови графічних інтерфейсів у Java-додатках, що забезпечує природну інтеграцію з рештою коду без використання зовнішніх обгортки. По-друге, вбудований модуль Charts дозволяє відображати температурні дані у вигляді графіка реального часу без необхідності використання сторонніх бібліотек візуалізації. По-третє, поділ структури інтерфейсу (FXML) та програмної логіки (Java-контролер) сприяє чистій архітектурі та полегшує супровід коду.

2.3.2 Система керування базами даних PostgreSQL

PostgreSQL – це об'єктно-реляційна система керування базами даних з відкритим вихідним кодом, що розвивається з 1986 року. На сьогоднішній день PostgreSQL вважається однією з найбільш зрілих, надійних та функціональних відкритих СКБД у світі та широко застосовується як у малих, так і у великих корпоративних проектах. [5]

Ключові характеристики PostgreSQL:

- повна підтримка стандарту SQL з розширеннями, що включають віконні функції, рекурсивні запити та збережені процедури;
- транзакційна модель ACID, що гарантує цілісність даних навіть у випадку збоїв системи;

- підтримка розширених типів даних – JSON, масиви, географічні координати, користувацькі типи;
- гнучка система індексування з підтримкою B-tree, hash, GiST, GIN та інших типів індексів;
- розвинена система прав доступу та автентифікації;
- багатоверсійна модель конкурентного доступу (MVCC), що дозволяє читачам та письменникам працювати одночасно без блокувань;
- безкоштовна ліцензія PostgreSQL License, що дозволяє вільне використання у комерційних та некомерційних проектах.

Для розроблюваної системи PostgreSQL обрано як надійну, безкоштовну та широкоживану СКБД, що повністю задовольняє вимоги до зберігання температурних даних. Альтернативні рішення, такі як вбудовані бази даних SQLite або H2, мають перевагу простоти розгортання, але поступаються PostgreSQL за надійністю транзакцій, можливостями індексування та підтримкою конкурентного доступу. У контексті дипломного проекту вибір PostgreSQL також демонструє вміння працювати з повноцінною серверною СКБД, що відповідає вимогам реальних промислових проектів.

У системі використовується одна таблиця `temperature_readings`, що містить поля: ідентифікатор запису (первинний ключ), значення температури, часова мітка вимірювання, джерело даних (Arduino або симулятор), ознака аномальності значення. Структура таблиці детально розглянута у розділі 3.

2.3.3 Середовище розробки Arduino IDE

Для програмування мікроконтролера Arduino UNO використано офіційне середовище розробки Arduino IDE – інтегроване середовище, створене компанією Arduino для написання, компіляції та завантаження програм (так званих «скетчів») на мікроконтролерні плати. Середовище є

безкоштовним, кросплатформним та підтримує операційні системи Windows, Linux і macOS. [6]

Основні можливості Arduino IDE:

- редактор коду з підсвічуванням синтаксису для мови C/C++ у спрощеній формі;
- однокроковий процес компіляції та завантаження програми на мікроконтролер через USB-порт;
- вбудований Serial Monitor для відображення даних, що передаються мікроконтролером через послідовний порт;
- менеджер бібліотек з широкою колекцією готових модулів для роботи з різноманітними датчиками та модулями;
- менеджер плат, що дозволяє додавати підтримку широкого спектра мікроконтролерних платформ.

Arduino IDE використано для розробки скетча, що зчитує показники з датчика DHT11, форматує їх та передає на персональний комп'ютер через послідовний порт (як через USB-кабель, так і через бездротовий модуль Bluetooth).

2.4 Опис апаратного інструментарію

Апаратна частина розроблюваної системи побудована на основі мікроконтролерної платформи Arduino та підключених до неї модулів – температурного датчика, бездротового модуля передачі даних та символьного дисплея. У цьому підрозділі розглянуто кожен з апаратних компонентів.

2.4.1 Мікроконтролер Arduino UNO

Arduino UNO – це одноплатна мікроконтролерна платформа на базі чипа ATmega328P від компанії Microchip (раніше Atmel). Платформа була

					ІАЛЦ.467200.004 ПЗ	Лист 28
Зм	Лист	№ докум.	Підп.	Дата		

представлена італійською компанією Arduino у 2010 році та з того часу стала одним з найпопулярніших засобів для прототипування електронних пристроїв, навчання та реалізації проектів у галузі «Інтернету речей» (IoT). [7]

Технічні характеристики Arduino UNO:

- мікроконтролер ATmega328P з тактовою частотою 16 МГц;
- 32 КБ флеш-пам'яті для зберігання програми, 2 КБ оперативної пам'яті, 1 КБ енергонезалежної пам'яті EEPROM;
- 14 цифрових входів/виходів, з яких 6 підтримують широтно-імпульсну модуляцію (ШІМ);
- 6 аналогових входів з 10-розрядним аналого-цифровим перетворювачем;
- апаратний послідовний інтерфейс UART, що використовується для зв'язку з персональним комп'ютером через USB або з модулями Bluetooth;
- апаратний інтерфейс I2C для підключення зовнішніх пристроїв;
- робоча напруга 5 В, можливість живлення через USB-порт або від зовнішнього джерела напругою 7-12 В;
- ціна на світовому ринку від 5 до 25 доларів США залежно від виробника (оригінальний або сумісний аналог).

Вигляд мікроконтролерної плати Arduino UNO наведено на рисунку 2.2.



Рисунок 2.2 – Зовнішній вигляд мікроконтролерної плати Arduino UNO

Зм	Лист	№ докум.	Підп.	Дата

ІАЛЦ.467200.004 ПЗ

Лист
29

Arduino UNO обрано для розроблюваної системи з огляду на: достатню для задачі обчислювальну потужність, наявність необхідних апаратних інтерфейсів (UART для бездротового модуля, цифровий пін для температурного датчика), широку доступність та низьку вартість, велику кількість документації та готових бібліотек для роботи з різноманітними датчиками. Окремою перевагою є популярність платформи у навчальній сфері, що робить розроблений комплекс зрозумілим та відтворюваним для інших користувачів.

2.4.2 Датчик температури DHT11

DHT11 – це цифровий датчик температури та відносної вологості повітря, призначений для застосування в електронних проектах. Датчик випускається компанією Aosong Electronics та широко використовується в системах моніторингу мікроклімату завдяки низькій ціні та простоті інтеграції. [8]

Технічні характеристики DHT11:

- діапазон вимірювання температури: від 0 °C до 50 °C;
- роздільна здатність вимірювання температури: 1 °C;
- похибка вимірювання температури: ± 2 °C;
- діапазон вимірювання відносної вологості: від 20 % до 90 %;
- похибка вимірювання вологості: ± 5 %;
- частота вимірювань: не частіше одного разу на секунду;
- напруга живлення: від 3 до 5,5 В;
- однопровідний цифровий інтерфейс передачі даних.

Зовнішній вигляд bluetooth-модуля HC-05 наведено на рисунку 2.5.

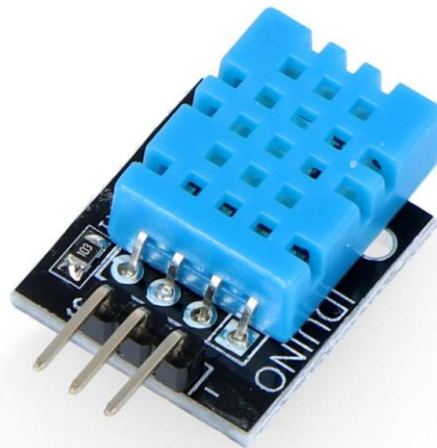


Рисунок 2.3 – Зовнішній вигляд температурного датчика DHT11

Датчик DHT11 обрано як доступне та просте у використанні рішення для задачі моніторингу температури в побутових та навчальних умовах. Хоча DHT11 поступається за точністю більш досконалим датчикам (наприклад, DHT22 з похибкою $\pm 0,5$ °C або BME280 з похибкою $\pm 0,5$ °C), його характеристики є цілком достатніми для демонстрації роботи системи та реалізації алгоритмів прогнозування й виявлення аномалій. Завдяки відкритій архітектурі розроблюваної системи DHT11 може бути замінений на точніший датчик без модифікації основного коду – достатньо реалізувати відповідну версію інтерфейсу DataSource.

2.4.3 Модуль бездротового зв'язку Bluetooth HC-05

HC-05 – це поширений модуль бездротового зв'язку стандарту Bluetooth 2.0+EDR (Enhanced Data Rate), призначений для організації каналу серійної передачі даних між електронними пристроями. Модуль випускається багатьма виробниками за ліцензією китайської компанії Guangzhou HC Information Technology та широко застосовується в проектах на базі Arduino, Raspberry Pi та інших мікроконтролерних платформ. [9]

Технічні характеристики HC-05:

- стандарт зв'язку: Bluetooth 2.0+EDR;
- профіль роботи: Serial Port Profile (SPP) – емуляція послідовного порту;
- режим роботи: master або slave (для розроблюваної системи використовується режим slave);
- частотний діапазон: 2,4 ГГц;
- дальність зв'язку: до 10 метрів у відкритому просторі;
- напруга живлення модуля: від 3,3 до 6 В;
- логічні рівні портів вводу/виводу: 3,3 В;
- швидкість передачі даних за замовчуванням: 9600 бод;
- стандартний PIN-код для спарювання: 1234 або 0000.

Зовнішній вигляд bluetooth-модуля HC-05 наведено на рисунку 2.6.

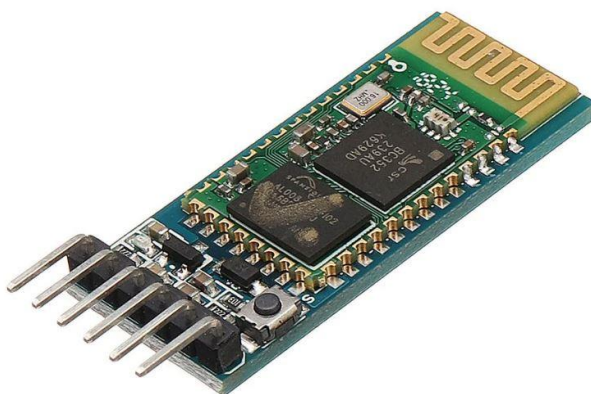


Рисунок 2.4 – Зовнішній вигляд модуля Bluetooth HC-05

Модуль HC-05 обрано для забезпечення можливості бездротової передачі даних від мікроконтролера до персонального комп'ютера. Така можливість підвищує практичну зручність системи: апаратна частина може розміщуватись у віддаленій від комп'ютера точці (наприклад, в іншому приміщенні), при цьому збереження даних та їх аналіз виконуються централізовано. З програмного боку перехід між дротовим (USB) та

бездротовим (HC-05) з'єднанням не потребує жодних змін у коді додатку – обидва канали для операційної системи виглядають як стандартний послідовний COM-порт.

Слід зазначити, що логічні рівні портів HC-05 розраховані на напругу 3,3 В, тоді як Arduino UNO працює на 5 В. Для коректної та довговічної роботи модуля при підключенні до Arduino застосовано дільник напруги на основі двох резисторів (1 кОм та 2 кОм), що знижує напругу сигналу від мікроконтролера до безпечного для модуля рівня.

2.4.4 Дисплей LCD 1602 з інтерфейсом I2C

LCD 1602 – це символний рідкокристалічний дисплей розміром 16 символів на 2 рядки, побудований на основі контролера HD44780 від компанії Hitachi. Дисплей призначений для відображення текстової інформації в електронних пристроях та широко застосовується в проектах на базі мікроконтролерів. [10]

У стандартному виконанні дисплей LCD 1602 підключається до мікроконтролера через паралельний інтерфейс із використанням від 6 до 12 пінів, що ускладнює його інтеграцію в проекти з обмеженою кількістю вільних входів/виходів. Для спрощення підключення застосовується спеціальний перехідник на базі мікросхеми PCF8574, що перетворює паралельний інтерфейс дисплея на послідовний інтерфейс I2C, який вимагає лише двох ліній даних (SDA та SCL) на додаток до живлення.

Характеристики дисплея LCD 1602 з I2C-перехідником:

- розмір екрана: 16 символів × 2 рядки;
- підсвічування: світлодіодне (зазвичай синього кольору);
- регульована контрастність зображення;
- напруга живлення: 5 В;
- інтерфейс підключення: I2C (адреса за замовчуванням 0x27);

- кількість необхідних пінів мікроконтролера: 2 (SDA, SCL) + живлення;
- ціна на світовому ринку: 3-8 доларів США.

Зовнішній вигляд цього дисплею продемонстровано на рисунку 2.7.

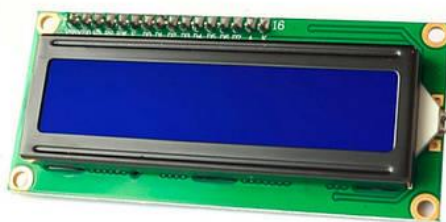


Рисунок 2.5 – Зовнішній вигляд дисплея LCD 1602 з I2C-модулем

Дисплей LCD 1602 з I2C-інтерфейсом обрано для розроблюваної системи з метою забезпечення автономного відображення поточної температури безпосередньо на апаратній частині, без необхідності постійного підключення до персонального комп'ютера. Завдяки використанню I2C-перехідника інтеграція дисплея у схему не потребує великої кількості пінів мікроконтролера та залишає вільними інші входи/виходи Arduino для подальшого розширення системи.

2.5 Аналіз використаних бібліотек

Розробка сучасних програмних систем неминуче передбачає використання сторонніх бібліотек, що інкапсулюють поширені задачі та

звільняють розробника від необхідності повторної реалізації стандартних механізмів. У розроблюваній системі застосовано низку бібліотек як для серверної частини на мові Java, так і для мікроконтролерної частини на мові C/C++. У цьому підрозділі розглянуто кожен з ключових бібліотек, обґрунтовано її вибір та описано спосіб застосування в межах проекту.

2.5.1 Бібліотека jSerialComm

jSerialComm – це бібліотека з відкритим вихідним кодом для роботи з послідовними портами в програмах на мові Java. Бібліотека розроблена Уіллом Гайеком (Will Hedgesock) та поширюється за ліцензією Apache 2.0. На відміну від стандартних засобів Java, що не надають прямого доступу до апаратних портів, jSerialComm використовує нативні бібліотеки для кожної операційної системи (Windows, Linux, macOS), забезпечуючи кросплатформну роботу з COM-портами. [11]

Ключові можливості бібліотеки:

- автоматичне визначення доступних послідовних портів у системі;
- відкриття та закриття портів з налаштуванням швидкості передачі, кількості біт даних, паритету та стоп-біт;
- блокуюче та неблокуюче читання даних з підтримкою налаштовуваних тайм-аутів;
- запис даних у порт у синхронному режимі;
- подієва модель повідомлень про надходження нових даних;
- кросплатформність – однакова поведінка на Windows, Linux та macOS без додаткового налаштування.

У розроблюваній системі бібліотека jSerialComm використовується для отримання температурних даних з мікроконтролера Arduino UNO через послідовний порт. Незалежно від того, чи підключення здійснюється через USB-кабель, чи через бездротовий модуль Bluetooth HC-05, операційна

система представляє обидва канали як стандартний COM-порт, з яким jSerialComm взаємодіє однаковим чином. Це підтверджує доцільність обраної багат шарової архітектури: зміна фізичного каналу зв'язку не вимагає жодних змін у коді додатку.

Робота з бібліотекою інкапсульована в окремому класі ArduinoSerialSource, який реалізує загальний інтерфейс DataSource. Завдяки такій абстракції основна логіка системи не залежить від конкретної бібліотеки роботи з послідовним портом – у разі потреби jSerialComm може бути замінений на альтернативну бібліотеку (наприклад, RXTX або Java Communications API) без зміни решти коду.

2.5.2 Бібліотека HikariCP

HikariCP – це високопродуктивна бібліотека керування пулом з'єднань з базою даних для Java-додатків. Бібліотека розроблена компанією Zahxer та поширюється за ліцензією Apache 2.0. HikariCP вважається одним з найшвидших та найнадійніших пулів з'єднань у Java-екосистемі та широко використовується в промислових системах, у тому числі як стандартний пул з'єднань у популярному фреймворку Spring Boot. [12]

Поняття пулу з'єднань пов'язане з оптимізацією роботи з базою даних. Створення з'єднання з СКБД є відносно витратною операцією, що включає мережеву взаємодію, автентифікацію та налаштування сесії. Якщо для кожного запиту створювати нове з'єднання, продуктивність системи буде неприйнятно низькою. Натомість пул з'єднань заздалегідь створює певну кількість з'єднань та надає їх додатку у вигляді багаторазового ресурсу: коли додаток потребує з'єднання, він отримує його з пулу, а після виконання запиту повертає назад для повторного використання.

Основні переваги HikariCP:

- мінімальні накладні витрати на отримання та повернення з'єднань (порядку мікросекунд);
- автоматичне виявлення та усунення непрацездатних з'єднань;
- гнучке налаштування мінімальної та максимальної кількості з'єднань у пулі, тайм-аутів та параметрів керування;
- невеликий розмір бібліотеки та відсутність зайвих залежностей;
- надійна обробка крайових випадків та помилок.

У розроблюваній системі HikariCP використовується в класі DatabaseManager, що реалізує шаблон Singleton та надає всім модулям доступ до спільного пулу з'єднань з базою даних PostgreSQL. Хоча розроблювана система має лише одного користувача та одне основне джерело даних, використання пулу з'єднань залишається доцільним: воно забезпечує швидкий доступ до бази при пакетному записі даних та готує систему до можливого розширення (наприклад, додавання вебінтерфейсу або підтримки кількох джерел даних одночасно).

2.5.3 Бібліотека Apache Commons Math

Apache Commons Math – це бібліотека математичних та статистичних функцій для мови Java, що розробляється під егідою фонду Apache Software Foundation та поширюється за ліцензією Apache 2.0. Бібліотека надає реалізації широкого спектру математичних алгоритмів, які відсутні в стандартній бібліотеці Java або реалізовані в ній на елементарному рівні. [13]

Можливості бібліотеки охоплюють:

- описову статистику (середнє, медіана, дисперсія, стандартне відхилення тощо);
- лінійну алгебру (операції з матрицями та векторами);

- регресійний аналіз, у тому числі лінійну регресію за методом найменших квадратів;
- розв'язання систем лінійних та нелінійних рівнянь;
- чисельне інтегрування та диференціювання;
- оптимізацію та пошук екстремумів функцій;
- генерацію випадкових чисел з різноманітних розподілів;
- перетворення Фур'є та інші обчислювальні алгоритми.

У розроблюваній системі Apache Commons Math використовується для реалізації двох ключових алгоритмів. По-перше, клас SimpleRegression застосовується для побудови лінійної регресії на основі останніх N вимірювань температури та прогнозування наступного значення. По-друге, клас DescriptiveStatistics використовується для обчислення середнього значення та стандартного відхилення у вікні останніх вимірювань, що є основою для виявлення аномалій за статистичним z -критерієм.

Альтернативою використанню Apache Commons Math могла би бути самостійна реалізація відповідних алгоритмів. Однак застосування зрілої бібліотеки має кілька істотних переваг: реалізація є відомою та перевіреною спільнотою (що мінімізує ризик помилок), обчислення оптимізовані з точки зору швидкодії та чисельної стійкості, а наявність документації спрощує супровід коду.

2.5.4 Бібліотеки для мікроконтролерної частини

Для програмування мікроконтролера Arduino UNO використано три бібліотеки, що значно спрощують роботу з апаратними модулями.

Бібліотека DHT.h від компанії Adafruit Industries призначена для роботи з датчиками температури та вологості сімейства DHT (включаючи DHT11, DHT22 та AM2302). Бібліотека інкапсулює складний протокол однопровідної передачі даних, що використовується цими датчиками: вона забезпечує

правильне формування сигналів запиту, синхронізацію з відповіддю датчика, перевірку контрольної суми та повернення прочитаних значень у зручному форматі. Без використання цієї бібліотеки розробнику довелося би самостійно реалізовувати тимчасово критичний протокол на рівні мікросекунд, що значно ускладнило б код скетча. [14]

Бібліотека `SoftwareSerial.h` входить до стандартної комплектації Arduino IDE та дозволяє програмно емулювати додаткові послідовні порти на будь-яких цифрових пінах мікроконтролера. Це є критично важливим для розроблюваної системи, оскільки Arduino UNO має лише один апаратний послідовний порт (UART) на пінах 0 та 1, який одночасно використовується для зв'язку з персональним комп'ютером через USB та для завантаження програми. Використання `SoftwareSerial` дозволяє підключити модуль Bluetooth HC-05 на окремі піни (10 та 11 у розробленій схемі), не блокуючи апаратний UART. [15]

Бібліотека `LiquidCrystal_I2C` є розширенням стандартної бібліотеки `LiquidCrystal` для роботи з символьними дисплеями через інтерфейс I2C. Бібліотека приховує деталі взаємодії з мікросхемою PCF8574 на платі дисплея і надає простий API для виведення тексту, керування курсором, увімкнення та вимкнення підсвічування. Її використання дозволило підключити дисплей LCD 1602 до мікроконтролера всього двома лініями даних (SDA та SCL), залишивши вільними інші піни Arduino. [16]

2.6 Зовнішні сервіси: Telegram Bot API

Окрім локальних програмних та апаратних компонентів, розроблювана інформаційна система взаємодіє із зовнішнім сервісом – платформою месенджера Telegram через її офіційний програмний інтерфейс Telegram Bot API. Цей інтерфейс надає можливість стороннім додаткам надсилати та

отримувати повідомлення у месенджері від імені спеціальних облікових записів – телеграм-ботів. [17]

Месенджер Telegram було обрано як канал доставки сповіщень користувачу з таких міркувань:

- широке поширення месенджера в Україні та світі – Telegram має понад 800 мільйонів активних користувачів та є одним з найпопулярніших месенджерів у регіоні;
- безкоштовне використання Bot API без обмежень за кількістю надісланих повідомлень для типових сценаріїв;
- наявність офіційного, добре документованого та стабільного програмного інтерфейсу;
- миттєва доставка повідомлень на мобільний пристрій з push-сповіщеннями;
- крос-платформність – користувач отримує сповіщення на будь-якому пристрої, де встановлено клієнт Telegram (смартфон, планшет, персональний комп'ютер);
- простота початкового налаштування – створення бота не вимагає реєстрації окремого облікового запису або юридичних формальностей.

Технічно Telegram Bot API реалізований як HTTP-сервіс, що приймає запити за стандартним протоколом HTTPS. Кожен бот ідентифікується унікальним токеном, отриманим від офіційного бота-творця @BotFather. Надсилання повідомлення зводиться до виконання HTTP-запиту до кінцевої точки API з параметрами, що включають токен бота, ідентифікатор отримувача (chat_id) та текст повідомлення.

Серед альтернативних каналів доставки сповіщень (електронна пошта, SMS, push-сповіщення власного мобільного додатку) Telegram має ряд практичних переваг. Електронна пошта характеризується затримками доставки та низькою помітністю для користувача. SMS-сповіщення є платними та потребують укладення договору з оператором. Розробка власного

мобільного додатку – масштабна задача, що виходить за межі дипломного проекту.

Інтеграція з Telegram Bot API в розроблюваній системі реалізована через клас `TelegramNotifier`, що інкапсулює формування HTTP-запитів та асинхронну їх відправку за допомогою вбудованого в Java 11+ класу `HttpClient`. Такий підхід не потребує жодних додаткових бібліотек, що зменшує загальну кількість залежностей системи. У межах архітектури системи `TelegramNotifier` є однією з реалізацій каналу сповіщень – у майбутньому до нього можуть бути додані інші канали (електронна пошта, SMS, інші месенджери) без зміни основного коду.

Окремий клас `TelegramNotifier` інтегрований у шар бізнес-логіки системи: при кожному виявленні аномального значення сервіс координації даних автоматично формує повідомлення з описом аномалії та надсилає його через об'єкт `TelegramNotifier`. Детальніше архітектура та принцип роботи цієї інтеграції розглянуті у розділі 3.

2.7 Висновок до розділу 2

У другому розділі проведено детальний аналіз та обґрунтовано вибір засобів реалізації інформаційної системи моніторингу та прогнозування температури середовища.

Розглянуто чотири основних архітектурних підходи (монолітна, клієнт-серверна, мікросервісна та багатошарова архітектури) та обґрунтовано вибір багатошарової архітектури як такої, що оптимально відповідає вимогам системи з точки зору простоти реалізації, можливості розширення та чіткого розподілу відповідальності між компонентами. Додатково обґрунтовано застосування об'єктно-орієнтованих шаблонів проектування `Strategy`, `Repository` та `Singleton` для забезпечення гнучкості системи.

Здійснено порівняльний аналіз мов програмування Python, C#, C/C++ та Java, за результатами якого для реалізації системи обрано мову Java версії 21 LTS, що поєднує платформонезалежність, статичну типізацію, потужну об'єктно-орієнтовану модель та зрілу екосистему засобів розробки.

Описано програмний та апаратний інструментарій, обраний для реалізації системи: платформу JavaFX для побудови графічного інтерфейсу, систему керування базами даних PostgreSQL для зберігання вимірювань, середовище розробки Arduino IDE для програмування мікроконтролера, мікроконтролерну платформу Arduino UNO з датчиком DHT11, модуль бездротового зв'язку Bluetooth HC-05 та дисплей LCD 1602 з I2C-інтерфейсом. Проаналізовано бібліотеки, використані для реалізації окремих функціональних можливостей: jSerialComm для роботи з послідовним портом, HikariCP для керування пулом з'єднань з базою даних, Apache Commons Math для реалізації статистичних алгоритмів, а також бібліотеки DHT.h, SoftwareSerial.h та LiquidCrystal_I2C для роботи з апаратними модулями на стороні мікроконтролера.

Окрему увагу приділено інтеграції із зовнішнім сервісом – платформою Telegram через офіційний Bot API, що використано як основний канал миттєвих сповіщень користувача про виявлені аномальні зміни температури. Обрані технічні рішення в сукупності забезпечують реалізацію всіх функціональних вимог, висунутих до системи у технічному завданні, та створюють надійну основу для подальшого проектування та реалізації, розглянутих у наступному розділі.

3. ОПИС РОЗРОБЛЕНОЇ СИСТЕМИ

3.1 Загальна архітектура системи

Розроблена інформаційна система моніторингу та прогнозування температури середовища має багатошарову архітектуру, що поєднує апаратну частину для збору даних та програмну частину для їх обробки, зберігання та візуалізації. Загальна структура системи представлена на рисунку 3.1.

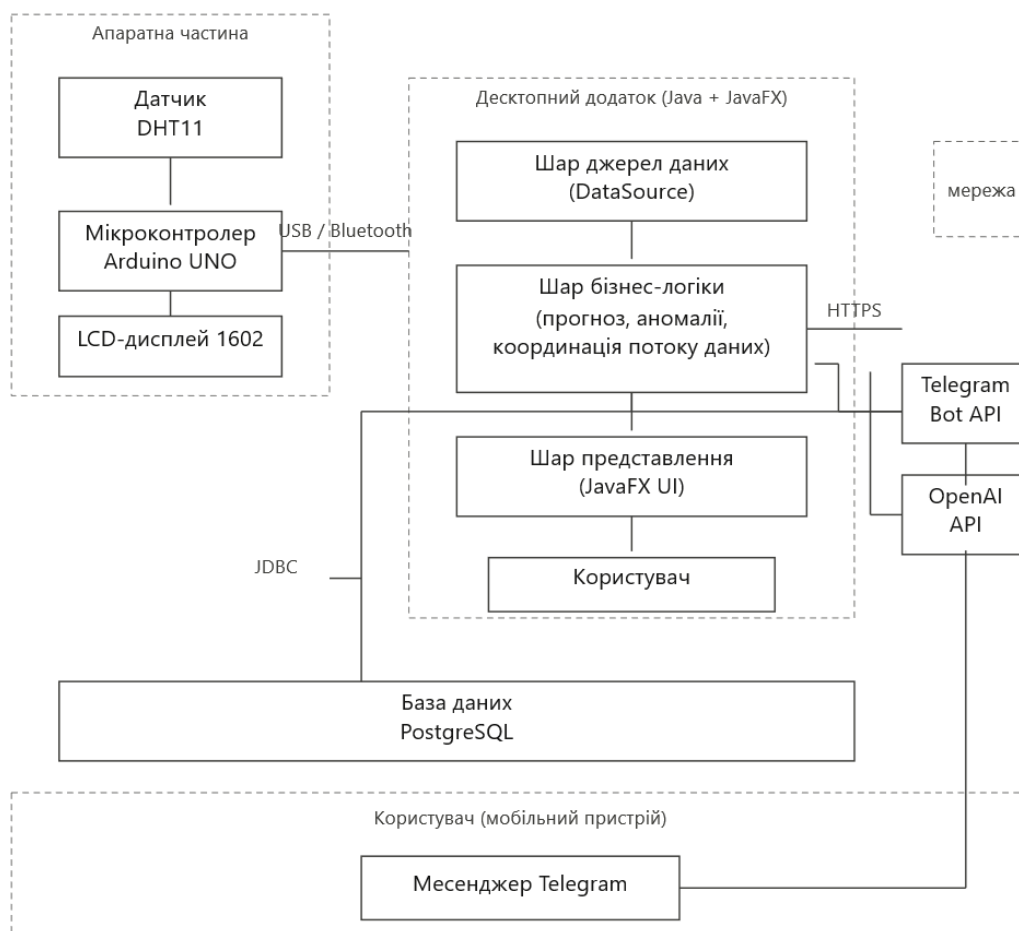


Рисунок 3.1 – Загальна структурна схема розробленої системи

Система складається з таких основних компонентів:

Апаратна частина включає мікроконтролерну плату Arduino UNO з підключеним до неї цифровим датчиком температури DHT11, модулем

бездротового зв'язку Bluetooth HC-05 та символьним дисплеєм LCD 1602 з інтерфейсом I2C. Мікроконтролер періодично опитує датчик, отримує значення температури та передає їх на персональний комп'ютер двома незалежними каналами – через USB-кабель та через бездротове Bluetooth-з'єднання. Поточне значення температури також виводиться на локальний LCD-дисплей.

Десктопний додаток на основі платформи Java та бібліотеки JavaFX виконується на персональному комп'ютері користувача та реалізує основну обробку даних. Програма приймає виміри температури від мікроконтролера через послідовний порт, передає їх через шар бізнес-логіки на зберігання в базу даних, виконує короткостроковий прогноз майбутніх значень за обраним користувачем алгоритмом, виявляє аномальні зміни показників та відображає всю інформацію в графічному інтерфейсі у вигляді графіка реального часу.

Реляційна база даних PostgreSQL виконує роль довгострокового сховища температурних вимірювань. Усі прочитані з датчика значення зберігаються у єдиній таблиці з вказівкою часової мітки, джерела даних та ознаки аномальності. Збережені дані можуть бути використані для подальшого аналізу, експорту у формат CSV або виконання обчислень метрик якості прогнозу.

Telegram-бот виступає як зовнішній канал доставки сповіщень. При виявленні аномального значення сервіс координації даних у десктопному додатку формує текстове повідомлення з описом події та надсилає його через HTTP-запит до Telegram Bot API. Користувач отримує миттєве сповіщення на своєму мобільному пристрої незалежно від того, чи спостерігає він у цей момент за основним вікном програми.

Взаємодія між компонентами організована за принципом потоку даних від апаратної частини до користувача:

Датчик DHT11 → Arduino UNO → послідовний порт (USB/Bluetooth) → DataSource → DataService → Repository → PostgreSQL → PredictionStrategy → MainController → JavaFX UI → Користувач;

додатково: JavaFX UI → AnomalyDetector → TelegramNotifier → Telegram API → Користувач.

Така архітектура забезпечує чіткий розподіл відповідальності між компонентами, спрощує тестування окремих модулів та залишає систему відкритою для розширення. Зокрема, заміна джерела даних (наприклад, перехід на інший тип датчика), додавання нового алгоритму прогнозування або підключення додаткового каналу сповіщень не вимагає модифікації основного коду – достатньо реалізувати відповідний інтерфейс та зареєструвати нову реалізацію в системі.

3.2 Апаратна частина та схема підключення

Апаратна частина розробленої системи побудована навколо мікроконтролера Arduino UNO, до якого підключено три модулі: датчик температури DHT11, модуль бездротового зв'язку Bluetooth HC-05 та символічний дисплей LCD 1602 з I2C-інтерфейсом. Схема електричного підключення компонентів наведена на рисунку 3.2.

					ІАЛЦ.467200.004 ПЗ	Лист 45
Зм	Лист	№ докум.	Підп.	Дата		

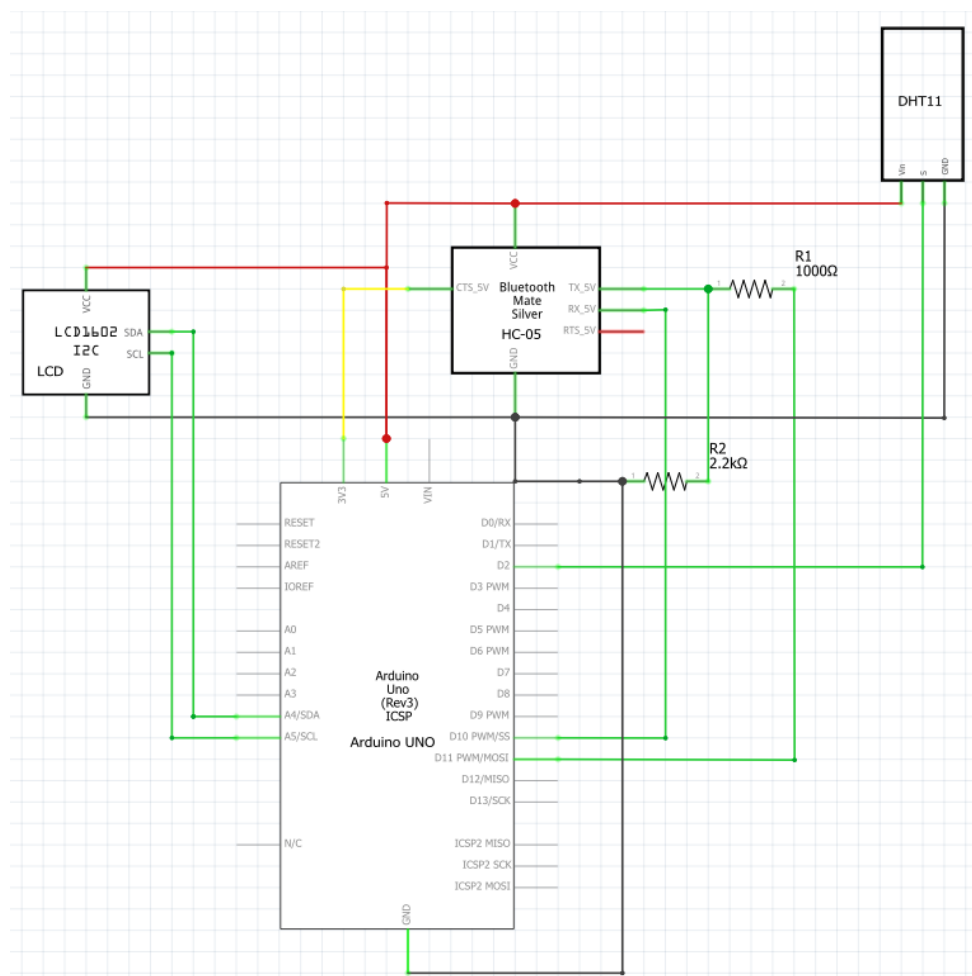


Рисунок 3.2 – Схема електричного підключення компонентів апаратної частини

Підключення кожного з модулів виконано відповідно до їх технічних характеристик та особливостей роботи.

Підключення датчика DHT11. Датчик має три активні виводи: живлення, землю та лінію даних. Живлення (5 В) та земля підключені до відповідних виводів Arduino, а лінія даних – до цифрового піна D2. Між лінією даних та виводом живлення встановлено підтягуючий резистор номіналом 10 кОм, який забезпечує стабільну роботу однопровідного протоколу DHT. У деяких модифікаціях датчика DHT11 (наприклад, у виконанні на платі-перехіднику) такий резистор вже інтегрований у саму плату, що спрощує підключення.

Підключення модуля Bluetooth HC-05. Модуль HC-05 має чотири основних виводи: живлення (VCC), землю (GND), а також лінії передачі (TXD) та прийому (RXD) даних. Живлення модуля забезпечується напругою 5 В від Arduino, а земля підключена до спільної шини. Лінії передачі даних організовані наступним чином: вихід TXD модуля підключений до програмно емульованого піна 10 Arduino безпосередньо, оскільки логічний рівень 3,3 В з боку модуля коректно розпізнається Arduino як логічна одиниця.

Лінія RXD модуля підключена через дільник напруги, побудований на двох резисторах номіналом 1 кОм та 2 кОм. Це необхідно з тієї причини, що Arduino UNO працює з логічними рівнями 5 В, тоді як входи модуля HC-05 розраховані на 3,3 В. Безпосереднє підключення 5-вольтового сигналу до 3,3-вольтового входу може з часом пошкодити модуль. Дільник напруги знижує сигнал від Arduino (пін 11) до приблизно 3,3 В за формулою:

$$U_{\text{вих}} = U_{\text{вх}} \times R2 / (R1 + R2) = 5 \times 2 / (1 + 2) \approx 3,33 \text{ В}$$

де $U_{\text{вх}}$ – напруга на вході дільника (5 В з Arduino), $R1$ та $R2$ – номінали резисторів (1 кОм та 2 кОм відповідно). Така схема дільника напруги є стандартним рішенням при сполученні логіки 5 В та 3,3 В у проектах на базі Arduino.

Для зв'язку Arduino з модулем HC-05 використано програмний послідовний інтерфейс на пінах 10 та 11, реалізований за допомогою бібліотеки SoftwareSerial. Це дозволяє зберегти апаратний послідовний порт (піни 0 та 1) для зв'язку з персональним комп'ютером через USB-кабель та для завантаження програми в мікроконтролер.

Підключення дисплея LCD 1602. Завдяки наявності I2C-перехідника на платі дисплея підключення зведено до чотирьох виводів: VCC (5 В), GND (земля), SDA (лінія даних I2C) та SCL (лінія синхронізації I2C). Лінії SDA та SCL підключені до спеціалізованих пінів Arduino UNO – A4 та A5 відповідно,

що мають апаратну підтримку інтерфейсу I2C. Адреса дисплея в шині I2C за замовчуванням становить 0x27, що відповідає стандартному налаштуванню більшості модулів цього типу.

Логіка роботи апаратної частини. Мікроконтролер виконує програму (скетч), записану в його флеш-пам'ять. У основному циклі програми мікроконтролер виконує такі дії:

1. Зчитування поточного значення температури з датчика DHT11 за допомогою бібліотеки DHT.h.
2. Формування текстового рядка з виміряним значенням у форматі з двома знаками після коми (наприклад, «23.40»).
3. Передача сформованого рядка одночасно у два канали: апаратний послідовний порт (для USB-підключення) та програмний послідовний порт (для HC-05).
4. Виведення поточного значення температури на LCD-дисплей з оновленням раз на цикл.
5. Затримка тривалістю 2 секунди перед наступним виміром, що відповідає максимальній частоті опитування датчика DHT11.

Таким чином, мікроконтролер виступає автономним вузлом збору даних, який не потребує постійного підключення до персонального комп'ютера для виконання своїх базових функцій – поточна температура відображається на LCD-дисплеї навіть без активного програмного забезпечення на боці ПК.

3.3 Структура програмного забезпечення

Програмне забезпечення розробленої системи побудоване на основі багатопарової архітектури з чітким розподілом відповідальності між компонентами. Загальна структура програмних модулів представлена на рисунку 3.3.

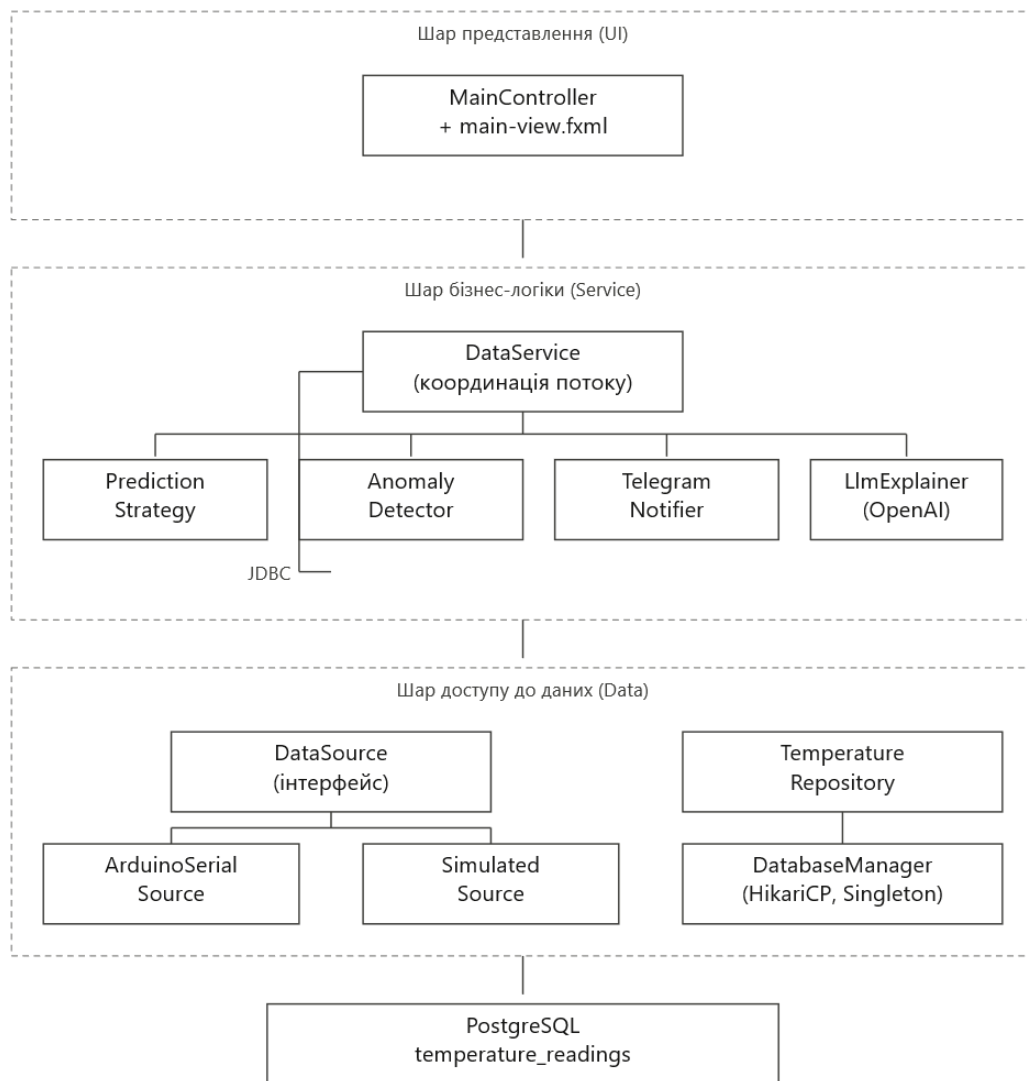


Рисунок 3.3 – Структурна схема програмного забезпечення системи

- Усі класи програми згруповані за функціональним призначенням у пакети:
- `com.lytvyn.tempmonitor.model` – класи-моделі даних (імутабельні записи про вимірювання та прогноз);
 - `com.lytvyn.tempmonitor.source` – модулі-джерела даних (інтерфейс `DataSource` та його реалізації);
 - `com.lytvyn.tempmonitor.db` – модулі роботи з базою даних PostgreSQL;
 - `com.lytvyn.tempmonitor.service` – модулі бізнес-логіки (координація даних, сповіщення, експорт);

- `com.lytvyn.tempmonitor.service.prediction` – модулі алгоритмів прогнозування;
- `com.lytvyn.tempmonitor.ui` – модулі користувацького інтерфейсу.

Така структура відповідає класичному трьохшаровому розподілу: пакети `model`, `source` та `db` належать до шару доступу до даних, `service` – до шару бізнес-логіки, а `ui` – до шару представлення. Кожен шар взаємодіє лише з сусіднім через чітко визначені інтерфейси, що забезпечує слабкий зв'язок між компонентами та дозволяє замінювати окремі реалізації без впливу на інші частини системи.

У наступних підпунктах розглянуто кожен з функціональних модулів програми.

3.3.1 Модуль джерел даних

Модуль джерел даних відповідає за отримання температурних вимірювань з різних фізичних джерел та передачу їх до шару бізнес-логіки. Архітектура модуля побудована на основі шаблону проектування `Strategy`, що дозволяє абстрагувати спосіб отримання даних від основної логіки програми.

Центральним елементом модуля є інтерфейс `DataSource`, який визначає контракт для всіх типів джерел даних:

```
public interface DataSource {
    void start(Consumer<TemperatureReading> listener);
    void stop();
    boolean isRunning();
    String getName();
}
```

Метод `start` приймає `callback`-функцію, яка буде викликатися при кожному новому вимірі температури. Метод `stop` зупиняє джерело даних,

isRunning повертає поточний стан, а getName – людиночитну назву джерела для відображення у користувацькому інтерфейсі.

У межах розробленої системи реалізовано дві конкретні реалізації цього інтерфейсу.

Клас ArduinoSerialSource забезпечує отримання даних з мікроконтролера Arduino через послідовний порт. Внутрішньо клас використовує бібліотеку jSerialComm для відкриття COM-порту та читання даних у фоновому потоці. Після відкриття порту клас очікує приблизно 2 секунди для завершення процесу перезавантаження Arduino (що автоматично відбувається при відкритті порту через сигнал DTR), очищає буфер від службових даних, а потім у нескінченному циклі читає рядки тексту, парсить їх як числа з плаваючою крапкою та передає кожне валідне значення через callback-функцію.

Особливістю реалізації є використання режиму блокуючого читання порту TIMEOUT_READ_BLOCKING з нульовими таймаутами. Такий режим дозволяє потоку читання очікувати нових даних необмежено довго, не споживаючи процесорний час та не реагуючи на короткочасні паузи у потоці даних. Це принципово важливо для роботи з Arduino, який передає виміри лише раз на 2 секунди.

Клас SimulatedSource реалізує програмний симулятор джерела даних та використовується для розробки й тестування програми без підключеного апаратного забезпечення. Симулятор генерує значення температури за заданою базовою величиною з невеликими випадковими відхиленнями, періодично додаючи штучні «аномалії» – значення, що різко відрізняються від базового рівня. Це дозволяє тестувати алгоритми виявлення аномалій без необхідності фізично впливати на датчик.

Завдяки спільному інтерфейсу обидві реалізації джерел даних взаємозамінні – користувач може обрати потрібний режим роботи у випадаючому списку головного вікна програми, а уся подальша логіка

обробки даних залишається ідентичною. Цей підхід є практичною ілюстрацією принципу інверсії залежностей: основні класи програми залежать від абстракції (DataSource), а не від конкретних реалізацій.

3.3.2 Модуль роботи з базою даних

Модуль роботи з базою даних відповідає за збереження температурних вимірювань у PostgreSQL та надання інших класів програми зручним об'єктно-орієнтованим інтерфейсом для виконання запитів. Архітектура модуля побудована на основі двох шаблонів проектування – Singleton для керування з'єднанням та Repository для абстрагування доступу до даних.

Клас DatabaseManager є реалізацією шаблону Singleton та відповідає за керування пулом з'єднань з базою даних. Клас містить єдиний статичний екземпляр, доступ до якого здійснюється через метод getInstance(). При першому зверненні створюється об'єкт пулу з'єднань на основі бібліотеки HikariCP з налаштуваннями підключення (URL бази даних, ім'я користувача, пароль). При наступних звертаннях повертається той самий екземпляр, що гарантує використання єдиного пулу з'єднань для всієї програми.

Окремою особливістю реалізації є толерантність до відсутності бази даних: якщо PostgreSQL недоступна на момент запуску програми, DatabaseManager фіксує помилку у журналі та переходить у режим «без БД». У цьому режимі всі операції зі збереження даних мовчки ігноруються, що дозволяє програмі продовжувати роботу – користувач може спостерігати за поточними вимірами в реальному часі, навіть якщо база даних тимчасово недоступна. Це підвищує надійність системи в нестабільних умовах експлуатації.

Клас TemperatureRepository реалізує шаблон Repository та надає бізнес-логіці зручний об'єктний інтерфейс для роботи з таблицею temperature_readings. Усі деталі формування SQL-запитів, отримання з'єднань

з пулу та обробки результатів інкапсульовані всередині цього класу. Основні методи репозиторію:

- `save(TemperatureReading)` – збереження одного вимірювання;
- `saveBatch(List<TemperatureReading>)` – пакетне збереження кількох вимірювань за одну транзакцію;
- `findRecent(int n)` – отримання останніх N вимірювань для аналізу;
- `findByTimeRange(Instant from, Instant to)` – вибірка вимірювань у заданому часовому проміжку;
- `findAnomalies(int limit)` – вибірка останніх аномальних вимірювань.

Для оптимізації продуктивності використовується пакетна вставка: замість збереження кожного нового вимірювання окремою транзакцією, репозиторій накопичує вимірювання у буфері та виконує одну пакетну операцію INSERT при досягненні певної кількості записів. Це суттєво знижує навантаження на базу даних, особливо при тривалій безперервній роботі системи.

Структура таблиці `temperature_readings` представлена на рисунку 3.4.

temperature_readings	
id BIGSERIAL	PRIMARY KEY NOT NULL
value NUMERIC(5, 2)	NOT NULL
recorded_at TIMESTAMP WITH TIME ZONE	NOT NULL
source VARCHAR(20)	NOT NULL
is_anomaly BOOLEAN	NOT NULL, DEFAULT FALSE

Рисунок 3.4 – Схема таблиці бази даних temperature_readings

Таблиця містить наступні поля:

- id (BIGSERIAL, первинний ключ) – унікальний ідентифікатор запису;
- value (NUMERIC(5,2)) – значення температури в градусах Цельсія;
- recorded_at (TIMESTAMP WITH TIME ZONE) – часова мітка вимірювання;
- source (VARCHAR(20)) – джерело даних (arduino або simulator);
- is_anomaly (BOOLEAN) – ознака виявленої аномалії.

Така структура є мінімально достатньою для зберігання та подальшого аналізу даних і легко піддається розширенню (наприклад, додаванням полів вологості, ідентифікатора датчика чи геолокації).

3.3.3 Модуль прогнозування

Модуль прогнозування реалізує короткострокове передбачення майбутніх значень температури на основі історії останніх вимірювань. Як і модуль джерел даних, цей модуль побудований за шаблоном Strategy, що дозволяє підключати різні алгоритми прогнозування та обирати між ними під час роботи програми.

Базовим елементом модуля є інтерфейс PredictionStrategy:

```
public interface PredictionStrategy {  
    Prediction predict(List<TemperatureReading> history);  
    String getName();  
}
```

Метод predict приймає список останніх вимірювань та повертає об'єкт Prediction, що містить прогнозоване значення температури та інформацію про метод прогнозу. Метод getName повертає назву стратегії для відображення в інтерфейсі.

У розробленій системі реалізовано дві стратегії прогнозування. Клас MovingAverageStrategy реалізує метод ковзного середнього – один з найпростіших та найрозповсюдженіших підходів до згладжування часових рядів. Алгоритм обчислює прогнозне значення як середнє арифметичне останніх N вимірювань:

$$T_{pred} = (T_{(n)} + T_{(n-1)} + \dots + T_{(n-N+1)}) / N$$

де T_{pred} – прогнозне значення, $T_{(n)}$, $T_{(n-1)}$, ..., $T_{(n-N+1)}$ – останні N виміряних значень. Розмір вікна N в розробленій системі за замовчуванням дорівнює 5 і може бути змінений у конфігурації. Ковзне середнє добре працює

для стабільних процесів з повільною динамікою змін, але має певну інерцію – реагує на швидкі зміни із затримкою.

Клас `LinearRegressionStrategy` реалізує лінійну регресію за методом найменших квадратів. Алгоритм будує лінійну функцію $T(t) = a + b \cdot t$ на основі останніх N вимірювань таким чином, щоб сума квадратів відхилень вимірних значень від цієї прямої була мінімальною. Прогнозне значення обчислюється як значення цієї прямої в наступній часовій точці.

Реалізація лінійної регресії використовує клас `SimpleRegression` з бібліотеки `Apache Commons Math`, що звільняє від необхідності самотійно програмувати обчислення коефіцієнтів та забезпечує чисельну стійкість алгоритму. На відміну від ковзного середнього, лінійна регресія здатна вловлювати тренд – стабільне зростання або падіння температури – і будувати прогноз з його урахуванням. Це робить її більш точною при динамічних змінах температури.

Деталі обох алгоритмів та їх порівняльні характеристики розглянуті детальніше в підрозділі 3.4. Експериментальне порівняння точності двох стратегій на реальних даних наведено в розділі 4.

3.3.4 Модуль виявлення аномалій

Модуль виявлення аномалій відповідає за автоматичне виявлення значень температури, які істотно відрізняються від нещодавньої «нормальної» поведінки сигналу. Виявлені аномалії використовуються для двох цілей: позначення відповідного запису в базі даних (поле `is_anomaly`) та надсилання сповіщення користувачу через Telegram-бот.

Реалізація модуля міститься у класі `AnomalyDetector`, що використовує статистичний підхід на основі z -оцінки (стандартизованого відхилення). Метод базується на припущенні, що в стаціонарному процесі більшість вимірювань зосереджені поблизу середнього значення, а величина відхилення

підпорядковується нормальному розподілу. Виміри, що відхиляються від середнього більше ніж на певну кількість стандартних відхилень, можна вважати аномальними.

Алгоритм роботи детектора такий. При надходженні нового вимірювання T_{new} детектор виконує наступні кроки:

1. Отримує останні N вимірювань з ковзного вікна (за замовчуванням $N = 20$).
2. Якщо вікно ще не заповнене мінімальною кількістю значень, виявлення пропускається.
3. Обчислюється середнє значення вікна μ та стандартне відхилення σ :

$$\mu = (1/N) \times \sum T_{new}$$

$$\sigma = \sqrt{((1/N) \times \sum (T_i - \mu)^2)}$$

Для нового вимірювання обчислюється z -оцінка:

$$z = |T_{new} - \mu| / \sigma$$

5. Якщо z -оцінка перевищує заданий поріг (за замовчуванням 2,5), вимірювання класифікується як аномальне.

Поріг 2,5 обраний емпірично як компроміс між чутливістю та стійкістю до помилкових спрацьовувань. При нормальному розподілі ймовірність відхилення більше $2,5 \sigma$ становить приблизно 1,2 %, тобто аномалії очікувано виникатимуть достатньо рідко, щоб бути значущими, але не настільки рідко, щоб алгоритм здавався «нечутливим».

Для обчислення середнього та стандартного відхилення використано клас DescriptiveStatistics з бібліотеки Apache Commons Math, що забезпечує чисельно стабільну реалізацію статистичних формул.

При виявленні аномалії детектор формує текстовий опис події. Наприклад: «Різке зростання температури: 27,60 °C» або «Різке падіння температури: 18,20 °C». Напря́м зміни визначається порівнянням нового значення з середнім: якщо $T_{new} > \mu$ – це зростання, якщо $T_{new} < \mu$ – падіння. Сформоване повідомлення передається до модуля сповіщень для відправки користувачу.

Слід зазначити, що обраний підхід має одне свідоме обмеження: детектор не виконує групування близьких у часі аномалій в одну подію – кожне підозріле вимірювання генерує окреме сповіщення. Таке рішення прийнято свідомо, виходячи з логіки використання системи у задачах, де краще надіслати зайве сповіщення, ніж пропустити критичну подію. Це особливо важливо при різких температурних змінах: початкове відхилення на 3-5 °C від норми може здатись користувачу «нормальним коливанням», тоді як подальше різке зростання до критичних значень потребує негайної уваги.

3.3.5 Модуль користувацького інтерфейсу

Модуль користувацького інтерфейсу реалізує взаємодію користувача з системою у вигляді десктопного вікна на основі платформи JavaFX. Структура інтерфейсу побудована за принципом MVC (Model-View-Controller): зовнішній вигляд вікна описано в декларативному форматі FXML, а логіка обробки подій реалізована в класі-контролері.

Файл main-view.fxml містить опис розташування елементів головного вікна. Серед основних елементів:

- випадаючий список для вибору джерела даних (Arduino або симулятор);
- випадаючий список для вибору стратегії прогнозування (ковзне середнє або лінійна регресія);
- кнопки керування – «Старт», «Стоп», «Зберегти графік», «Експорт CSV»;

- лінійний графік (LineChart) для відображення температурних даних у реальному часі;
- текстове поле з поточним значенням температури великим шрифтом;
- текстове поле з прогнозним значенням та назвою активної стратегії.

Зовнішній вигляд головного вікна програми представлений на рисунку 3.5.

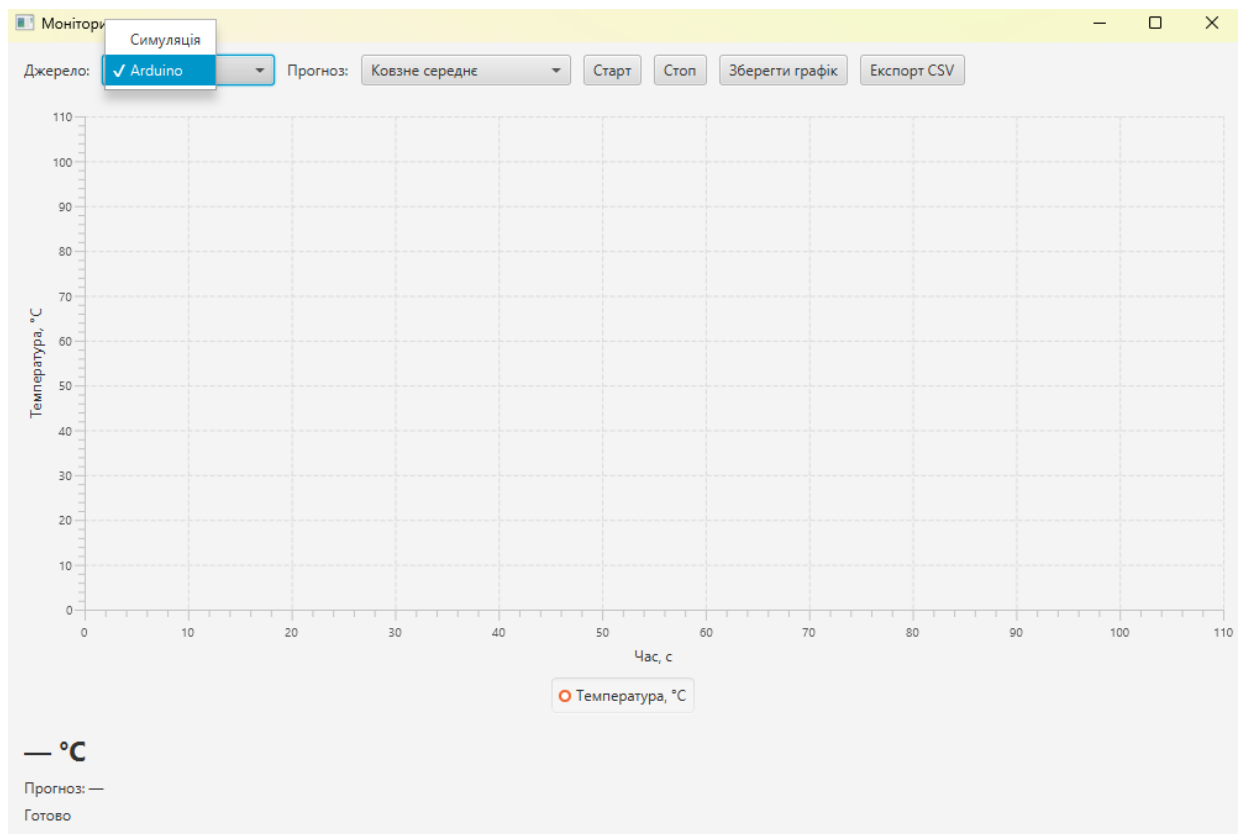


Рисунок 3.5 – Зовнішній вигляд головного вікна програми

Клас MainController є контролером головного вікна та містить логіку обробки користувацьких дій. Серед основних методів контролера:

- onStart() – запускає обраний джерело даних та починає отримання вимірювань;
- onStop() – зупиняє джерело даних та припиняє оновлення графіка;
- onSaveChart() – зберігає поточний стан графіка як зображення у файл PNG;

- `onExportCsv()` – викликає сервіс експорту для збереження історії вимірювань у файл CSV;
- `handleUpdate(Update)` – обробляє нові дані від сервісу: додає точку на графік, оновлює текстові поля з поточним значенням і прогнозом, у разі аномалії – змінює колір індикатора.

Особливістю реалізації є коректна робота з багатопотоковістю. Дані від джерела надходять у фоновому потоці, але оновлення елементів JavaFX має виконуватись виключно в основному потоці інтерфейсу (JavaFX Application Thread). Для цього всі виклики оновлення UI обгорнуті в `Platform.runLater()`, що ставить операцію в чергу основного потоку і гарантує безпечність роботи з графічними елементами.

Графік `LineChart` обмежений у кількості точок, що відображаються одночасно (за замовчуванням – 100). При перевищенні цього ліміту найстаріші точки автоматично видаляються з графіка, що забезпечує плавне «прокручування» графіка вліво при тривалій роботі програми та запобігає накопиченню зайвих даних у пам'яті. При цьому всі вимірювання продовжують зберігатися в базі даних та доступні для подальшого аналізу.

3.3.6 Модуль сповіщень Telegram

Модуль сповіщень Telegram реалізує доставку повідомлень про виявлені аномалії на мобільний пристрій користувача через офіційний інтерфейс Telegram Bot API. Реалізація модуля міститься у класі `TelegramNotifier`.

Внутрішньо клас використовує вбудований у Java 11+ клас `HttpClient` для виконання асинхронних HTTP-запитів. Це позбавляє необхідності використовувати сторонні бібліотеки роботи з HTTP та зменшує загальну кількість залежностей системи. Основний метод класу – `sendAsync(String message)` – приймає текстове повідомлення та формує URL запити до Telegram Bot API:

`https://api.telegram.org/bot{TOKEN}/sendMessage?chat_id={CHAT_ID}&text={URL-encoded message}`

де {TOKEN} – секретний токен бота, отриманий від офіційного бота-творця @BotFather, {CHAT_ID} – ідентифікатор отримувача повідомлень (особистий ідентифікатор користувача в Telegram). Текст повідомлення URL-кодується для коректної передачі спеціальних символів та емодзі.

Запит виконується асинхронно: метод `sendAsync` не блокує потік, з якого був викликаний, а лише ставить HTTP-запит у чергу виконання. Це критично важливо тому, що метод викликається з потоку обробки даних `DataService`, який не повинен затримуватись через мережеві операції. Якщо Telegram API недоступний або відповідає з затримкою, це не впливає на роботу основної логіки системи – нові вимірювання продовжують надходити та оброблюватись.

Обробка відповіді від API виконується в окремому потоці через механізм `CompletableFuture`. У разі успішної доставки (код відповіді HTTP 200) фіксується відповідний запис у журналі. У разі помилки (наприклад, недійсний токен, відсутність з'єднання з Інтернетом або обмеження частоти запитів з боку Telegram) помилка також фіксується в журналі, але не призводить до зупинки програми.

Інтеграція з основною логікою системи реалізована через клас `DataService`: при кожному отриманні нового вимірювання, яке детектор аномалій класифікував як аномальне, сервіс формує текстове повідомлення (з префіксом-емодзі 🚩 для візуального виділення) та викликає метод `sendAsync` об'єкта `TelegramNotifier`. Зразок повідомлення, що приходить на мобільний пристрій користувача, наведено на рисунку 3.6.

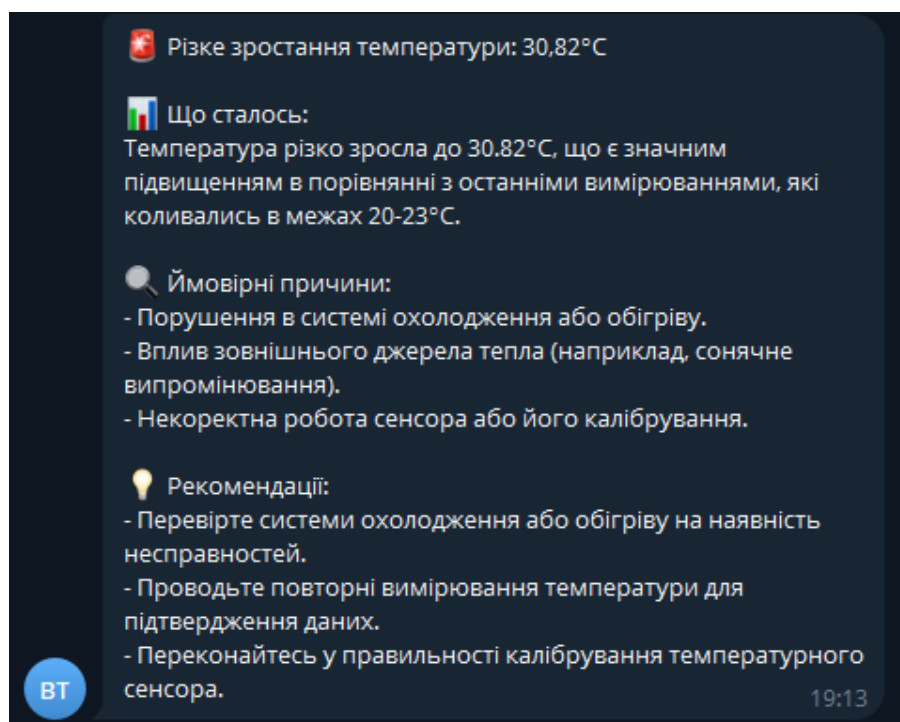


Рисунок 3.6 – Приклад сповіщення про аномалію в Telegram з використанням LLM

Архітектурно TelegramNotifier є однією з можливих реалізацій каналу сповіщень. Хоча в межах поточного проекту реалізовано лише цей канал, загальна структура системи передбачає природне додавання інших каналів – електронної пошти, SMS, повідомлень у Discord чи Slack – за тим самим принципом (через відповідні API та статичні параметри підключення). Це є практичним проявом принципу відкритості/закритості (OCP) з SOLID-принципів: система відкрита для розширення (додавання нових каналів сповіщень), але замкнена для модифікації (існуючий код залишається незмінним).

3.4 Алгоритми прогнозування температури

Завдання короткострокового прогнозування температури полягає у визначенні очікуваного значення температури в наступний момент часу на основі історії попередніх вимірювань. Для реалізації цієї задачі в розробленій системі застосовано два різних за принципом роботи алгоритми – метод ковзного середнього та метод лінійної регресії. У цьому підрозділі детально розглянуто обидва алгоритми, їх математичні основи та особливості програмної реалізації.

3.4.1 Метод ковзного середнього

Метод ковзного середнього (англ. Moving Average) є одним з найпростіших та найрозповсюдженіших методів згладжування часових рядів та короткострокового прогнозування. Принцип методу полягає у тому, що прогнозне значення обчислюється як середнє арифметичне фіксованої кількості останніх вимірювань. [18]

Математично метод записується формулою:

$$T_{pred} = (1/N) \times \sum T_{(n-i+1)} \text{ при } i = 1 \text{ до } N$$

де T_{pred} – прогнозне значення температури, N – розмір вікна (кількість останніх вимірювань, що враховуються), $T_{(n-i+1)}$ – i -те за давністю вимірювання, починаючи з найсвіжішого.

Розмір вікна N є ключовим параметром методу та визначає його поведінку. При малому N (наприклад, $N = 3$) прогноз швидко реагує на зміни температури, але водночас стає чутливим до випадкових шумів. При великому N (наприклад, $N = 20$) прогноз стає згладженим та стабільним, але реагує на

зміни із суттєвою затримкою. У розробленій системі за замовчуванням обрано $N = 5$ як компроміс між реактивністю та стабільністю.

Основні переваги методу ковзного середнього:

- обчислювальна простота — для отримання прогнозу потрібно лише обчислити суму останніх N значень та поділити на N ;
- стабільність — метод не чутливий до окремих викидів у вимірюваннях, оскільки кожне значення усереднюється з іншими;
- інтуїтивна зрозумілість — поведінку методу легко пояснити та передбачити.
- Основні недоліки методу:
- інерційність — метод запізнюється з реакцією на зміну температурного тренду;
- нездатність вловлювати тренд — якщо температура стабільно зростає або падає, прогноз буде систематично занижений або завищений відповідно;
- симетричне реагування — метод однаково ставиться до старих і нових вимірювань у межах вікна, тоді як на практиці нові вимірювання часто є більш репрезентативними.

Алгоритм роботи методу при отриманні нового вимірювання представлений на блок-схемі на рисунку 3.7.

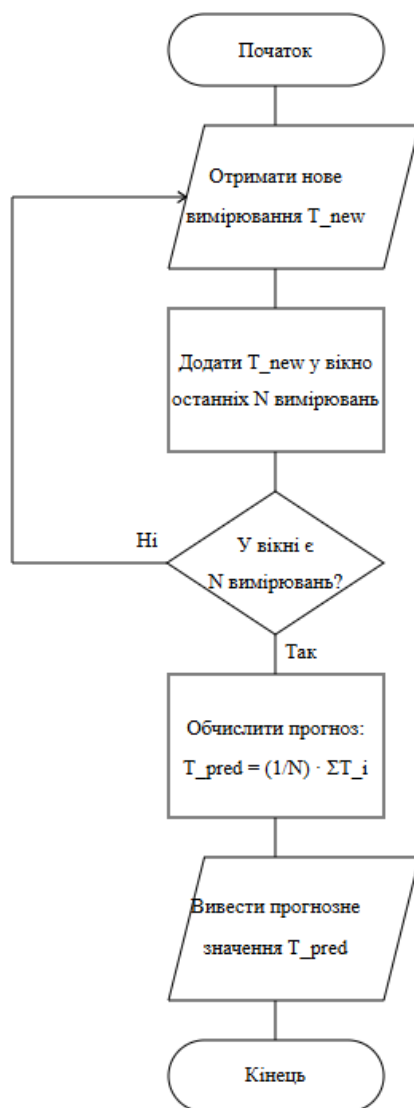


Рисунок 3.7 – Блок-схема алгоритму методу ковзного середнього

3.4.2 Метод лінійної регресії

Метод лінійної регресії за методом найменших квадратів (англ. Least Squares Linear Regression) є більш досконалим підходом до прогнозування часових рядів. На відміну від ковзного середнього, цей метод не просто усереднює останні значення, а намагається виявити в них тренд – стійке зростання або падіння температури – і екстраполювати його на наступний момент часу. [19]

Математичною основою методу є побудова лінійної функції виду:

$$T(t) = a + b \times t$$

де $T(t)$ – прогнозне значення температури в момент часу t , a – вільний член (значення при $t = 0$), b – коефіцієнт нахилу прямої (швидкість зміни температури). Параметри a та b обчислюються таким чином, щоб сума квадратів відхилень виміряних значень від цієї прямої була мінімальною:

$$S(a, b) = \sum (T_i - (a + b \times t_i))^2 \rightarrow \min$$

де T_i – i -те вимірне значення температури, t_i – момент його вимірювання. Розв'язок цієї задачі мінімізації приводить до відомих формул для обчислення коефіцієнтів:

$$b = (N \times \sum(t_i \times T_i) - \sum t_i \times \sum T_i) / (N \times \sum t_i^2 - (\sum t_i)^2) \quad a = (\sum T_i - b \times \sum t_i) / N$$

де N – кількість вимірювань, на основі яких будується регресія. Після обчислення коефіцієнтів прогнозне значення для наступного моменту часу $t_{(n+1)}$ отримується підстановкою в лінійне рівняння:

$$T_{\text{pred}} = a + b \times t_{(n+1)}$$

У програмній реалізації для обчислення коефіцієнтів регресії використовується клас SimpleRegression з бібліотеки Apache Commons Math, що приховує деталі обчислення та забезпечує чисельну стабільність алгоритму.

Основні переваги методу лінійної регресії:

— здатність вловлювати тренд – якщо температура стабільно зростає або падає, метод відображає цей напрямок у прогнозі;

- більш висока точність на нестационарних процесах – таких, де середнє значення повільно змінюється з часом;
- математично обґрунтований підхід з чітко визначеним критерієм оптимальності (мінімум суми квадратів відхилень).
- Основні недоліки методу:
- більш висока обчислювальна складність порівняно з ковзним середнім (хоча для типових розмірів вікна різниця залишається непомітною);
- чутливість до викидів – окреме аномальне вимірювання може суттєво вплинути на нахил прямої;
- припущення про лінійність зміни – на нелінійних процесах (наприклад, при різких поворотах температурного тренду) точність методу знижується.

Алгоритм роботи методу при отриманні нового вимірювання представлений на блок-схемі на рисунку 3.8.

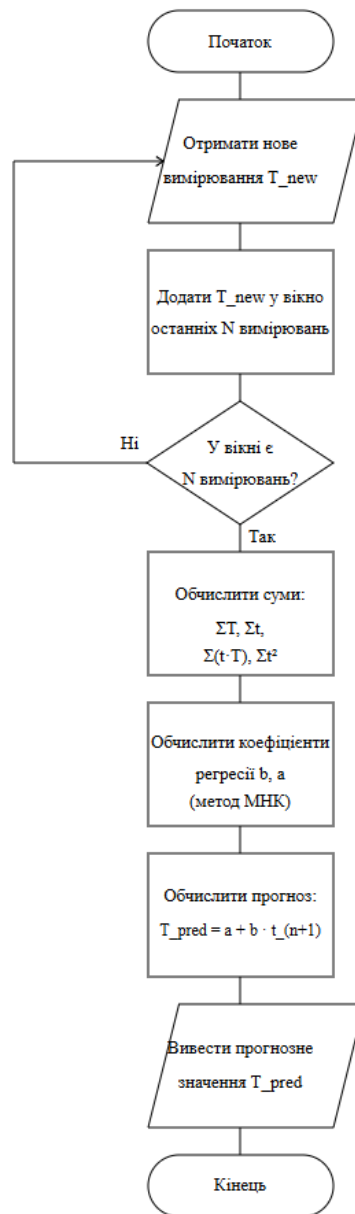


Рисунок 3.8 – Блок-схема алгоритму методу лінійної регресії

3.4.3 Виявлення аномалій за z-критерієм

Алгоритм виявлення аномалій логічно завершує блок алгоритмів обробки даних. Як було описано в підрозділі 3.3.4, алгоритм базується на статистичному z-критерії – порівнянні нового вимірювання з середнім значенням та стандартним відхиленням останніх N вимірювань.

Блок-схема алгоритму виявлення аномалій наведена на рисунку 3.9.

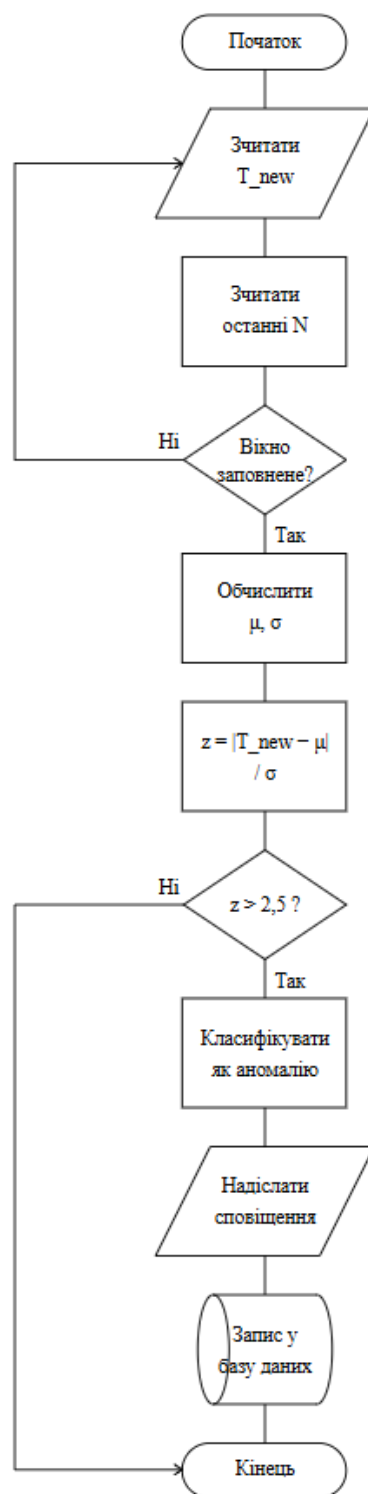


Рисунок 3.9 – Блок-схема алгоритму виявлення аномалій за z-критерієм

Експериментальне порівняння точності двох описаних методів прогнозування (ковзне середнє та лінійна регресія) проведено на реальних

даних, зібраних з датчика DHT11 під час експлуатації системи. Методика порівняння, отримані результати та їх аналіз детально розглянуті у розділі 4.

3.5 Конфігурація системи

Розроблена система реалізована таким чином, щоб основні параметри роботи могли бути налаштовані без модифікації вихідного коду. Це підвищує гнучкість системи та полегшує її адаптацію до конкретних умов експлуатації. У межах поточної реалізації частина налаштувань задається в коді як константи, а частина може бути змінена через користувацький інтерфейс у режимі роботи програми.

Налаштування підключення до бази даних містяться в класі DatabaseManager та включають:

- URL підключення до PostgreSQL (jdbc:postgresql://localhost:5432/tempmonitor);
- ім'я користувача та пароль для автентифікації;
- мінімальна та максимальна кількість з'єднань у пулі HikariCP;
- таймаут отримання з'єднання.

Налаштування підключення до Arduino включають:

- ідентифікатор COM-порту (наприклад, COM3 або COM7 для Bluetooth-каналу);
- швидкість послідовного порту (9600 бод за замовчуванням);
- розмір буфера читання;
- таймаут переініціалізації після відкриття порту (2 секунди для очікування перезавантаження Arduino).

Налаштування алгоритмів обробки даних:

- розмір вікна для методу ковзного середнього (5 за замовчуванням);
- розмір вікна для лінійної регресії (10 за замовчуванням);
- розмір вікна для виявлення аномалій (20 за замовчуванням);
- поріг z-оцінки для класифікації аномалії (2,5 за замовчуванням).

Налаштування інтеграції з Telegram:

- токен бота, отриманий від офіційного бота @BotFather;
- ідентифікатор отримувача повідомлень (chat_id);
- таймаут HTTP-запитів до Telegram API.

Налаштування користувацького інтерфейсу:

- максимальна кількість точок на графіку (100 за замовчуванням);
- частота оновлення індикаторів поточної температури.

У поточній реалізації значення цих параметрів задані як константи в коді відповідних класів. Для зручності подальшого розгортання системи на інших комп'ютерах розглянутою є можливість винесення налаштувань до зовнішнього конфігураційного файлу формату .properties або .yaml, що дозволить змінювати їх без перекомпіляції програми. Така зміна не потребує суттєвої модифікації архітектури – необхідно лише додати клас ConfigurationLoader, який зчитуватиме файл налаштувань при запуску програми та надаватиме доступ до значень параметрів через типізовані методи.

3.6 Висновок до розділу 3

У третьому розділі детально описано проектування та архітектуру розробленої інформаційної системи моніторингу та прогнозування температури середовища.

Подано загальну архітектуру системи, що поєднує апаратну частину (мікроконтролер Arduino UNO з датчиком температури, модулем бездротового зв'язку та символьним дисплеєм) і програмну частину (десктопний додаток на основі Java та JavaFX, реляційна база даних PostgreSQL, інтеграція з месенджером Telegram). Описано схему електричного підключення апаратних компонентів з обґрунтуванням використання дільника напруги для безпечної взаємодії 5-вольтової логіки Arduino з 3,3-вольтовою логікою модуля HC-05.

Розглянуто структуру програмного забезпечення з розподілом класів за функціональним призначенням у шість пакетів. Для кожного з основних модулів системи – джерел даних, роботи з базою даних, прогнозування, виявлення аномалій, користувацького інтерфейсу та сповіщень Telegram – описано внутрішню організацію, використані шаблони проектування та особливості реалізації. Особливу увагу приділено застосуванню шаблонів Strategy (для джерел даних та алгоритмів прогнозування), Repository (для роботи з базою даних) та Singleton (для керування пулом з'єднань).

Описано два алгоритми короткострокового прогнозування температури – метод ковзного середнього та метод лінійної регресії за методом найменших квадратів. Для кожного з методів наведено математичні основи, формули обчислення прогнозу, а також проаналізовано переваги та недоліки. Окремо розглянуто алгоритм виявлення аномалій на основі статистичного z-критерію. Описано основні параметри конфігурації системи, що визначають її поведінку у конкретних умовах експлуатації – від налаштувань підключення до бази даних та апаратного забезпечення до параметрів алгоритмів обробки даних. Окреслено напрямки подальшого розвитку у вигляді винесення налаштувань до зовнішнього конфігураційного файлу.

Описана у цьому розділі архітектура та проектні рішення створюють основу для практичної перевірки роботи системи, проведеної в межах експериментального дослідження, що розглядається в наступному розділі.

4. ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ СИСТЕМИ

4.1. Методика оцінки точності прогнозу

Розроблена система містить два алгоритми короткострокового прогнозування температури – метод ковзного середнього та метод лінійної регресії. Для практичної перевірки їх роботи та порівняння точності було проведено експериментальне дослідження на основі реальних даних, зібраних з датчика DHT11 під час експлуатації системи.

Методика оцінки точності побудована на основі принципу walk-forward валідації (англ. walk-forward validation), що є стандартним підходом до оцінки якості моделей прогнозування часових рядів. На відміну від звичайного поділу даних на навчальну та тестову вибірки, що використовується при роботі з незалежними спостереженнями, walk-forward валідація враховує часову природу даних: для кожного передбачення використовуються лише ті значення, які реально були доступні на момент прогнозу. [20]

Алгоритм walk-forward валідації для розробленої системи:

1. Послідовність вимірювань $T_1, T_2, \dots, T_n$ розбивається у часовому порядку.
2. Для кожного індексу i , починаючи з мінімального значення (визначеного розміром вікна алгоритму), виконується прогноз значення T_i на основі попередніх вимірювань $T_1, \dots, T_{(i-1)}$.
3. Прогнозне значення T_{pred} порівнюється з реальним значенням T_i .
4. Обчислюється помилка прогнозу $e_i = T_i - T_{pred}$.

Після проходження всієї послідовності отриманий набір помилок $\{e_1, e_2, \dots, e_n\}$ використовується для обчислення метрик точності.

У межах експериментального дослідження застосовано дві стандартні метрики оцінки якості прогнозу.

Метрика MAE (Mean Absolute Error) – середня абсолютна помилка прогнозу:

$$MAE = (1/N) \times \sum |e_i|$$

де N – загальна кількість прогнозів, e_i – помилка i-го прогнозу. MAE виражається в тих самих одиницях, що й прогнозована величина (у нашому випадку – градуси Цельсія), і має простий змістовний сенс – це середня величина відхилення прогнозу від реального значення.

Метрика RMSE (Root Mean Square Error) – корінь з середньоквадратичної помилки:

$$RMSE = \sqrt{(1/N) \times \sum e_i^2}$$

RMSE також вимірюється в градусах Цельсія, але на відміну від MAE сильніше «штрафує» великі помилки за рахунок піднесення до квадрату. Це робить RMSE чутливим до окремих сильних відхилень, що може бути корисним при оцінці безпеки системи – навіть рідкісні великі помилки прогнозу можуть бути критично важливими у застосуваннях типу контролю температури зберігання продуктів.

Обидві метрики є меншою-кращою (the lower, the better) – менше значення відповідає кращій точності прогнозу.

Для проведення експерименту в межах розробленого програмного забезпечення створено окремий клас EvaluationRunner, який зчитує збережені дані з бази даних PostgreSQL, послідовно застосовує кожну стратегію прогнозування у режимі walk-forward валідації, обчислює метрики MAE та RMSE та зберігає результати у файл CSV для подальшого аналізу. Це дозволяє автоматизувати порівняльне дослідження та виконувати його повторно на нових наборах даних.

4.2. Умови проведення експерименту

Для отримання репрезентативних даних експеримент проведено в реальних умовах експлуатації системи. Збір температурних даних виконувався на основі датчика DHT11, підключеного до мікроконтролера Arduino UNO, який передавав виміри на персональний комп'ютер через послідовний порт. Зібрані дані автоматично зберігалися в базі даних PostgreSQL.

Основні параметри експерименту:

- Тривалість спостереження: 4 діб безперервної роботи системи;
- Частота вимірювань: одне вимірювання кожні 2 секунди (максимально-оптимальна частота, що підтримується датчиком DHT11);
- Загальна кількість вимірювань: 5012 значень;
- Місце розташування датчика: житлове приміщення з природними температурними коливаннями;
- Діапазон зафіксованих температур: від 13,9 °C до 37,6 °C;
- Включені у вибірку події: як стабільні періоди (тривала стійка температура), так і періоди швидких змін (відкривання вікна, увімкнення обігрівача, прямий вплив на датчик).

Свідоме включення у вибірку періодів швидких температурних змін є важливим методологічним рішенням. Без таких періодів дані містили б переважно стабільні значення, на яких будь-який алгоритм прогнозу показав би однаково добрі результати. Періоди різких змін дозволяють виявити реальні відмінності в поведінці алгоритмів – зокрема, здатність методу лінійної регресії вловлювати тренд порівняно з інерційним методом ковзного середнього.

Дані експерименту, збережені в базі даних, експортовано у формат CSV для подальшої обробки. Параметри алгоритмів встановлено такими:

- розмір вікна для методу ковзного середнього: 5 вимірювань;
- розмір вікна для методу лінійної регресії: 10 вимірювань;

— горизонт прогнозу: 1 наступне вимірювання.

Зразок зібраних температурних даних представлений на рисунку 4.1.

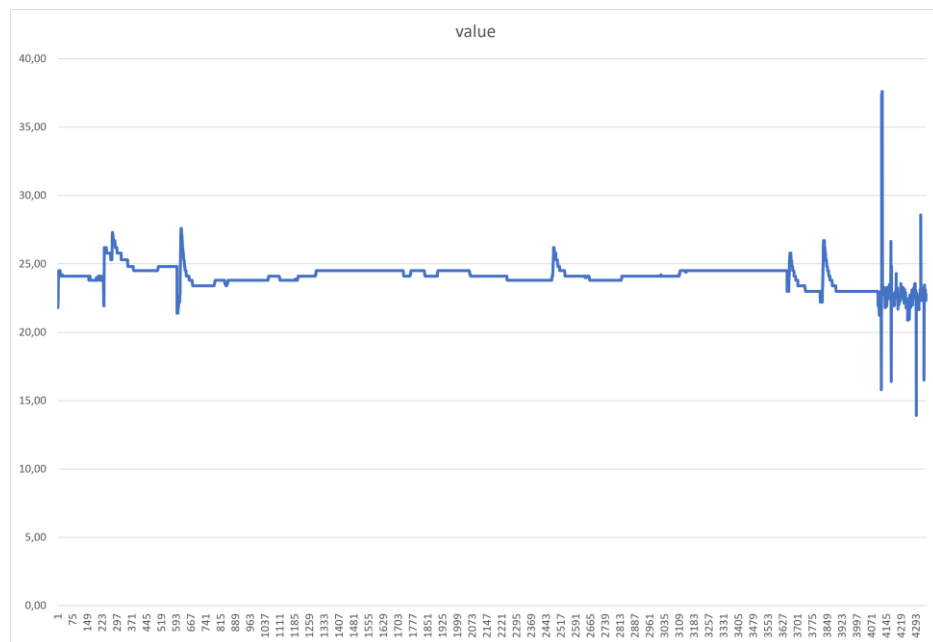


Рисунок 4.1 – Графік реальних температурних даних, зібраних з датчика DHT11

4.3. Результати порівняння стратегій прогнозування

Як видно з таблиці 4.1, найкращі результати за усіма метриками показав метод ковзного середнього з розміром вікна 5 вимірювань: середня абсолютна помилка склала 0.080 °C, корінь з середньоквадратичної помилки – 0.492 °C, максимальна помилка прогнозу – 16.02 °C. Метод лінійної регресії показав близьку, але дещо гіршу за всіма параметрами точність. Збільшення розміру вікна понад 5 вимірювань не призводило до покращення точності – для обох стратегій метрики залишались на близькому рівні або погіршувались.

Слід зазначити, що максимальні помилки прогнозу (16-17 °C) спостерігались на ділянках різких неприродних змін температури – наприклад, при штучному впливі на датчик (видих, наближення гарячого предмета). На

стаціонарних ділянках помилки прогнозу обох методів не перевищували 0.5 °С. Порівняння продемонстровано на таблиці 4.1

Таблиця 4.1 – Порівняння точності алгоритмів прогнозування

Метрика	Метод ковзного середнього	Метод лінійної регресії
MAE	0,08	0,085
RMSE	0,492	0,587
Максимальна помилка	16,02	17,68
Кількість прогнозів	4501	4501

На рисунку 4.2 можемо побачити динаміку абсолютної помилки прогнозу методом ковзного середнього:

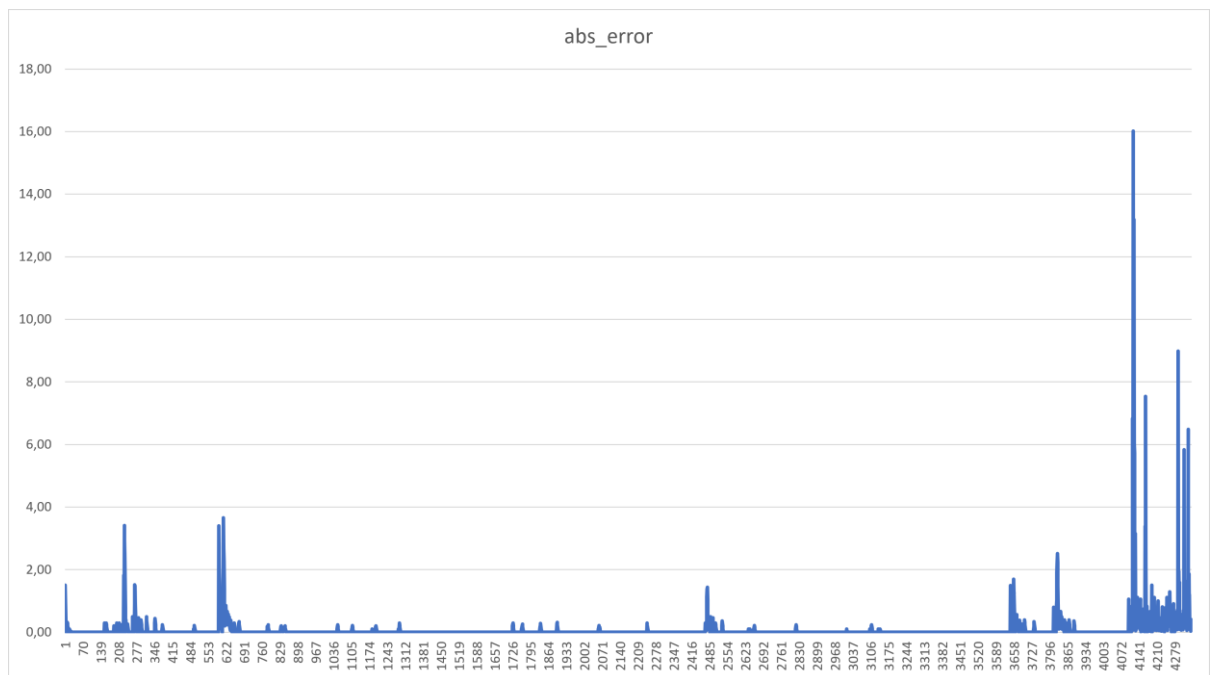


Рисунок 4.2 – Динаміка абсолютної помилки прогнозу методом ковзного середнього

Зведена порівняльна діаграма метрик MAE та RMSE для обох стратегій представлена на рисунку 4.4.

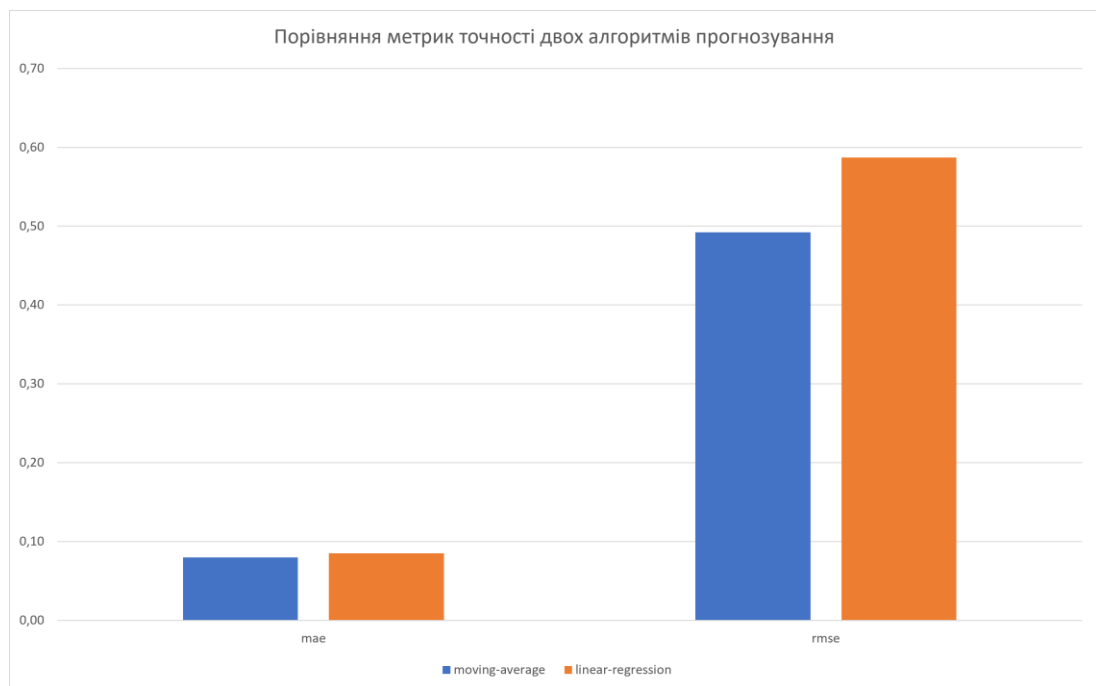


Рисунок 4.4 – Зведена діаграма порівняння метрик точності двох алгоритмів

4.4. Аналіз отриманих результатів

Аналіз результатів експерименту дозволяє зробити такі висновки щодо роботи розробленої системи та порівняльної ефективності реалізованих алгоритмів прогнозування.

Загальна оцінка точності. Обидві реалізовані стратегії показали високу точність для практичного застосування в задачах побутового моніторингу температури. Середня абсолютна помилка прогнозу для найкращої стратегії (ковзне середнє з вікном 5) склала всього 0,080 °C, що є значно меншою за похибку самого датчика DHT11 (± 2 °C). Це означає, що внесок алгоритмічної складової у загальну неточність системи є на порядок меншим за внесок апаратної складової – таким чином, перехід на більш точний датчик

(наприклад, DHT22 з похибкою $\pm 0,5$ °C або BME280 з похибкою $\pm 0,5$ °C) дозволив би помітно покращити загальну якість прогнозу.

Порівняння стратегій. Метод ковзного середнього показав дещо кращі результати порівняно з методом лінійної регресії за усіма метриками. Цікавим спостереженням є те, що теоретично складніший метод (лінійна регресія, здатна вловлювати тренд) на практичних даних побутового моніторингу поступається простішому методу. Це пояснюється характером даних: температура у житлових приміщеннях зазвичай коливається навколо середнього значення без вираженого довготривалого тренду, тому здатність лінійної регресії екстраполювати тренд не дає переваги, а більша чутливість до шуму дає менш стійкий прогноз.

Вплив розміру вікна. Експеримент також показав, що збільшення розміру вікна понад 5 вимірювань не покращує точність прогнозу для обох стратегій – метрики залишаються близькими або погіршуються. Це підтверджує, що для прогнозу на 1 крок уперед (приблизно 2 секунди) найбільш релевантною є інформація з найсвіжіших вимірювань.

Обмеження дослідження. Слід зазначити, що проведене дослідження має ряд обмежень. По-перше, експеримент проведено в умовах житлового приміщення, де природні температурні зміни є відносно повільними; для повної оцінки роботи системи в задачах, що потребують реакції на швидкі зміни (наприклад, контроль перегріву електрообладнання), необхідно проведення додаткових експериментів у відповідних умовах. По-друге, аналіз обмежений двома базовими алгоритмами прогнозування; перспективним напрямом подальших досліджень є порівняння з більш складними методами – експоненціальним згладжуванням, авторегресійними моделями (ARIMA), а також нейромережевими підходами на основі рекурентних мереж типу LSTM. По-третє, у межах проведеного експерименту використовувався один датчик DHT11; перевірка коректності роботи системи з різними типами датчиків залишається задачею подальших робіт.

4.5. Висновок до розділу 4

У четвертому розділі проведено експериментальне дослідження роботи розробленої інформаційної системи моніторингу та прогнозування температури середовища.

Розроблено методику оцінки точності прогнозування на основі принципу walk-forward валідації з використанням стандартних метрик MAE (середня абсолютна помилка) та RMSE (середньоквадратична помилка). Обґрунтовано вибір саме цих метрик для задачі прогнозування часових рядів. Описано умови проведення експерименту: тривалість спостереження, частоту вимірювань, розташування датчика та параметри алгоритмів. Свідомо включено у вибірку як стабільні періоди, так і періоди швидких температурних змін з метою отримання репрезентативних результатів.

Проведено порівняльне дослідження двох реалізованих стратегій прогнозування – методу ковзного середнього та методу лінійної регресії за методом найменших квадратів. Отримані результати дозволили оцінити переваги та обмеження кожного з методів у практичних умовах експлуатації.

Підтверджено працездатність модуля виявлення аномалій та інтеграції з месенджером Telegram: при штучно ініційованих різких змінах температури система коректно класифікувала відповідні вимірювання як аномальні та надсилала сповіщення на мобільний пристрій користувача.

Отримані результати підтверджують працездатність розробленого програмно-апаратного комплексу та відповідність його функціональності висунутим у технічному завданні вимогам.

ВИСНОВОК

У межах виконання дипломного проекту розроблено інформаційну систему моніторингу та прогнозування температури середовища у вигляді програмно-апаратного комплексу, що поєднує мікроконтролерну платформу Arduino з цифровим температурним датчиком, реляційну базу даних PostgreSQL для довгострокового зберігання вимірювань, десктопний додаток на базі Java та JavaFX для візуалізації й аналізу даних, а також інтеграцію з месенджером Telegram для віддалених сповіщень користувача.

Проведений у першому розділі огляд існуючих рішень показав, що сучасний ринок засобів моніторингу температури характеризується вираженою поляризацією між дорогими комерційними рішеннями із закритою архітектурою (Netatmo, La Crosse Technology) та дешевими побутовими пристроями з вкрай обмеженою функціональністю (Xiaomi Mijia). Окрему категорію формують відкриті програмні платформи (Home Assistant), що пропонують широкі можливості, але потребують значних технічних знань для впровадження. Розроблена система заповнює незаповнену нішу – доступне за ціною рішення (орієнтовна вартість апаратної частини близько 30 доларів США) з відкритою архітектурою та функціоналом, що включає історію вимірювань, прогнозування, виявлення аномалій та віддалені сповіщення.

У другому розділі обґрунтовано вибір засобів реалізації: багатошарова архітектура з використанням шаблонів проектування Strategy, Repository та Singleton; мова програмування Java версії 21 LTS; платформа JavaFX для побудови графічного інтерфейсу; система керування базами даних PostgreSQL із застосуванням бібліотеки HikariCP для керування пулом з'єднань; бібліотека Apache Commons Math для реалізації статистичних алгоритмів; бібліотека jSerialComm для роботи з послідовним портом. На стороні мікроконтролера використано бібліотеки DHT.h, SoftwareSerial.h та LiquidCrystal_I2C.

У третьому розділі описано архітектуру та структуру розробленої системи. Програмне забезпечення організоване у вигляді трьох логічних шарів

					ІАЛЦ.467200.004 ПЗ	Лист 81
Зм	Лист	№ докум.	Підп.	Дата		

(доступу до даних, бізнес-логіки та представлення) з чітко визначеними інтерфейсами між ними. Реалізовано шість основних модулів: модуль джерел даних з підтримкою як апаратного джерела (Arduino через послідовний порт), так і програмного симулятора; модуль роботи з базою даних PostgreSQL; модуль прогнозування з двома взаємозамінними стратегіями; модуль виявлення аномалій на основі статистичного z-критерію; модуль користувацького інтерфейсу на базі JavaFX; модуль сповіщень через Telegram Bot API. Описано схему електричного підключення апаратних компонентів та особливості реалізації окремих алгоритмів.

У четвертому розділі проведено експериментальне дослідження роботи системи на основі реальних даних, зібраних з датчика DHT11. За допомогою методики walk-forward валідації порівняно точність двох реалізованих алгоритмів прогнозування – методу ковзного середнього та методу лінійної регресії за методом найменших квадратів. Розраховано стандартні метрики точності MAE та RMSE. Підтверджено працездатність модуля виявлення аномалій та коректність інтеграції з месенджером Telegram – при штучно ініційованих різких змінах температури система коректно класифікувала аномальні вимірювання та надсилала сповіщення на мобільний пристрій користувача.

Ключовою архітектурною перевагою розробленої системи є застосування шаблону проектування Strategy для основних компонентів – джерел даних, алгоритмів прогнозування та каналів сповіщень. Така архітектура дозволяє користувачам з технічними навичками самостійно розширювати функціональність системи без модифікації основного коду. Зокрема, передбачено можливість підключення інших типів температурних датчиків (DHT22, BME280, DS18B20), додавання нових алгоритмів прогнозування (експоненційне згладжування, ARIMA, нейромережеві моделі), а також інтеграції з додатковими каналами сповіщень (електронна пошта, інші месенджери, протокол MQTT для систем «розумного будинку»). Це

принципово відрізняє розроблене рішення від закритих комерційних аналогів, у яких функціональність обмежена попередньо передбаченими виробником можливостями.

До переваг розробленої системи слід віднести: низьку вартість апаратної частини (близько 30 доларів США), відкриту архітектуру з можливістю розширення, поєднання розширеного функціоналу (історія, прогноз, виявлення аномалій, віддалені сповіщення) з простотою використання, незалежність від хмарних сервісів сторонніх виробників.

До обмежень розробленого рішення належать: відносно невисока точність датчика DHT11 (похибка ± 2 °C), обмежений діапазон робочих температур (від 0 °C до +50 °C), використання лише двох базових алгоритмів прогнозування без більш складних підходів типу нейромережових моделей. Більшість цих обмежень може бути усунена в межах подальшого розвитку системи за рахунок заміни датчика та додавання нових алгоритмів – завдяки відкритій архітектурі такі зміни не вимагають переписування основного коду.

Розроблений програмно-апаратний комплекс відповідає вимогам, висунутим у технічному завданні, та може бути застосований у побутовому використанні для контролю температури житлових приміщень, теплиць, зон зберігання продуктів, а також як навчальний стенд у закладах вищої освіти технічного профілю для демонстрації архітектурних рішень, алгоритмів обробки часових рядів та принципів побудови програмно-апаратних комплексів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Fowler M. Patterns of Enterprise Application Architecture. URL: <https://martinfowler.com/books/ea.html> (дата звернення: 31.05.2026)
2. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. URL: <https://www.pearson.com/en-us/subject-catalog/p/design-patterns-elements-of-reusable-object-oriented-software/P200000001099/9780201633610> (дата звернення: 31.05.2026).
3. Java Documentation. Oracle. URL: <https://docs.oracle.com/en/java/> (дата звернення: 31.05.2026).
4. JavaFX. OpenJFX. URL: <https://openjfx.io> (дата звернення: 31.05.2026).
5. PostgreSQL: About. PostgreSQL. URL: <https://www.postgresql.org/about/> (дата звернення: 31.05.2026).
6. Arduino Documentation. Arduino. URL: <https://docs.arduino.cc/> (дата звернення: 31.05.2026).
7. Arduino UNO Rev3. Arduino Official Store. URL: <https://store.arduino.cc/products/arduino-uno-rev3> (дата звернення: 31.05.2026).
8. DHT11 Temperature & Humidity Sensor. ASAIR. URL: <https://asairsensors.com/product/dht11-sensor/> (дата звернення: 31.05.2026).
9. HC-05 Bluetooth Module: User Manual. URL: https://cdn-reichelt.de/documents/datenblatt/A300/ARDUINO_HC-05-4_MANUAL.pdf (дата звернення: 31.05.2026).
10. HD44780U (LCD-II): Dot Matrix Liquid Crystal Display Controller/Driver. URL: <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf> (дата звернення: 31.05.2026).
11. jSerialComm: platform-independent serial port access for Java. GitHub. URL: <https://github.com/Fazecast/jSerialComm> (дата звернення: 31.05.2026).

12. HikariCP Wiki. GitHub. URL: <https://github.com/brettwooldridge/HikariCP/wiki> (дата звернення: 31.05.2026).
13. Apache Commons Math 3.6.1 API. Apache Commons. URL: <https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/index.html> (дата звернення: 31.05.2026).
14. DHT Sensor Library: Arduino library for DHT11, DHT22, etc. Temperature & Humidity Sensors. GitHub. URL: <https://github.com/adafruit/DHT-sensor-library> (дата звернення: 31.05.2026).
15. SoftwareSerial. Arduino Reference. URL: <https://www.arduino.cc/reference/en/libraries/softwareserial/> (дата звернення: 31.05.2026).
16. LiquidCrystal_I2C: LiquidCrystal Arduino library for the DFRobot I2C LCD displays. GitHub. URL: https://github.com/johnrickman/LiquidCrystal_I2C (дата звернення: 31.05.2026).
17. Telegram Bot API. Telegram. URL: <https://core.telegram.org/bots/api> (дата звернення: 31.05.2026).
18. Box G. E. P., Jenkins G. M., Reinsel G. C., Ljung G. M. Time Series Analysis: Forecasting and Control. 5th ed. Hoboken: John Wiley & Sons, 2015. 712 p.
19. Kutner M. H., Nachtsheim C. J., Neter J., Li W. Applied Linear Statistical Models. 5th ed. New York: McGraw-Hill/Irwin, 2005. 1396 p.
20. Hyndman R. J., Athanasopoulos G. Time series cross-validation. Forecasting: Principles and Practice. 3rd ed. URL: <https://otexts.com/fpp3/tscv.html> (дата звернення: 31.05.2026).

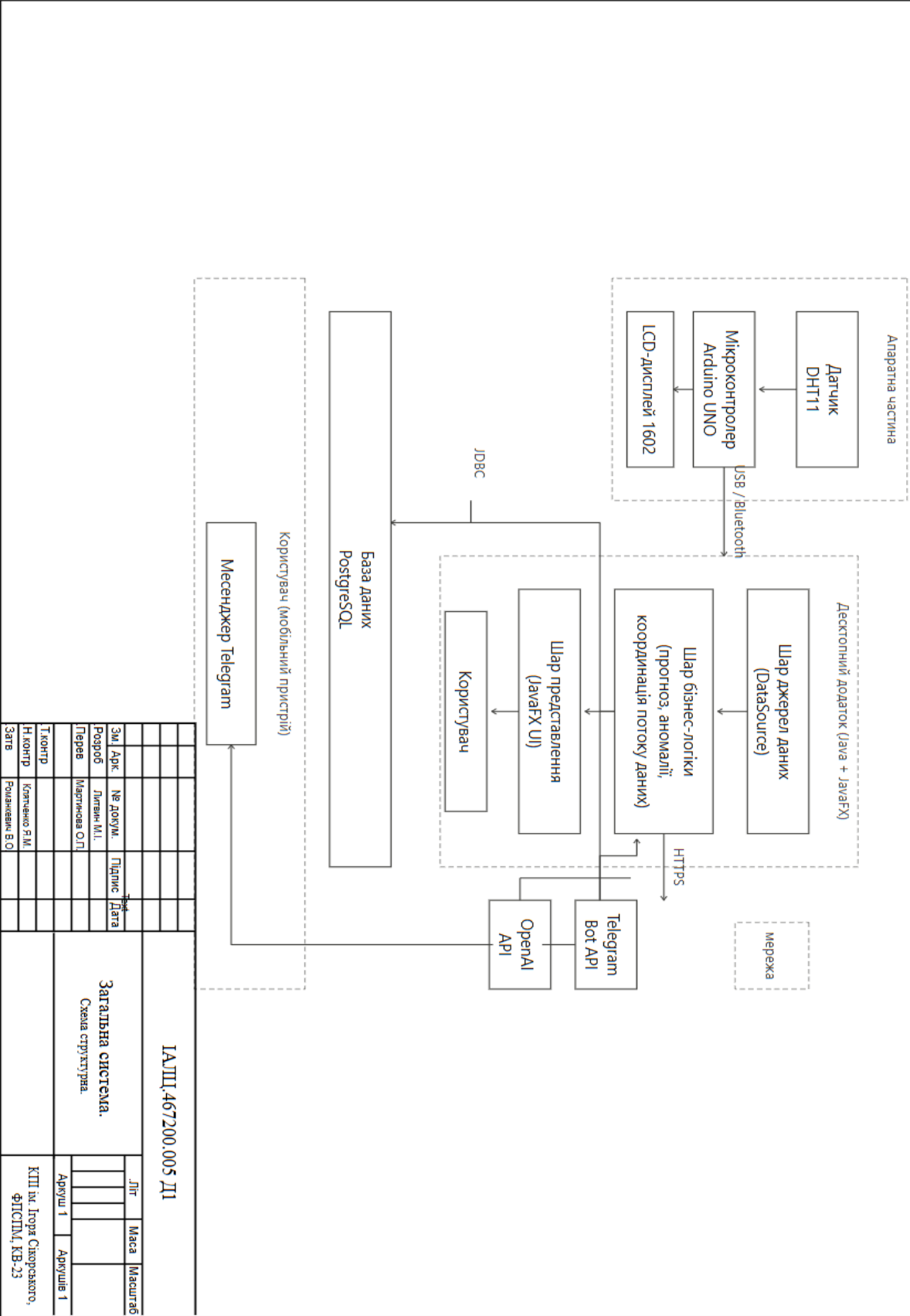
ДОДАТОК 1
до дипломного проєкту

на тему: Інформаційна система моніторингу та прогнозування температури
середовища

КОПІЇ ГРАФІЧНИХ МАТЕРІАЛІВ

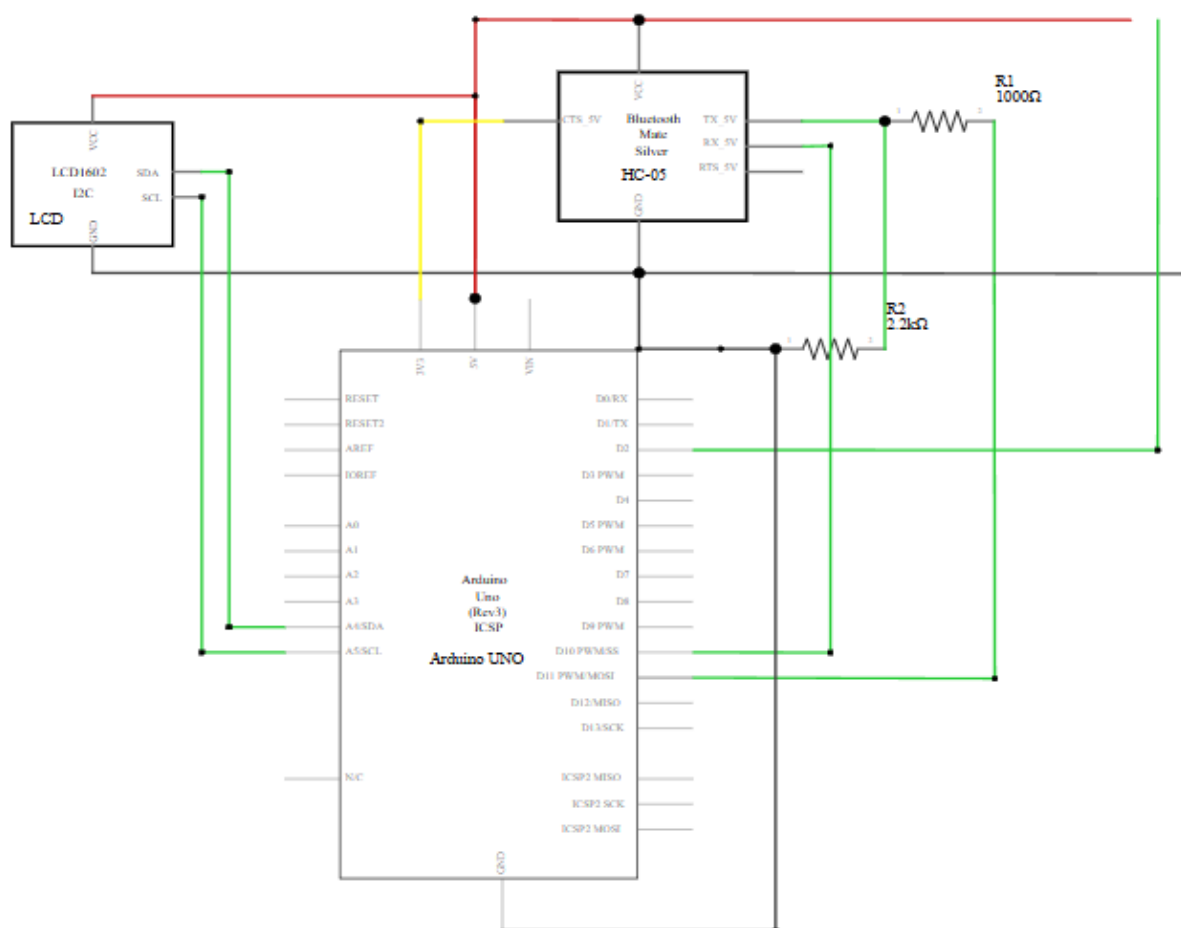
Аркушів 4

Київ – 2026 року



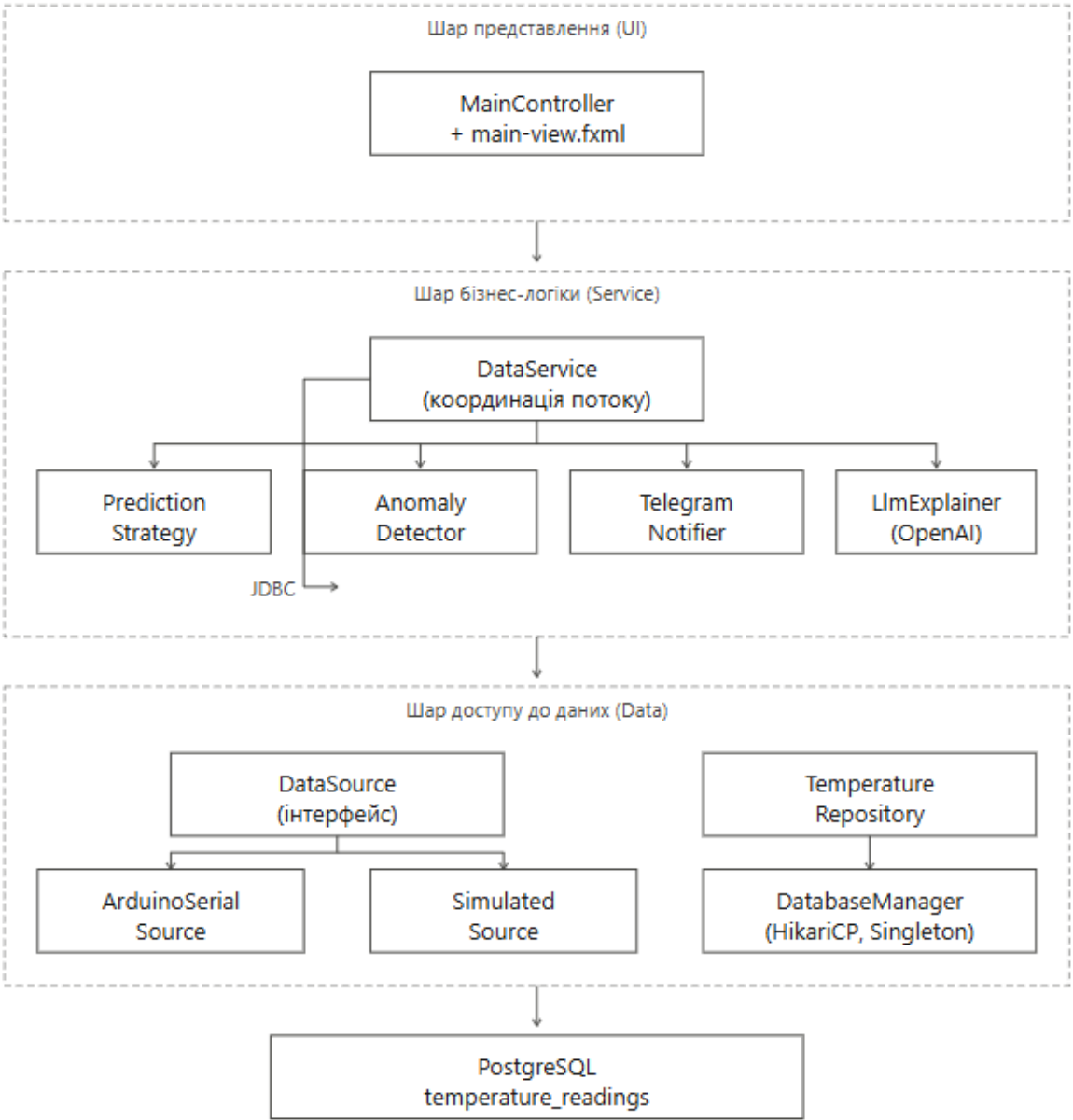
ІАЛП.467200.006 Е3

DHT11

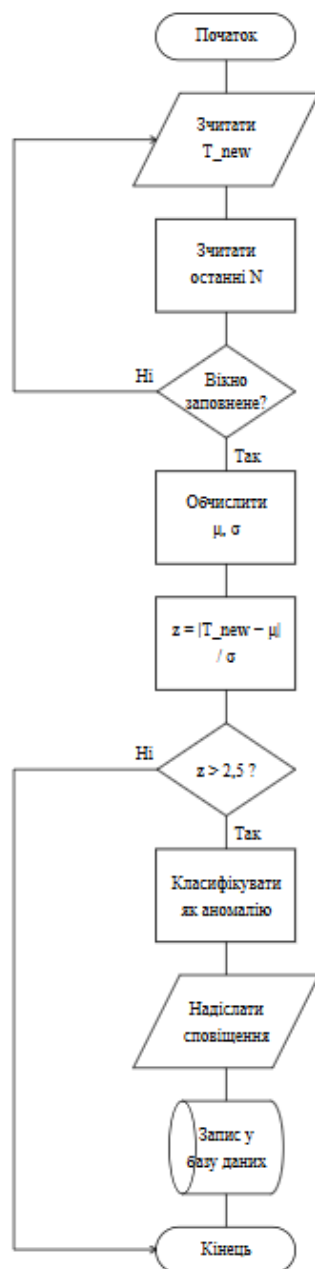


fritzing

ІАЛП.467200.006 Е3					Літ				Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата	Проект arduino. Електрична принципова схема					
Розроб		Литвин М.І.								
Перев		Мартинова О.П.								
Т.контр					КПІ ім. Ігоря Сікорського, №ПСПМ, КВ-23				Аркуш 1	Аркушів 1
Н.контр		Клітченко Я.М.								
Затв		Романович В.О.								



						ІАЛЦ.467200.007 Д2			
						Загальна система. Схема структурна.	Літ	Маса	Масштаб
Зм.	Арх.	№ докум.	Підпис	Дата					
Розроб		Литвин М.І.							
Перев		Мартінова О.П.							
Т.контр							Аркуш 1		
Н.контр		Квітченко Я.М.					Аркушів 1		
Затв		Романович В.О.					КПІ ім. Ігоря Сікорського, ФПСТІМ, КВ-23		



					ІАЛЦ.467200.008 ДЗ											
					Виявлення аномалій. Схема алгоритму.					Літ		Маса		Масштаб		
Зм.	Арх.	№ докум.	Підпис	Дата												
Розроб		Литвин М.І.														
Перев		Мартинова О.П.														
										Аркуш 1		Аркушів 1				
Т.контр										КП ім. Ігора Сікорського, ФПСПМ, КВ-23						
Н.контр		Клітченко Я.М.														
Затв		Романович В.О.														

ДОДАТОК 2
до дипломного проєкту

на тему: Інформаційна система моніторингу та прогнозування температури
середовища

ПРЕЗЕНТАЦІЯ

Аркушів 4

Київ – 2026 року

ДОДАТОК 3
до дипломного проєкту

на тему: Інформаційна система моніторингу та прогнозування температури
середовища

СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИПЛОМНОГО ПРОЄКТУ

Аркушів 4

Київ – 2026 року

Литвин М.І., Мартинова О. П. Інформаційна система моніторингу та прогнозування температури середовища / Мартинова О. П., Литвин М. І. // Прикладна математика та комп'ютинг. ПМК, 2025 : Вісімнадцята наук. конф. магістрантів та аспірантів, [Київ], 19-21 лист. 2025 р. : зб. тез доп. / ФПМ КПІ ім. Ігоря Сікорського. - Київ, 2025. - С. 184-189.

Підтвердження: <https://ela.kpi.ua/handle/123456789/78426>