Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:
Система моніторингу вологості і температури приміщення в режимі реального часу. Частина 2. Мобільний застосунок

Виконав: студент 4 курсу
групи 5ПІ-21б спеціальності
121 – Інженерія програмного забезпечення

Д.С. Чістяков
ініціали та прізвище

Керівник: к.т.н., доц. каф. ПЗ О.В. Мельник
ініціали та прізвище

Рецензент: к.т.н., доц. каф. ОТ С.В. Богомолов
ініціали та прізвище

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. О.Н. Романюк
(ініціали та прізвище)
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., проф. О. Н. Романюк
« 21 » березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Чістякову Дмитру Сергійовичу

1. Тема роботи – розробка системи моніторингу вологості і температури приміщення в режимі реального часу. Частина 2. Мобільний застосунок
2. Керівник роботи: Мельник Олександр Васильович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 20 » березня 2025 р. №97.
3. Строк подання студентом роботи 2 червня 2025 р.
4. Вихідні дані до роботи: мобільна платформа .NET MAUI; бібліотеки Plugin.BLE для реалізації Bluetooth-з'єднання та Syncfusion.Maui.Charts для побудови графіків; мікроконтролер Arduino з підключеним сенсором температури і вологості DHT22; модуль бездротової передачі даних ESP32; протокол обміну даними Bluetooth Low Energy (BLE); параметри для візуалізації – значення температури та вологості в реальному часі; конфігурація інтерфейсу – адаптивний дизайн, кнопки керування, індикатори порогових значень, блок графіків; збереження історії вимірювань у локальному сховищі; порогові межі для генерації попереджень; вихідні дані графіки, текстові значення та сповіщення, що оновлюються в реальному часі.
5. Зміст текстової частини: аналіз предметної області та обґрунтування актуальності проєкту; порівняльний аналіз аналогічних рішень для мобільного моніторингу кліматичних параметрів; вибір апаратної та програмної платформи; постановка задачі та визначення вимог до системи; моделювання структури застосунку: діаграма діяльності, блок-схеми алгоритмів, структура даних; розробка інтерфейсу користувача з підтримкою адаптивності, темної/світлої теми та порогових індикаторів; реалізація Bluetooth-з'єднання з Arduino для зчитування даних із сенсора DHT22; побудова графіків

температури та вологості за допомогою бібліотеки Syncfusion.Maui.Char розробка модуля історії вимірювань і функції експорту даних; реалізац системи порогових сповіщень; тестування функціоналу застосунк підготовка інструкції користувача.

6. Перелік ілюстративного матеріалу: діаграми діяльності та блок-схем алгоритмів роботи застосунку; макети екранів мобільного інтерфейсу; графі температури і вологості в реальному часі; таблиці з історією вимірюван скріншоти процесу зчитування, сповіщень, експорту та налаштува інтерфейсу.

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Мельник О.В., к.т.н., доцент каф. ПЗ	24.03.25	12.06.25

8. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області, визначення мети, завдань, обґрунтування вибору апаратної та програмної платформи	25.03.25 – 31.03.25	Вик.
2	Проєктування архітектури застосунку, моделювання алгоритмів, побудова діаграм і блок-схем	01.04.25 – 07.04.25	Вик.
3	Реалізація інтерфейсу, Bluetooth-з'єднання та збору даних з сенсор	08.04.25 – 14.04.25	Вик.
4	Розробка модуля візуалізації, історії вимірювань, налаштування порогів і генерації сповіщень	15.04.25 – 30.04.25	Вик.
5	Тестування системи, написання інструкції користувача,	01.05.25 – 30.05.25	Вик.

Студент _____ (підпис) **Д. С. Чістяков** (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **О. В. Мельник** (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.912.032.26

Чістяков Д. С. Розробка системи моніторингу вологості і температури приміщення в режимі реального часу. Частина 2. Мобільний застосунок: бакалаврська кваліфікаційна робота зі спеціальності 121 «Інженерія програмного забезпечення», освітня програма — інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 61 с.

На укр. мові. Бібліогр.: назв.: 21; рис.: 35; табл.: 2.

У бакалаврській кваліфікаційній роботі розроблено кросплатформену мобільну систему для моніторингу температури та вологості приміщень у реальному часі. Система забезпечує безперервне зчитування показників із сенсорів, підключених до Arduino через Bluetooth, та відображення їх на мобільному пристрої з використанням графічних елементів.

У роботі використано платформу .NET MAUI, бібліотеку Plugin.BLE для Bluetooth-зв’язку, Syncfusion.Maui.Charts для візуалізації та середовище Visual Studio 2022. Реалізовано інтерфейс користувача з підтримкою адаптивного дизайну, темної/світлої тем, порогових сповіщень та історії вимірювань. Застосунок функціонує без необхідності підключення до Інтернету та є відкритим до розширення.

Отримані результати можуть бути застосовані для побудови автономних IoT-рішень у сферах побуту, промисловості або навчального процесу.

Ключові слова: мобільний застосунок, моніторинг, температура, вологість, .NET MAUI, Arduino, Bluetooth, візуалізація, порогові сповіщення.

ABSTRACT

Chistiakov D. S. Development of a system for monitoring the humidity and temperature of a room in real time. Part 2. Mobile application: bachelor's qualification work in the speciality 121 ‘Software Engineering’, educational programme - software engineering. Vinnytsia: VNTU, 2025. 61c.

In Ukrainian. Bibliogr.: titles 21; figs: 35; tables: 2.

In the bachelor's qualification work, a cross-platform mobile system for real-time monitoring of indoor temperature and humidity was developed. The system provides continuous reading of readings from sensors connected to Arduino via Bluetooth and displays them on a mobile device using graphical elements.

The .NET MAUI platform, the Plugin.BLE library for Bluetooth communication, Syncfusion.Maui.Charts for visualisation, and the Visual Studio 2022 environment were used in the work. The user interface is implemented with support for adaptive design, dark/light themes, threshold notifications, and measurement history. The application functions without the need for an Internet connection and is open for expansion.

The results obtained can be used to build autonomous IoT solutions in the areas of everyday life, industry, or education.

Keywords: mobile application, monitoring, temperature, humidity, .NET MAUI, Arduino, Bluetooth, visualisation, threshold alerts.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний розвиток мобільних технологій та інтернету речей (IoT) суттєво розширює можливості для контролю за параметрами навколишнього середовища як у побуті, так і в промисловості. Серед ключових кліматичних показників, що впливають на комфорт, безпеку та енергоефективність, особливе місце займають температура і вологість повітря. Ці кліматичні параметри мають критичне значення в системах опалення та вентиляції, зберіганні чутливих товарів, сільському господарстві, медичних установах, лабораторіях, а також у житлових приміщеннях. Несвоєчасна реакція на відхилення може призвести до погіршення умов перебування людей, втрат матеріальних ресурсів або виходу з ладу обладнання.

З огляду на вищезазначене, виникає необхідність створення інтелектуальних систем моніторингу, що здатні не лише фіксувати показники кліматичних умов у реальному часі, а й оперативно інформувати користувача про критичні зміни. Одним із найбільш зручних рішень є мобільні застосунки, які дозволяють організувати постійний доступ до даних із будь-якого місця, незалежно від часу й географічного положення користувача. Завдяки сучасним платформам, зокрема NET MAUI [1], з’явилася можливість реалізовувати кросплатформенні мобільні застосунки з єдиною кодовою базою, що значно спрощує розробку та обслуговування систем.

Актуальність теми зумовлена зростаючою потребою у зручних, функціональних і масштабованих рішеннях для контролю мікроклімату з використанням мобільних пристроїв. Створення такого застосунку з візуалізацією даних, історією показників та гнучкими налаштуваннями дозволяє задовольнити потреби як окремих користувачів, так і підприємств, що прагнуть автоматизувати моніторинг кліматичних умов.

Мета та завдання дослідження. Метою бакалаврської кваліфікаційної роботи є розробка функціонального кросплатформенного мобільного застосунку

для моніторингу температури та вологості повітря в реальному часі з інтерактивним графічним інтерфейсом, адаптивним дизайном і підтримкою виводу сповіщень при відхиленні від встановлених параметрів.

Для досягнення поставленої мети передбачено виконання таких завдань:

– проаналізувати предметну область і наявні програмні рішення в цій галузі;

– визначити вимоги до системи з урахуванням функціональності, зручності використання та можливості масштабування;

– обґрунтувати вибір середовища розробки, архітектурного підходу та допоміжних бібліотек;

– розробити програмну архітектуру застосунку з урахуванням модульності та кросплатформенності;

– реалізувати основні модулі: підключення до джерела даних, збір температурних і вологісних показників, їх збереження та візуалізацію;

– впровадити функції налаштування порогових значень і відправки сповіщень;

– провести тестування та оцінити ефективність і зручність використання програмного продукту.

Об’єкт дослідження – процес моніторингу мікроклімату приміщень з використанням програмного забезпечення для мобільного моніторингу параметрів мікроклімату в реальному часі.

Предмет дослідження – методи, інструменти та технології збору, обробки, візуалізації та зберігання кліматичних даних із сенсорів температури та вологості та їх відслідковування в режимі реального часу.

Методи дослідження. Під час виконання роботи використано методи структурного аналізу, об'єктно-орієнтованого проєктування, моделювання взаємодії компонентів системи, методи побудови графічного інтерфейсу користувача, а також експериментальні методи – тестування роботи застосунку з симульованими даними.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод організації збору кліматичних даних у мобільному середовищі, що базується на використанні кросплатформенної технології .NET MAUI та дозволяє забезпечити обробку і візуалізацію даних у реальному часі без потреби в складній серверній інфраструктурі, що забезпечує гнучкість і автономність рішення.

2. Вперше запропоновано програмну модель адаптивного порогового сповіщення про критичні відхилення температури та вологості, яка дозволяє користувачу індивідуально встановлювати межі нормальних значень та отримувати повідомлення при їх перевищенні, що підвищує оперативність реагування на зміни кліматичних умов.

3. Вперше реалізовано модуль інтерактивної історії вимірювань із графічним представленням даних у вигляді динамічних графіків, що дає змогу проводити візуальний аналіз змін кліматичних параметрів у часі, підвищуючи інформативність застосунку.

Практичне значення отриманих результатів. Практичне значення отриманих результатів полягає в тому, що на основі теоретичних досліджень розроблений застосунок може бути використаний у різних сферах: житлові приміщення, склади, теплиці, серверні кімнати, лабораторії тощо. Його відкритість до інтеграції з апаратними сенсорами робить можливим подальший розвиток у межах IoT-систем. Водночас, застосунок може бути використаний у навчальному процесі як приклад реалізації прикладного мобільного ПЗ із реальним практичним застосуванням.

Особистий внесок здобувача

Усі наукові результати, викладені у комплексній бакалаврській кваліфікаційній роботі, що стосуються мобільного застосунку, отримані автором особисто. Комплексна бакалаврська кваліфікаційна робота виконана у співпраці з Костюком А. О.

Апробація матеріалів бакалаврської дипломної роботи

Результати бакалаврської кваліфікаційної роботи доповідалися на IV Всеукраїнській науково-технічній конференції молодих вчених, аспірантів і студентів «Комп'ютерні ігри і мультимедіа, як інноваційний підхід до комунікації - 2024», 26-27 вересня 2024 р.

Публікації

Тези доповідей «СИСТЕМА МОНІТОРИНГУ ВОЛОГОСТІ І ТЕМПЕРАТУРИ ПРИМІЩЕННЯ» опубліковані на IV Всеукраїнській науково-технічній конференції молодих вчених, аспірантів і студентів «Комп'ютерні ігри і мультимедіа, як інноваційний підхід до комунікації - 2024», 26-27 вересня 2024 р.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ СИСТЕМИ МОНІТОРИНГУ ТЕМПЕРАТУРИ І ВОЛОГОСТІ ПРИМІЩЕННЯ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ

1.1 Стан технологій моніторингу кліматичних параметрів приміщень

Сучасні технології моніторингу мікроклімату активно застосовуються у побуті, промисловості, агросекторі, на складах і в наукових лабораторіях. Найпоширенішими є системи, які використовують датчики температури та вологості з передачею даних через бездротові інтерфейси, зокрема Bluetooth [2], Wi-Fi або LoRaWAN. Особливо популярними стали рішення з використанням мікроконтролерів Arduino [3], які забезпечують просту інтеграцію з датчиками та мобільними застосунками.

У цій роботі реалізовано підхід, за якого дані зчитуються датчиками температури та вологості, підключеними до Arduino, а передача здійснюється через Bluetooth до мобільного застосунку. Це дозволяє забезпечити реальний зв'язок між фізичними сенсорами та цифровим інтерфейсом користувача без потреби в складній мережевій інфраструктурі.

Попри доступність окремих апаратних і програмних компонентів, більшість існуючих рішень є фрагментованими, обмеженими конкретним виробником або недостатньо адаптованими під мобільні потреби. Внаслідок цього виникає потреба у створенні універсального, кросплатформенного мобільного застосунку, який би підтримував візуалізацію, зберігання, сповіщення та інтерактивну взаємодію з користувачем.

1.2 Порівняльний аналіз аналогів

На ринку присутні різноманітні мобільні застосунки, що забезпечують моніторинг температури і вологості. Одними з найпопулярніших є Mi Home, Govee Home, SensorPush. Усі вони працюють із власними сенсорами, мають власну екосистему та підтримують графічне відображення змін показників.

Mi Home [4] – це мобільний застосунок (рис. 1.1) від компанії Xiaomi, який використовується для керування розумними пристроями з їхнього асортименту, зокрема сенсорами температури та вологості. Програма дозволяє підключатися до таких сенсорів, як Xiaomi Mijia Bluetooth Hygrothermometer, і виводити поточні показники в реальному часі. Вона зберігає історію вимірювань, надсилає сповіщення при виході значень за межі, підтримує створення автоматизованих сценаріїв, наприклад: якщо вологість опускається нижче 40%, автоматично вмикається зволожувач. Для передачі даних Mi Home використовує Bluetooth, Wi-Fi або Zigbee залежно від типу пристрою.

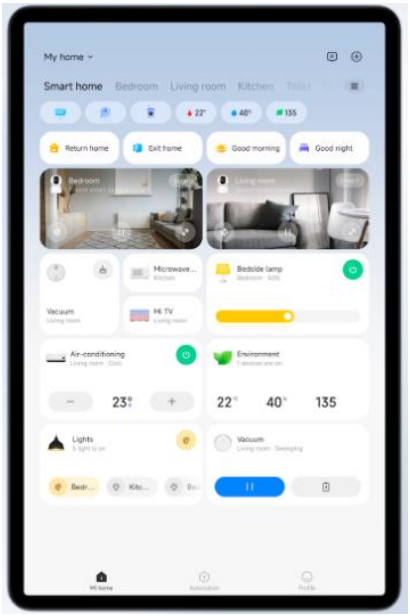

Рисунок 1.1 – Інтерфейс застосунку «Mi Home»

Переваги:

- повна інтеграція з екосистемою Xiaomi;
- стабільна робота з фірмовими датчиками;
- хмарна синхронізація і можливість перегляду з будь-якого пристрою;
- можливість автоматизації розумного будинку.

Недоліки:

– працює тільки з пристроями Xiaomi/Mijia;
– закритий протокол;
– інтерфейс не адаптується під сторонні задачі;
– частина функцій працює тільки через хмару.

Govee Home [5] – це мобільний застосунок (рис. 1.2) для керування сенсорами температури та вологості компанії Govee. Він забезпечує перегляд показників у реальному часі, графіки, збереження історії та сповіщення про відхилення. Працює по Bluetooth навіть без Інтернету, а хмарна синхронізація можлива через шлюз. Підтримуються автоматизовані сценарії, однак застосунок сумісний лише з фірмовими пристроями і не підтримує сторонні рішення.

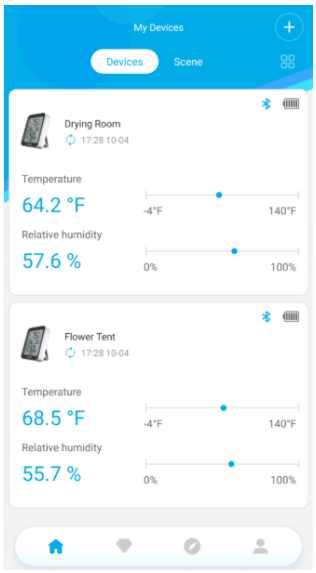

Рисунок 1.2 – Інтерфейс застосунку «Govee Home»

Переваги:

– працює по Bluetooth без Інтернету;
– має графіки, історію, сповіщення;
– підтримка автоматизованих сценаріїв;

– хмарна синхронізація через шлюз.

Недоліки:

– працює лише з пристроями Govee;
– не підтримує Arduino та сторонні сенсори;
– закрита екосистема;
– частина функцій потребує шлюзу.

SensorPush [6] – це мобільний застосунок (рис 1.3) для роботи з високоточними сенсорами температури та вологості однойменного бренду. Він забезпечує моніторинг у реальному часі, зберігання великого обсягу історичних даних, виведення статистики та графіків. Передача даних здійснюється через Bluetooth, а з Wi-Fi-шлюзом можливий хмарний доступ з будь-якої точки. Застосунок має розвинений аналітичний функціонал, однак працює лише з фірмовими пристроями, не підтримує Arduino й орієнтований на професійне використання.

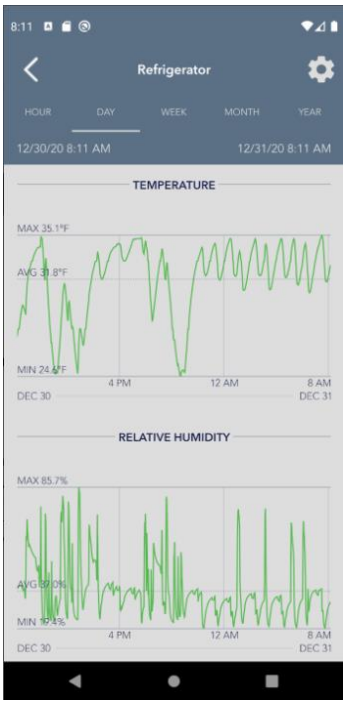

Рисунок 1.3 – Інтерфейс застосунку «SensorPush»

Переваги:

- висока точність вимірювань;
- професійна візуалізація та аналітика;
- хмарний доступ із будь-якої точки;
- зберігання великого обсягу історії.

Недоліки:

- працює лише з власними сенсорами;
- немає підтримки Arduino або кастомних пристроїв;
- обмежена інтеграція;
- орієнтований на професійне використання.

Запропонований у кваліфікаційній роботі застосунок вирізняється тим, що забезпечує пряму взаємодію з апаратною платформою Arduino, отримує реальні дані через Bluetooth, підтримує адаптивний інтерфейс, історію вимірювань, експортування та налаштовувані порогові значення. Це робить його універсальним для освітнього, побутового чи експериментального використання.

Результати порівняння застосунків зображено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій	Mi Home	Govee Home	SensorPush	Власна розробка
Працює з Arduino	-	-	-	+
Кастомізація інтерфейсу	-	-	-	+
Працює без Інтернету	+	+	-	+
Підтримка експорту та історії	-	+	+	+
Адаптація під мобільні пристрої	+	+	-	+

Продовження таблиці 1.1

Критерій	Mi Home	Govee Home	SensorPush	Власна розробка
Підтримка нестандартних сценаріїв	-	-	-	+
Підсумок	2	3	1	6

На відміну від існуючих комерційних застосунків, які мають жорстку прив’язку до власних пристроїв, розроблений у межах кваліфікаційної роботи, мобільний застосунок демонструє вищу гнучкість, адаптивність і універсальність. Він підтримує роботу з апаратною платформою Arduino, що забезпечує вільний вибір сенсорів і просту інтеграцію. Програма не вимагає шлюзів або підключення до Інтернету для базового функціоналу, надає можливість експорту даних і зберігання історії, підтримує зміну зовнішнього вигляду інтерфейсу, а також дозволяє створювати нестандартні сценарії реагування. Це робить її придатною не лише для побутового використання, але й для освітніх, лабораторних та експериментальних цілей, де потрібна максимальна відкритість і налаштовуваність.

Таким чином, розроблений застосунок має низку переваг, зокрема відкритість, гнучкість, незалежність від конкретного виробника та адаптованість до апаратних і програмних потреб користувача.

1.3 Аналіз методів розв’язання задачі

Розв’язання задачі створення мобільного застосунку для моніторингу кліматичних параметрів приміщення вимагало ґрунтовного аналізу можливих технічних рішень. Основними критеріями вибору стали: сумісність із апаратною платформою Arduino, автономність роботи (без постійного доступу до Інтернету), кросплатформеність, адаптивність інтерфейсу, підтримка Bluetooth-з’єднання та розширюваність функціоналу.

Перший розглянутий підхід – використання готових хмарних [7] IoT-платформ, таких як Blynk або ThinkSpeak [8]. Ці сервіси дозволяють швидко під’єднати сенсори та передавати дані на візуальні панелі в браузері або мобільному застосунку. Проте такі платформи залежать від Інтернет-з’єднання, що обмежує їх використання в автономному режимі. Крім того, користувач має обмежений контроль над виглядом інтерфейсу, логікою роботи, частотою опитування даних та не може гнучко налаштовувати інтервал оновлення, зовнішній вигляд графіків чи поведінку сповіщень. Також у таких рішеннях часто відсутня підтримка локального зберігання даних на пристрої [9].

Другий варіант – створення нативного застосунку під Android із використанням Java або Kotlin, який би безпосередньо з’єднувався через Bluetooth із Arduino. Цей підхід дозволяє розробнику реалізовувати логіку передачі, обробки та візуалізації даних самостійно, формувати інтерфейс, обирати період оновлення, визначати умови для сповіщень тощо. Але основним недоліком є те, що застосунок буде обмежений лише платформою Android, і для створення версії для iOS доведеться виконувати окрему реалізацію з нуля, що значно збільшує обсяг робіт.

Оптимальним рішенням виявилася платформа .NET MAUI (Multi-platform App UI) – сучасна кросплатформена технологія від компанії Microsoft, яка дозволяє створювати застосунки одночасно для Android, iOS, Windows та macOS з використанням єдиної кодової бази на мові програмування C#. У контексті кваліфікаційної роботи, вона забезпечує як розширену підтримку Bluetooth, так і можливість гнучко формувати UI з підтримкою темної та світлої тем, адаптацією до розмірів екрана та інтеграцією сторонніх бібліотек, зокрема для побудови графіків (Syncfusion.Maui.Charts).

Використання .NET MAUI у поєднанні з Arduino дозволило досягти таких технічних переваг:

– повний контроль над обміном даними по Bluetooth;
– локальне зберігання історії вимірювань;

– можливість налаштування порогових значень та формування сповіщень;
– адаптивна побудова графіків зі зміною кольорів залежно від значень;
– збереження даних у вигляді текстових звітів;
– просте масштабування функціоналу для майбутніх потреб.

Завдяки цьому застосунок не залежить від хмарних сервісів, не потребує підключення до Інтернету для основної роботи, може бути легко адаптований до нових сенсорів або інших платформ. Такий підхід забезпечує найкращий баланс між гнучкістю реалізації, зручністю використання, візуальною привабливістю та технічною надійністю.

Отже, серед усіх розглянутих методів саме використання .NET MAUI у поєднанні з Arduino та Bluetooth-модулем є найбільш ефективним способом реалізації задачі, що дозволяє створити якісний, функціонально повний і адаптивний мобільний застосунок для моніторингу температури й вологості приміщень у реальному часі.

1.4 Вимоги до функціональності застосунку

Розроблювана система повинна забезпечувати моніторинг температури та вологості приміщення в режимі реального часу з використанням фізичних сенсорів на базі платформи Arduino з передачею даних через Bluetooth. Застосунок має надавати користувачеві зручний інтерфейс для візуалізації показників, керування режимами роботи, збереження історії вимірювань та отримання сповіщень при перевищенні порогових значень.

Для досягнення мети поставлено такі задачі:

– дослідити предметну область та існуючі програмно-апаратні рішення;
– обґрунтувати вибір апаратної бази (Arduino, Bluetooth-модуль, сенсори);
– вибрати інструменти розробки мобільного застосунку;
– спроєктувати архітектуру застосунку та користувацький інтерфейс;

- реалізувати Bluetooth-з'єднання з Arduino та обробку отриманих даних;
- створити модулі візуалізації, експорту, історії даних та сповіщень;
- протестувати роботу застосунку з фізичним пристроєм у реальних умовах

Таким чином, мобільний застосунок має бути гнучким, адаптивним, функціональним та здатним працювати автономно, забезпечуючи повноцінний контроль кліматичних параметрів приміщення через взаємодію з апаратною платформою.

1.5 Висновки

Провівши аналіз предметної області моніторингу кліматичних параметрів приміщення, було сформульовано основні вимоги до програмного забезпечення. Також, було розглянуто поточний стан технологій, визначено недоліки існуючих рішень, таких як Mi Home, Govee Home і SensorPush, а також обґрунтовано доцільність створення власного мобільного застосунку.

Порівняльний аналіз показав, що більшість наявних застосунків є закритими, працюють виключно з власними пристроями та не передбачають кастомізації або розширення функціоналу. На відміну від них, запропонований застосунок забезпечує підтримку відкритої платформи Arduino, має адаптивний інтерфейс, не вимагає доступу до Інтернету та дозволяє реалізовувати індивідуальні сценарії.

У результаті аналізу методів реалізації було обрано технологічну базу .NET MAUI, яка забезпечує кросплатформену підтримку, інтерактивний графічний інтерфейс, можливість Bluetooth-з’єднання та повний контроль над логікою застосунку. Обраний підхід є оптимальним для створення універсального мобільного застосунку, який поєднує гнучкість, автономність та зручність користування.

Таким чином, результати проведеного аналізу стали основою для формування технічного бачення системи та підготовки її реалізації.

2 РОЗРОБКА МОДЕЛЕЙ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

Під час розробки мобільного застосунку для моніторингу температури і вологості було визначено, що необхідно забезпечити підтримку таких функцій, як бездротова передача даних із датчиків Arduino, відображення графіків у режимі реального часу, збереження історії вимірювань, підтримка сповіщень при перевищенні порогів та гнучкий інтерфейс, адаптований до мобільних пристроїв. Виходячи з цих вимог, було обрано технологічну базу .NET MAUI (Multi-platform App UI) [10], яка є кросплатформеним фреймворком нового покоління від компанії Microsoft. Вона дозволяє створювати мобільні застосунки одночасно для Android, iOS та Windows з використанням єдиної кодової бази на мові C#, що суттєво пришвидшує та спрощує підтримку розробки.

.NET MAUI забезпечує високу продуктивність, просту інтеграцію з бібліотеками, можливість гнучкого управління елементами інтерфейсу, а також підтримку роботи з Bluetooth-пристроями. Ці властивості зробили її оптимальним вибором для реалізації розробки [11]. Зокрема, для відображення графіків температури і вологості було інтегровано потужну бібліотеку Syncfusion.Maui.Charts, яка дозволяє будувати лінійні графіки, змінювати кольори в залежності від значень, додавати маркери, підписи осей, легенду та реалізовувати інтерактивну взаємодію з графіком.

Зчитування показників температури і вологості здійснюється з фізичних сенсорів, підключених до мікроконтролера Arduino. Для передачі даних використано модуль Bluetooth ESP32, який послідовно надсилає дані у текстовому форматі. На стороні мобільного пристрою застосунок виконує сканування доступних Bluetooth-пристроїв, встановлює з'єднання з ESP32 і зчитує отримані значення в реальному часі. Для цього використовується бібліотека Plugin.BLE, яка адаптована до .NET MAUI та забезпечує стабільну

роботу з Bluetooth Low Energy пристроями на рівні API, підтримуваному Android і iOS.

У ролі середовища розробки використано Visual Studio 2022 із встановленим MAUI Workload. Це середовище забезпечує повний цикл розробки – від написання коду до його тестування і налагодження безпосередньо на фізичних мобільних пристроях або емуляторах. Visual Studio має потужні засоби налагодження, зокрема відстеження стану Bluetooth-з'єднання, перегляд потоків даних і тестування адаптивності інтерфейсу під різні розміри екранів [12].

Отже, обрана платформа та засоби розробки дозволили забезпечити технічну реалізацію всіх запланованих функцій – від приймання даних із датчиків до побудови гнучкого інтерфейсу з візуалізацією, історією та адаптивністю. Такий вибір є повністю обґрунтованим і відповідає як вимогам технічного завдання, так і сучасним підходам до розробки кросплатформених мобільних застосунків.

2.2 Розробка інтерфейсу програмного додатку

Інтерфейс користувача є ключовим компонентом будь-якого мобільного застосунку, особливо якщо йдеться про візуалізацію динамічних даних. Основним завданням на цьому етапі стало створення логічної, інтуїтивно зрозумілої та адаптивної структури екранів, яка дозволяє користувачеві швидко орієнтуватися в інформації, що надходить із сенсорів температури та вологості.

Основний екран застосунку (рис. 2.1) складається з кількох зон: панелі інструментів з кнопками керування, графічного блоку для відображення температури та вологості у вигляді лінійних графіків, текстових індикаторів поточних значень, а також елементів для вибору інтервалу оновлення, теми оформлення і доступу до історії та налаштувань.

Панель керування містить кнопки для запуску, зупинки збору даних, оновлення графіка, експорту результатів та перегляду історії. Графічний блок дозволяє відстежувати зміни температури і вологості у реальному часі.

Індикатори відображають актуальні значення, змінюючи колір при перевищенні встановлених меж. Налаштування дають змогу змінювати пороги сповіщень та тему оформлення інтерфейсу.

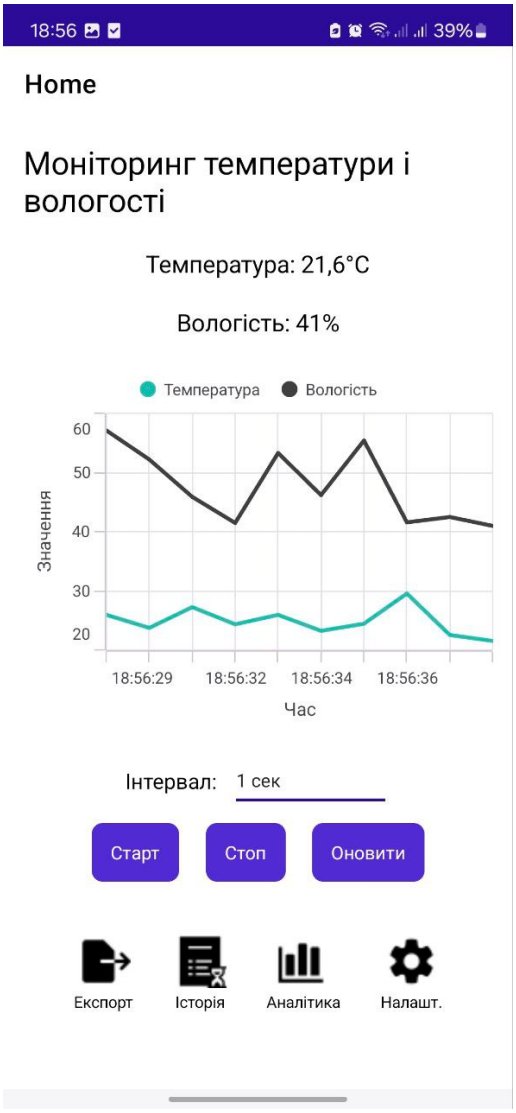

Рисунок 2.1 – Основний екран застосунку

Кнопки синього кольору (рис. 2.2) розміщені у нижній центральній частині екрана призначені для керування основним процесом збору даних. Кнопка «Старт» ініціює зчитування даних із сенсорів, «Стоп» – припиняє обробку сигналу, а «Оновити» дозволяє вручну перезапустити цикл збору. Таке

розміщення забезпечує зручний доступ навіть при використанні пристрою однією рукою.

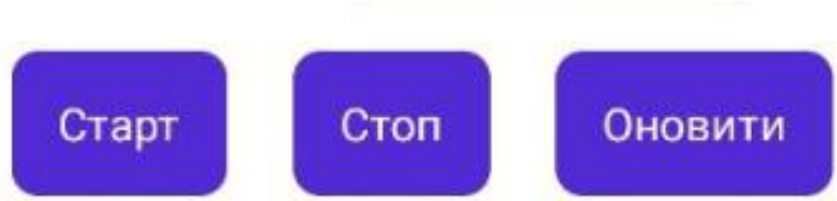

Рисунок 2.2 – Кнопки керування основним процесом збору даних

Основний блок інтерфейсу займає графік (рис. 2.3), побудований на основі бібліотеки Syncfusion.Maui.Charts. Він дозволяє візуалізувати зміни температури та вологості у вигляді лінійних кривих. Графік оновлюється динамічно, масштабовується та супроводжується маркерами даних, що полегшують аналіз змін.

Рисунок 2.3 – Графік показників

Числові значення температури та вологості (рис. 2.4) виводяться безпосередньо над графіками у великому шрифті. Значення автоматично

оновлюються в реальному часі. При перевищенні встановлених порогів текст змінює колір, сигналізуючи користувачу про критичні умови у приміщенні.

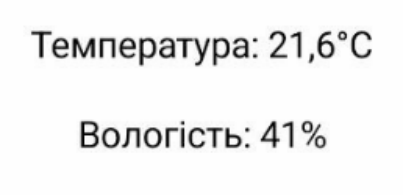

Рисунок 2.4 – Відображення температури і вологості

Користувач може обирати частоту зчитування нових даних із сенсора. Інтервали задаються через випадаючий список (рис. 2.5) і застосовуються до фонової служби збору даних. Це дозволяє балансувати між точністю і навантаженням на систему.

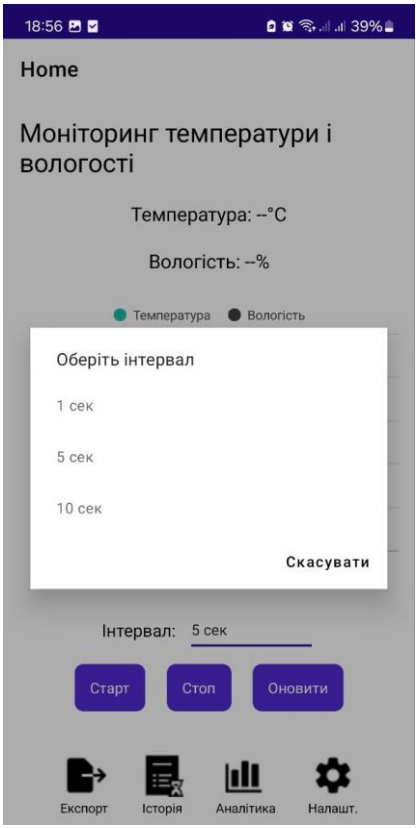

Рисунок 2.5 – Вибір інтервалу оновлення

Інтерфейс підтримує зміну теми – темну або світлу. Перемикач доступний у меню налаштувань (рис. 2.6). Усі компоненти автоматично адаптуються до вибраної схеми кольорів, що підвищує комфорт користування в різний час доби.

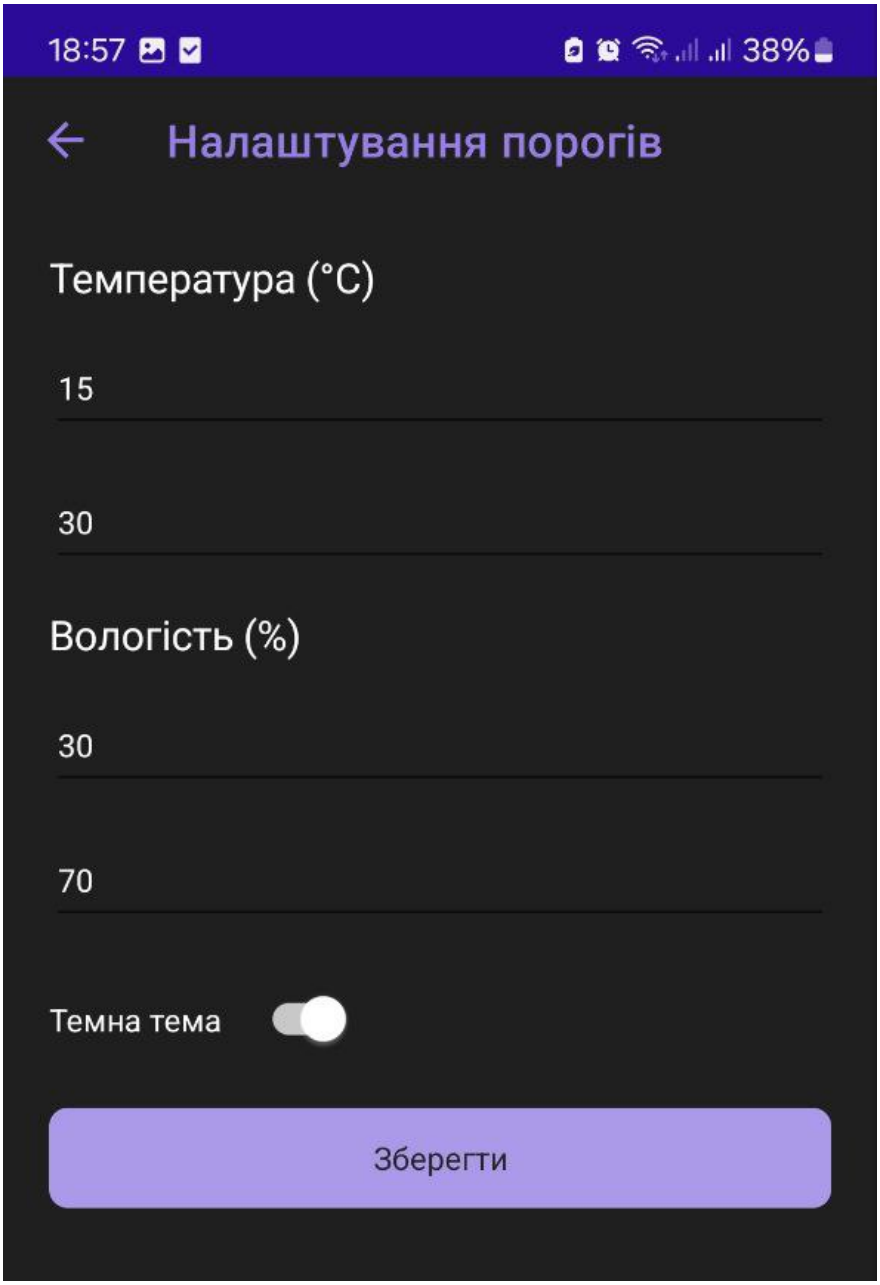

Рисунок 2.6 – Перемикач із світлої на темну тему

Кнопка «Експорт» – дозволяє зберігати поточні вимірювання у файл формату TXT (рис. 2.7). Після натискання відкривається діалогове вікно для

вибору місця збереження. Дані експортуються з точністю до часу та дати, що дає змогу проводити подальший аналіз у сторонніх редакторах наприклад Excel.

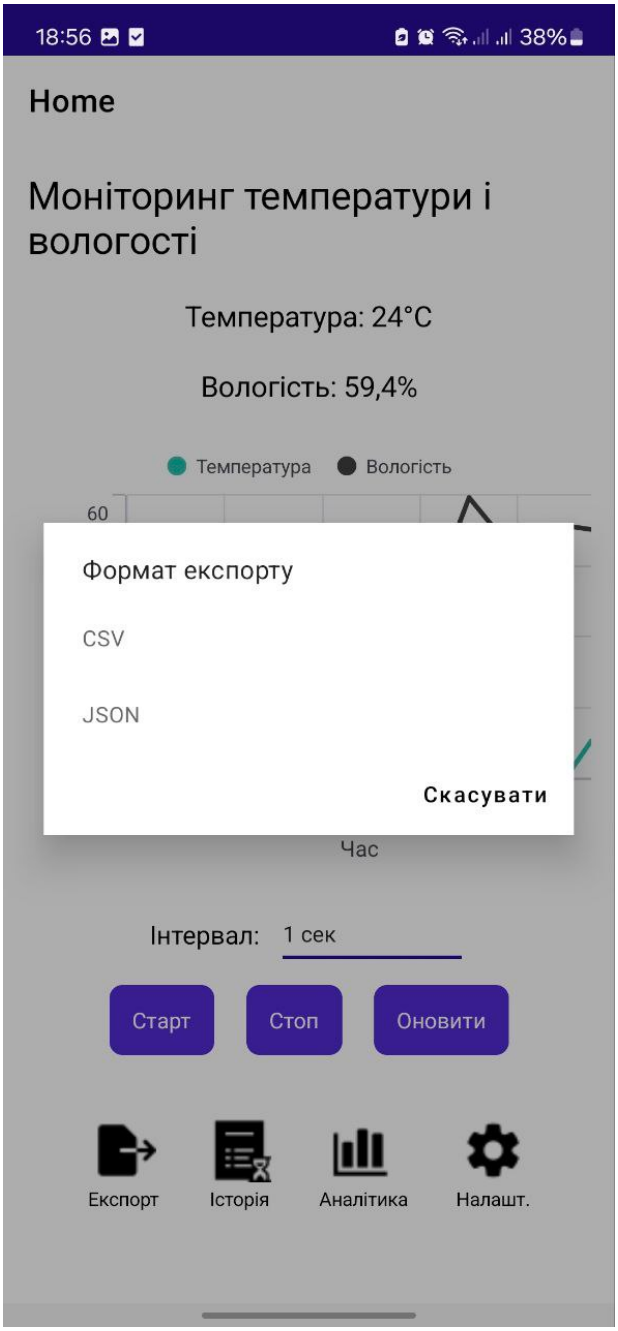

Рисунок 2.7 – Експорт даних

Кнопка «Історія» – відкриває окреме вікно з таблицею, в якій відображено збережені значення за попередні сесії (рис. 2.8). Користувач може відсортувати

або відфільтрувати записи за датою або діапазоном температур/вологості. Це полегшує виявлення довготривалих тенденцій у мікрокліматі.

Рисунок 2.8 – Історія спостережень

Кнопка «Аналітика» – показує зведену інформацію про середні значення температури і вологості (рис. 2.9). Присутня міні-діаграма, що ілюструє

динаміку змін. Розділ орієнтований на швидкий аналіз без необхідності детального перегляду історії.

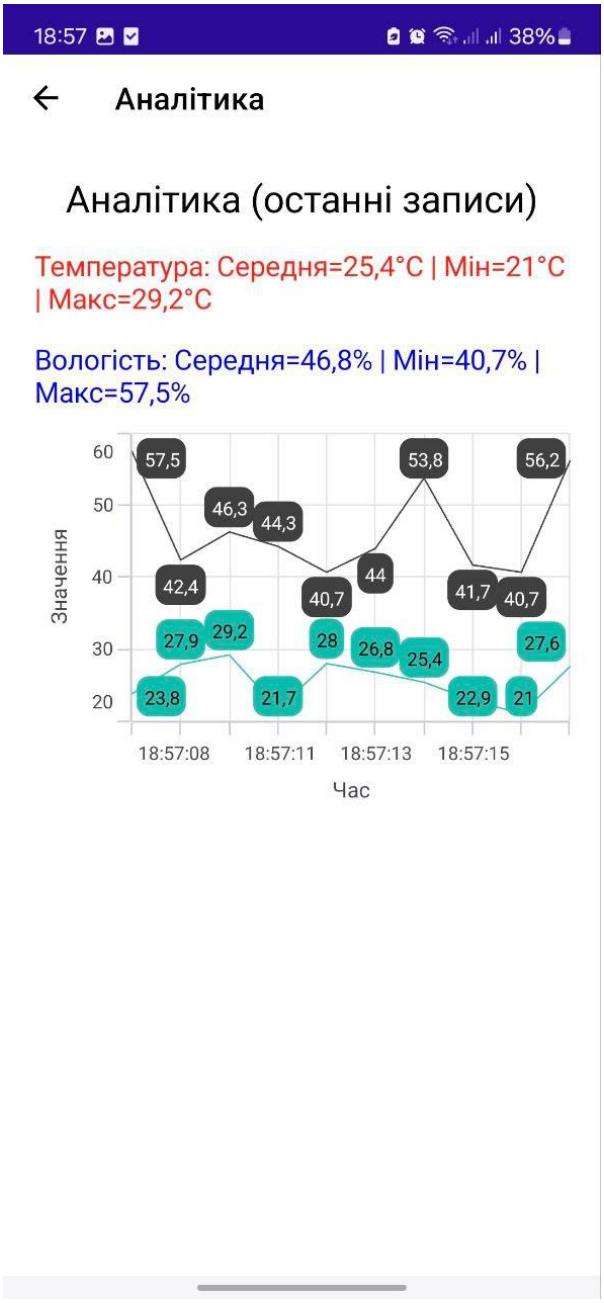

Рисунок 2.9 – Аналітика даних

Кнопка «Налаштування» відкриває меню (рис. 2.10), у якому користувач має можливість задати граничні порогові значення температури та вологості, що дозволяє адаптувати застосунок під індивідуальні умови середовища. Крім того,

у цьому меню можна перемкнути тему інтерфейсу (світла/темна), що покращує зручність використання в різних умовах освітлення.

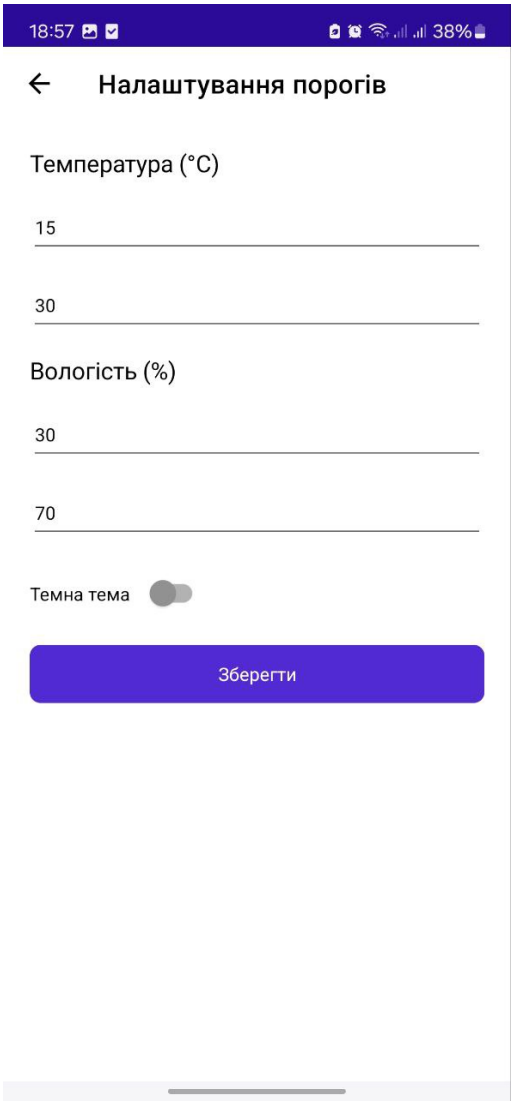

Рисунок 2.10 – Налаштування теми і порогів температури та вологості

Таким чином, кожен елемент інтерфейсу реалізовано відповідно до функціонального призначення, забезпечуючи зручність, ефективність та інтуїтивність взаємодії з користувачем. Програму було реалізовано мовою C# у середовищі Visual Studio 2022 із використанням кросплатформеної технології .NET MAUI. Сам застосунок розроблено для мобільних пристроїв із підтримкою Android і Windows.

2.3 Розробка моделі системи

Для забезпечення логічної послідовності виконання всіх етапів взаємодії користувача із мобільним застосунком було побудовано діаграму діяльності, яка відображає структуру поведінки системи у різних сценаріях роботи. Дана модель є основою для реалізації функціоналу та відображає ключові етапи обробки даних, перевірки коректності, виведення результатів і вибору подальших дій користувачем (рис. 2.11).

На початковому етапі після запуску програми відбувається ініціалізація Bluetooth-з’єднання. Якщо модуль не виявлено, система повідомляє про помилку. У випадку успішного з’єднання – програма переходить до етапу зчитування даних з датчиків Arduino. Далі відбувається перевірка отриманих даних на коректність. Якщо дані некоректні – виводиться відповідне повідомлення.

У разі успішного отримання та валідації показників температури і вологості, значення зберігаються в історію вимірювань, а також оновлюється графік на головному екрані. Додатково проводиться перевірка, чи виходять отримані значення за встановлені порогові межі. Якщо порушення фіксуються – система ініціює відображення відповідного сповіщення для користувача.

Після цього система переходить у режим очікування взаємодії з користувачем. Доступні такі варіанти подальших дій: перегляд історії вимірювань, експорт даних у файл формату .txt або перегляд аналітичного зведення. Усі ці дії є незалежними та можуть бути виконані у будь-якій послідовності.

Таким чином, побудована модель системи чітко визначає послідовність обробки даних, умови переходу між етапами та реагування на критичні ситуації. Це дозволяє забезпечити не лише стабільну роботу мобільного застосунку, а й своєчасне виявлення та інформування користувача про відхилення кліматичних параметрів, підтримувати цілісність даних, гарантувати безперервність

моніторингу й надійну візуалізацію інформації на всіх етапах функціонування системи.

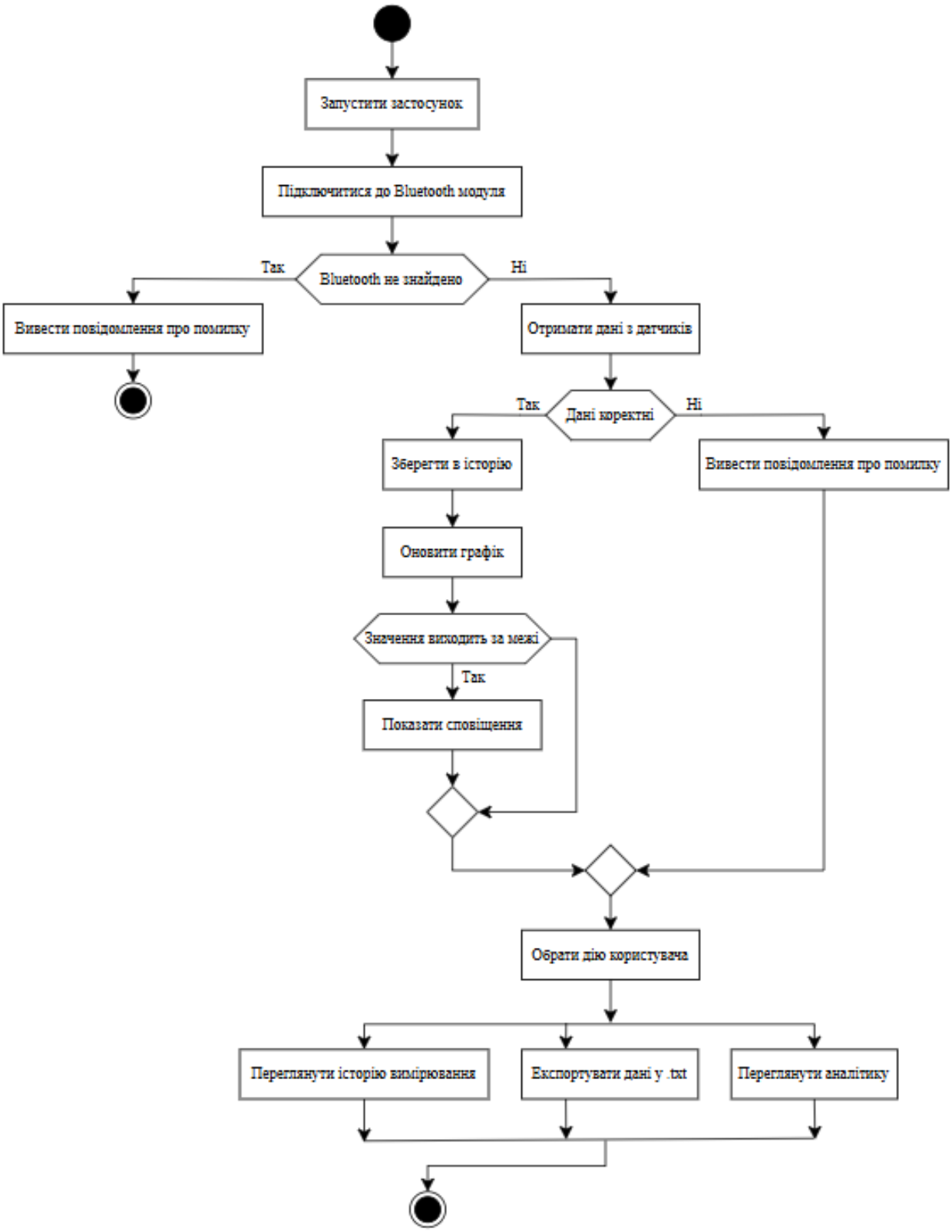

Рисунок 2.11 – Модель системи у вигляді діаграма діяльності дій застосунку

2.4 Розробка алгоритмів роботи системи

Алгоритм роботи мобільного застосунку (рис. 2.12) побудовано з урахуванням послідовної обробки подій, починаючи від ініціалізації інтерфейсу та завершуючи виконанням дій користувача. Основна мета алгоритму – забезпечення стабільного збору, аналізу та збереження даних із сенсора температури та вологості, підключеного через Bluetooth.

Після запуску застосунку автоматично ініціалізується інтерфейс користувача та запускається процедура сканування доступних Bluetooth-пристроїв. Якщо жоден пристрій не знайдено, користувач одразу інформується про помилку з'єднання, і подальше виконання припиняється. Це дозволяє уникнути помилкової обробки даних та зменшує навантаження на систему.

Якщо сенсор знайдено, застосунок переходить до етапу підключення до пристрою та запиту даних. Після отримання відповідей виконується перевірка коректності даних. У разі виявлення помилки користувач отримує відповідне сповіщення, і робота алгоритму припиняється. Якщо ж дані є коректними – значення температури і вологості зберігаються в локальній історії.

Зібрані значення миттєво виводяться на графік, що дає змогу візуалізувати зміни в реальному часі. Система додатково виконує перевірку на вихід значень за задані порогові межі, які зберігаються у локальних налаштуваннях. Якщо такі порушення виявлено – застосунок виводить відповідне сповіщення для попередження користувача про критичний стан середовища.

Після оновлення інтерфейсу та графіка система переходить у режим очікування взаємодії з користувачем. Відповідно до обраної дії, користувач може переглянути історію вимірювань, експортувати дані у текстовий файл або переглянути узагальнену статистичну аналітику. Усі ці дії реалізуються без повторного звернення до сенсора, що дозволяє зменшити кількість Bluetooth-запитів і оптимізувати використання ресурсів пристрою.

Таким чином, розроблений алгоритм забезпечує повний цикл збору, обробки, збереження та взаємодії з даними, при цьому залишаючись гнучким до

дій користувача й адаптованим до умов відсутності з'єднання чи помилок передавання. Це робить систему стабільною, зручною у використанні та придатною до розширення у майбутньому.

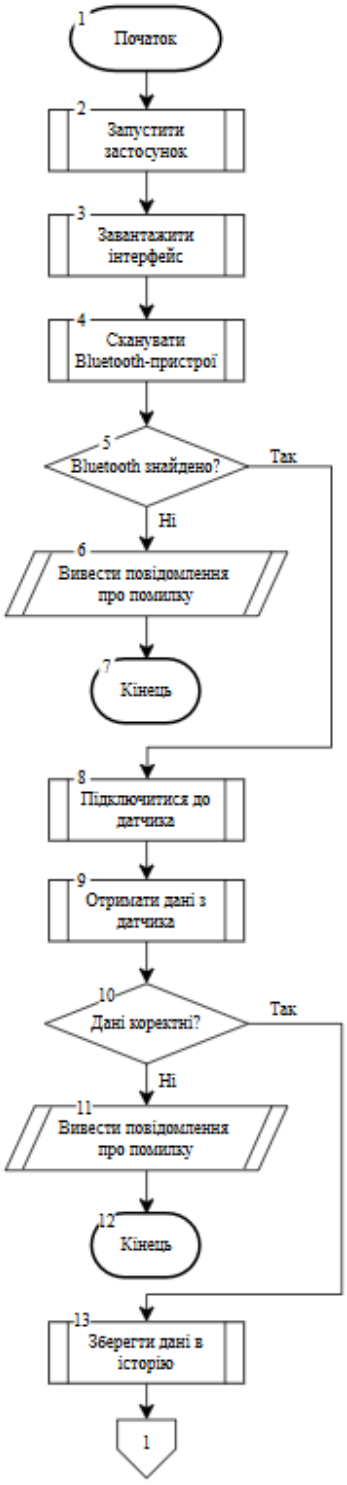

Рисунок 2.12 – Алгоритм роботи програмного застосунку

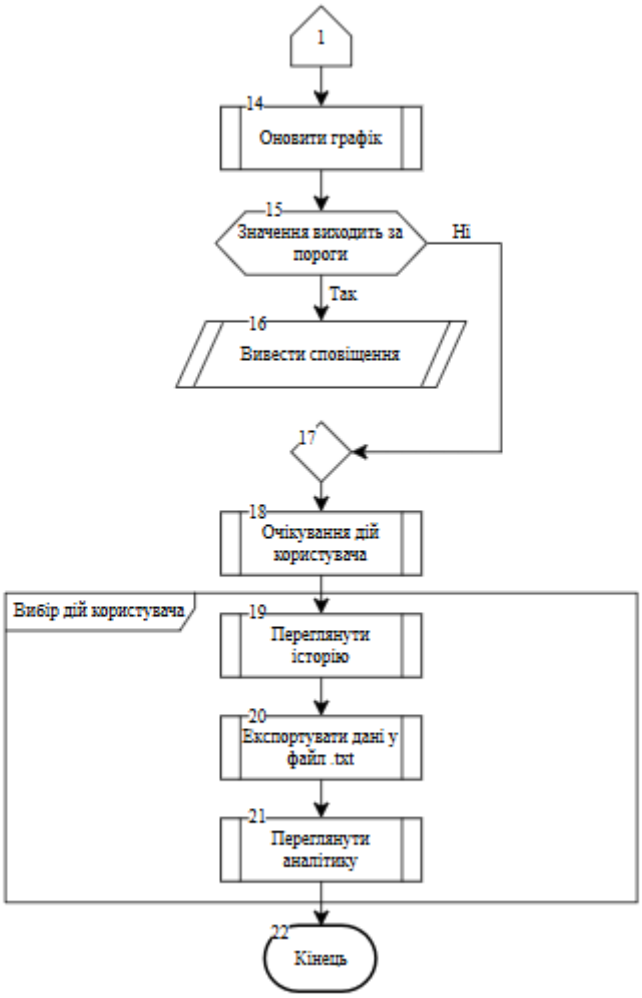

Рисунок 2.13 – Продовження алгоритму роботи програмного застосунку

Для забезпечення коректної роботи мобільного застосунку, який реалізує моніторинг температури та вологості приміщення, було розроблено алгоритм, що відображає логіку роботи головного модуля (рис. 2.14).

Блок-схема, представлена на рисунку, охоплює два основних етапи роботи застосунку. Перший етап включає запуск і ініціалізацію Bluetooth-з’єднання з датчиком. Після запуску програми здійснюється завантаження графічного інтерфейсу користувача та автоматичний пошук доступних Bluetooth-пристроїв. Якщо датчик виявлено, виконується підключення та зчитування даних. У разі успішного зчитування перевіряється коректність отриманих значень, після чого дані зберігаються у файл історії.

Другий етап відображає обробку зібраних даних: оновлення графіка, перевірка на перевищення порогових значень температури та вологості. У разі перевищення меж виводиться відповідне сповіщення, а інформація записується у файл сповіщень. Далі програма очікує дії користувача, серед яких можливі: перегляд історії, експорт даних у форматі .txt та перегляд аналітичних результатів.

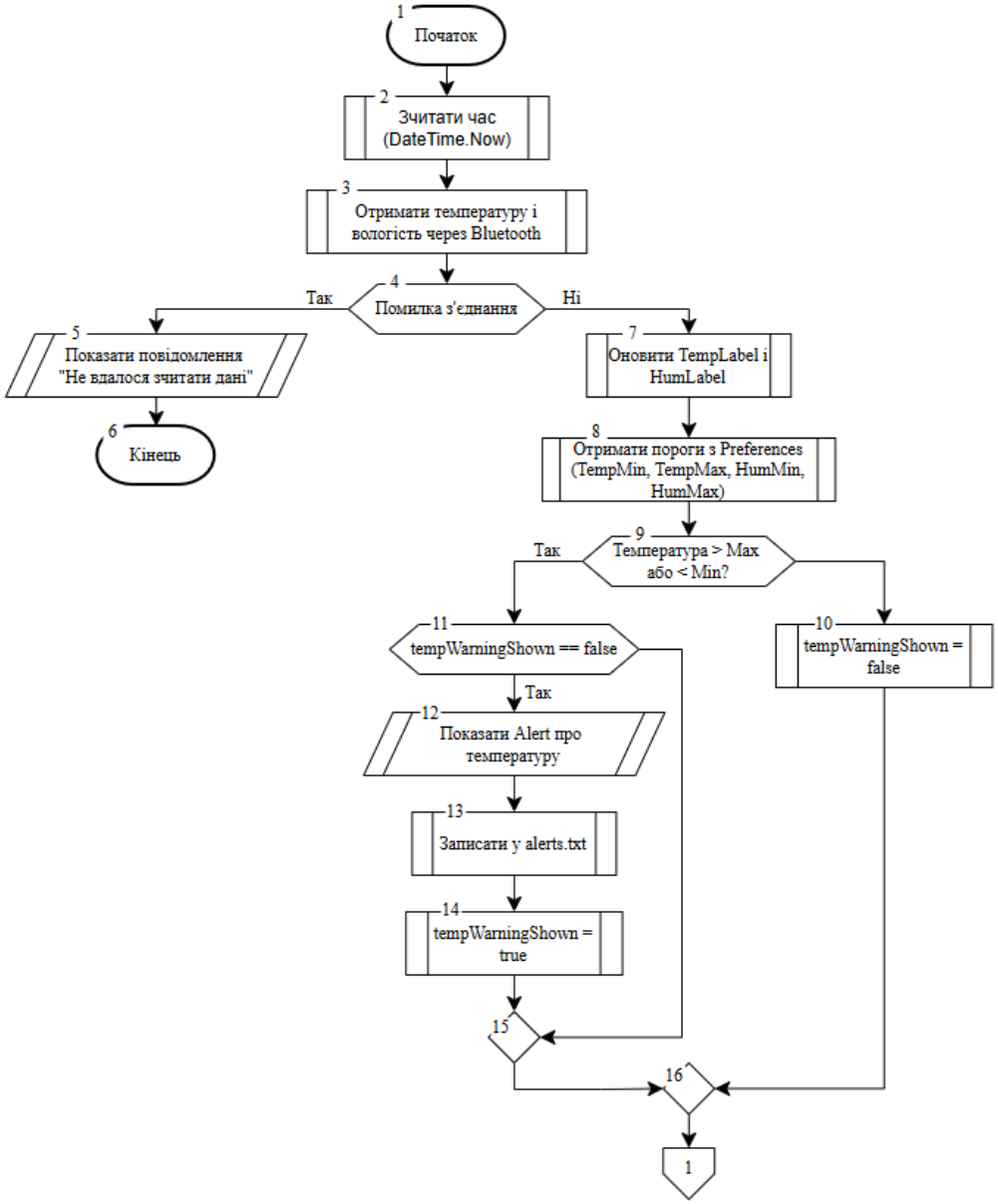

Рисунок 2.14 – Алгоритм функціонування системи моніторингу температури і вологості

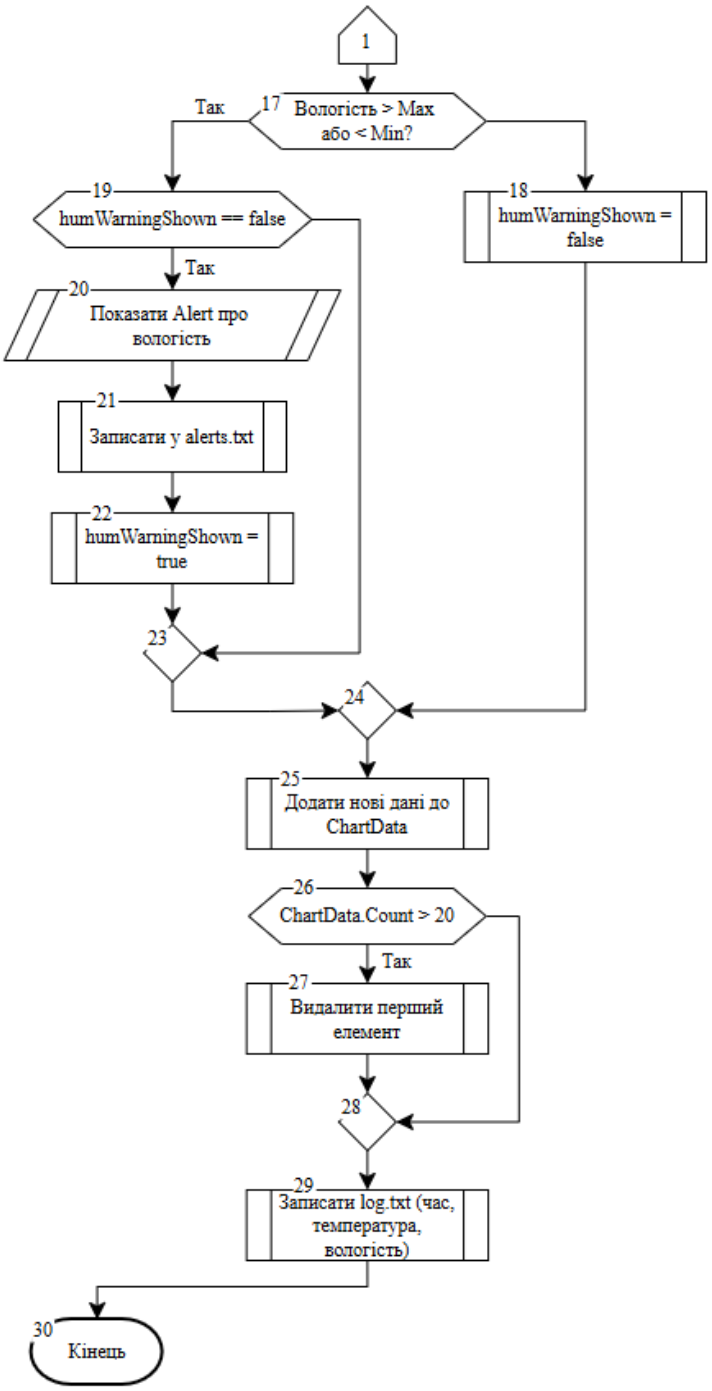

Рисунок 2.15 – Продовження алгоритму функціонування системи моніторингу температури і вологості

Таким чином, алгоритм забезпечує безперервну обробку отриманих від датчика даних та їх відображення у реальному часі, а також інтерактивну

взаємодію з користувачем, що дозволяє ефективно контролювати параметри мікроклімату у приміщенні.

2.5 Висновки

У другому розділі було виконано повний цикл розробки структури та алгоритмів функціонування мобільного застосунку для моніторингу температури та вологості приміщення в реальному часі. Проведено аналіз вхідних даних системи, що дозволив визначити технічні вимоги до застосунку, вибрати відповідне середовище розробки – .NET MAUI, а також забезпечити підтримку роботи з Bluetooth-пристроями за допомогою бібліотеки Plugin.BLE. Розроблений користувацький інтерфейс реалізовано відповідно до принципів зручності, адаптивності та наочності. Завдяки інтеграції з бібліотекою Syncfusion.Maui.Charts було досягнуто ефективної візуалізації даних у вигляді динамічних графіків.

Алгоритм роботи системи побудований таким чином, щоб забезпечити безперервне зчитування даних із сенсора, перевірку їх коректності, аналіз на відповідність порогам, відображення результатів та інтерактивну взаємодію з користувачем. Побудовані блок-схеми детально описують основні етапи роботи застосунку – від ініціалізації Bluetooth-з’єднання до збереження історії вимірювань, виводу сповіщень та аналітичного узагальнення. Таким чином, розроблена структура повністю відповідає поставленому функціоналу і забезпечує стабільну роботу мобільного застосунку в умовах реального використання.

3. РОЗРОБКА ФУНКЦІОНАЛЬНИХ МОДУЛІВ СИСТЕМИ МОНІТОРИНГУ ТЕМПЕРАТУРИ І ВОЛОГОСТІ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

3.1 Обґрунтування вибору середовища та інструментів для розробки програмного забезпечення

У процесі розробки мобільного застосунку для моніторингу температури та вологості приміщення важливим етапом стало обґрунтування вибору середовища програмування, технологічної платформи та додаткових інструментів. Основними критеріями для вибору засобів розробки стали такі вимоги, як підтримка кросплатформенності, висока продуктивність роботи інтерфейсу користувача, можливість роботи з Bluetooth-пристроями, підтримка сучасних засобів візуалізації даних, а також зручність у налагодженні та розширенні функціоналу.

В якості основної технологічної платформи було обрано .NET MAUI (Multi-platform App UI) – сучасний кросплатформений фреймворк від компанії Microsoft. .NET MAUI дозволяє створювати застосунки одразу для кількох операційних систем, зокрема Android, iOS та Windows, з використанням єдиної кодової бази на мові програмування C#. Вибір саме .NET MAUI обумовлений його гнучкою архітектурою, високою швидкістю розробки, наявністю великої кількості вбудованих засобів для роботи з графічним інтерфейсом, а також підтримкою Bluetooth API [13] через сторонні бібліотеки.

Для реалізації функціоналу підключення та обміну даними з Bluetooth-пристроями було використано бібліотеку Plugin.BLE [14]. Ця бібліотека надає простий у використанні API для виявлення пристроїв, встановлення з'єднання та обміну даними з модулями Bluetooth Low Energy (BLE), такими як сенсорний модуль на базі Arduino з підключеним DHT22.

Для відображення графічних даних про температуру та вологість у реальному часі було обрано бібліотеку Syncfusion.Maui.Charts, яка забезпечує

побудову гнучких і динамічних графіків з можливістю налаштування осей, кольорів, підписів і легенди. Використання цієї бібліотеки дозволяє реалізувати наочну та адаптивну візуалізацію змін показників мікроклімату, що суттєво підвищує зручність аналізу даних користувачем.

Середовищем розробки обрано Visual Studio 2022 [15] з установленим пакетом MAUI Workload. Visual Studio забезпечує повний цикл розробки, включаючи написання, тестування та налагодження коду. Воно також дозволяє здійснювати безпосереднє розгортання застосунку на фізичних пристроях Android та емуляторах, що істотно пришвидшує тестування реальної роботи Bluetooth-з'єднань та адаптивності інтерфейсу.

Вибір цих інструментів обумовлений їх стабільною роботою, відповідністю сучасним вимогам до мобільних застосунків, активною підтримкою спільноти розробників, широкою документацією та можливістю розширення функціоналу в майбутньому. Обрана сукупність технологій і середовищ забезпечила реалізацію всіх необхідних функцій застосунку [16]: зчитування реальних даних з сенсора, обробку і візуалізацію показників, ведення журналу спостережень, сповіщення про перевищення порогів та інтерактивну взаємодію з користувачем.

Таким чином, застосування платформи .NET MAUI у поєднанні з бібліотеками Plugin.BLE та Syncfusion.Maui.Charts [17], а також використання Visual Studio 2022 як основного середовища розробки повністю відповідає поставленим вимогам до функціоналу, надійності та ефективності мобільного застосунку.

3.2 Розробка модуля зчитування даних із сенсора через Bluetooth

Модуль зчитування даних із сенсора є одним із ключових компонентів мобільного застосунку, оскільки саме він забезпечує безперервний обмін даними між фізичним пристроєм Arduino (з підключеним датчиком DHT22) та мобільним застосунком через Bluetooth-з'єднання [18].

На цьому етапі застосунок виконує ініціалізацію основних механізмів роботи. Створюється таймер, який періодично викликатиме зчитування даних із сенсора, забезпечуючи безперервний моніторинг температури і вологості в реальному часі. Також відбувається підготовка до роботи з Bluetooth: створюється об'єкт адаптера для подальшого сканування пристроїв. Ця частина коду визначає фундаментальну частоту оновлення даних і формує основу всього процесу збору інформації. На рисунку 3.1 зображено блок-схему цього процесу.

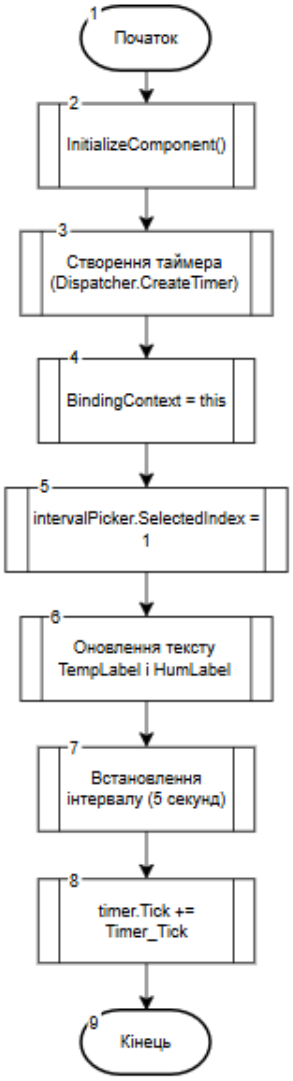

Рисунок 3.1 – Блок-схема ініціалізації таймера та підготовки до зчитування даних

На рисунку 3.2 зображено блок-схему виявлення пристрою ESP32 серед доступних Bluetooth-пристроїв. Після запуску процесу сканування система аналізує список знайдених пристроїв і виконує пошук за іменем, яке повинно містити рядок «ESP32». Якщо відповідний пристрій не знайдено, генерується виключення з повідомленням про помилку підключення. У разі успішного виявлення пристрою відбувається перехід до наступних етапів роботи з ним. Цей процес є критично важливим для подальшого зчитування даних із сенсора, оскільки визначає наявність каналу зв’язку між застосунком і пристроєм.

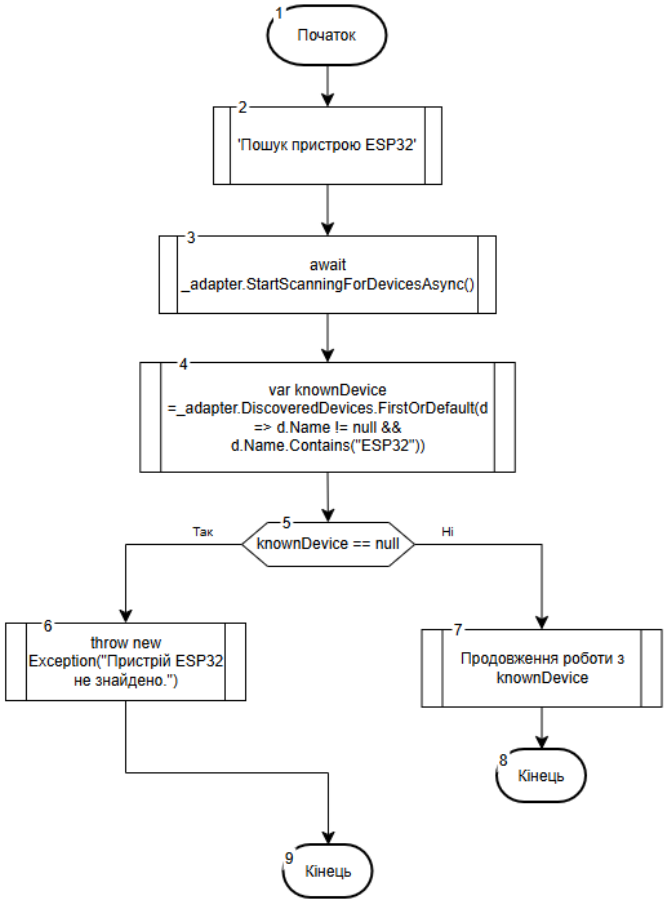

Рисунок 3.2 – Блок- схема підключення до Bluetooth-пристрою

На рисунку 3.3 зображено блок-схему, яка відповідає за фактичне отримання екологічних даних із сенсора. Дані надходять у вигляді масиву байтів,

який обробляється для відновлення значень температури і вологості у зручному для людини форматі (double). Після успішного зчитування значення передаються для оновлення інтерфейсу та подальшої обробки. Правильна обробка зчитаних даних є важливою для забезпечення актуальності графіка і коректного виведення порівняльних результатів у подальшому.

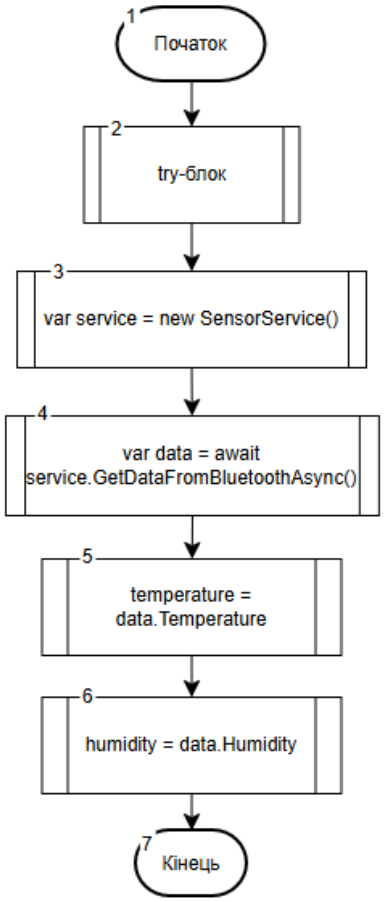

Рисунок 3.3 – Блок-схема зчитування даних температури і вологості

На рисунку 3.4 зображено блок-схему, де нові значення температури і вологості виводяться на екран у вигляді текстових міток для швидкого ознайомлення користувача з поточним станом мікроклімату. Паралельно відбувається додавання нових даних до графіка, що дозволяє в реальному часі спостерігати зміну параметрів. Для уникнення перенавантаження інтерфейсу

реалізовано обмеження на кількість відображуваних записів (наприклад, не більше 20 останніх значень). Це забезпечує зручність і високу швидкість оновлення візуального представлення.

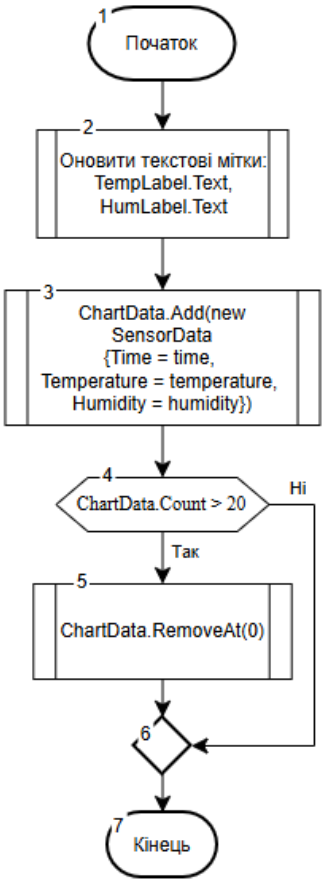

Рисунок 3.4 – Блок-схема оновлення інтерфейсу користувача та побудова графіка

На рисунку 3.5 зображено блок-схему, яка реалізує перевірку актуальних показників на відповідність встановленим порогам, які задаються користувачем через налаштування. Якщо температура або вологість виходить за межі допустимих значень, система автоматично формує повідомлення-попередження для користувача через інтерфейс. Окрім виводу сповіщення, інформація про порушення записується у файл сповіщень для подальшого аналізу. Такий механізм дозволяє користувачу оперативно реагувати на небезпечні зміни в

навколишньому середовищі. ось мої пункти і я тобі скинув код, напиши уривки коду, які треба взяти для цих пунктів

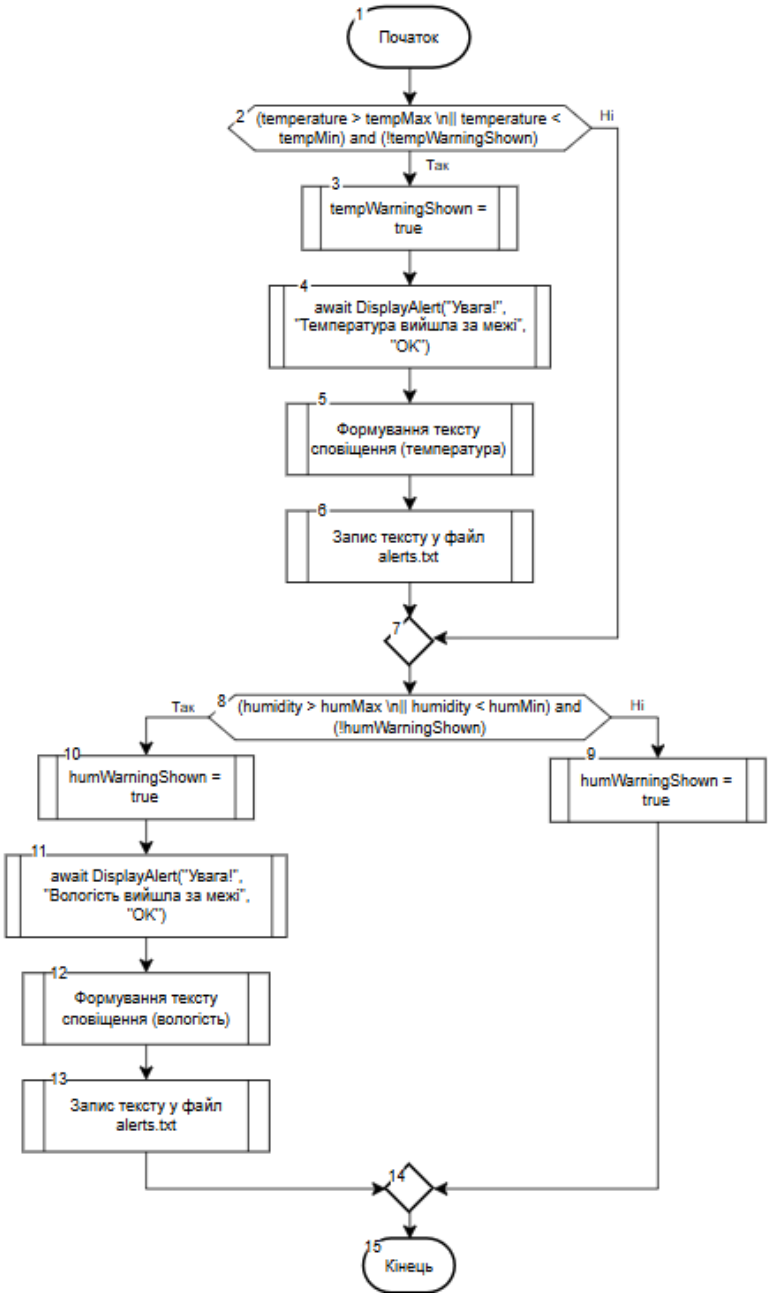

Рисунок 3.5 – Блок-схема перевірки порогових значень і генерація сповіщень

Отже, даний модуль забезпечує високу надійність обробки даних і дозволяє застосунку працювати в реальному часі. Завдяки цьому користувач отримує актуальні показники температури і вологості без помітних затримок.

3.3 Розробка модуля візуалізації та керування пороговими сповіщеннями

Розробка функціональних можливостей мобільного застосунку для моніторингу температури та вологості є ключовим етапом реалізації розробки. Основна задача цього етапу – забезпечити безперервний збір даних із сенсора, їх коректне відображення, обробку, аналіз та можливість взаємодії користувача з результатами вимірювань.

Розроблений застосунок включає основні функціональні компоненти: запуск і зупинку збору даних, очищення історії вимірювань, експорт даних для подальшого аналізу, а також графічне відображення змін температури та вологості у реальному часі.

На рисунку 3.6 зображено блок-схему, яка реалізує функцію запуску циклічного збору даних із сенсора. При натисканні на кнопку «Старт» здійснюється перевірка, чи не запущений вже таймер збору даних. Якщо таймер неактивний, він запускається, і застосунок починає періодично виконувати зчитування показників температури та вологості через функцію Timer_Tick [19]. Таким чином забезпечується безперервний моніторинг стану навколишнього середовища.

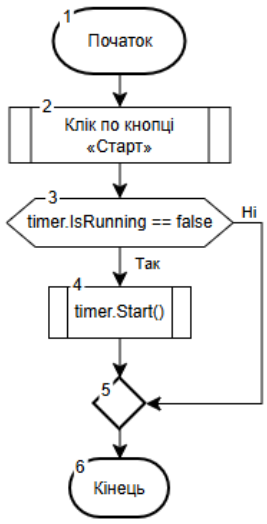

Рисунок 3.6 – Блок-схема обробки натискання кнопки «Старт»

На рисунку 3.7 зображено блок-схему, яка зупиняє цикл збору даних. Якщо таймер активний, при натисканні кнопки «Стоп» він зупиняється, що припиняє зчитування нових значень температури і вологості. Це дозволяє користувачу контролювати, коли саме потрібно припинити моніторинг середовища, наприклад, для збереження ресурсів акумулятора пристрою або для фіксації конкретного моменту часу.

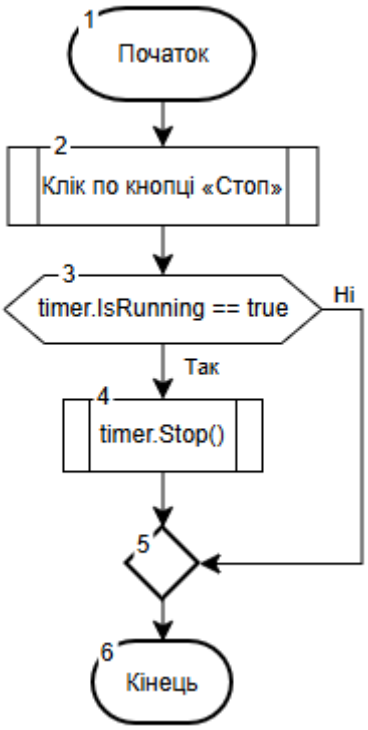

Рисунок 3.7 – Блок-схема обробки натискання кнопки «Стоп»

Блок-схема зображена на рисунку 3.8 забезпечує збереження отриманих даних у текстовий файл. Якщо дані є у колекції ChartData, формується текстовий документ із датами, часом вимірювання та відповідними значеннями температури і вологості. Файл може бути переданий за допомогою стандартних засобів операційної системи користувача через механізм поділитися (наприклад,

на Google Drive, Email). Це полегшує архівування результатів моніторингу або подальший їх аналіз.

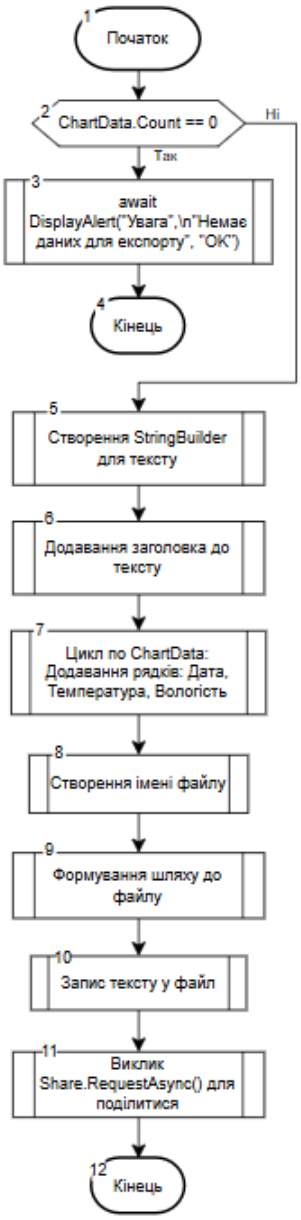

Рисунок 3.8 – Блок-схема експорту даних у файл TXT

Динамічне оновлення графіка [20] температури і вологості (рис. 3.9) реалізовано за допомогою бібліотеки Syncfusion.Maui.Charts. Кожне нове значення автоматично додається у відповідну серію графіка, що дозволяє користувачу спостерігати зміни параметрів середовища у реальному часі. Графік

має підписи осей, легенду та можливість перегляду точних значень через підказки.

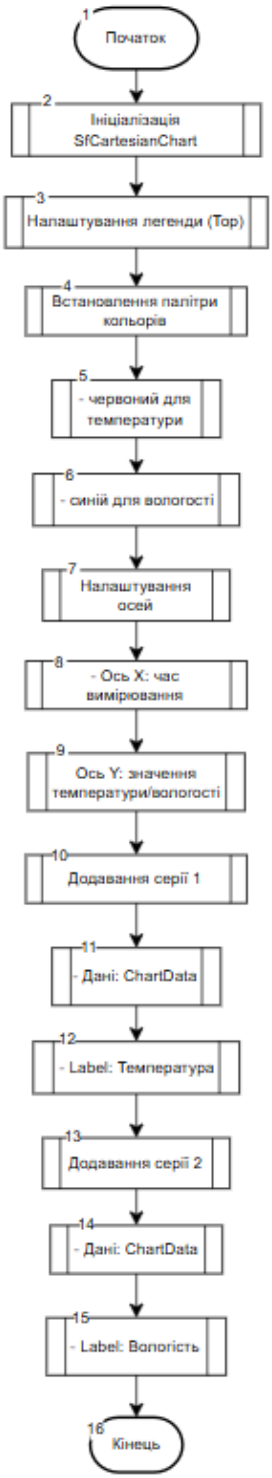

Рисунок 3.9 – Блок-схема побудови графіка на основі даних

Після натискання кнопки «Оновити» (рис. 3.10) відбувається повне очищення графічних даних (ChartData.Clear()), а також скидання текстових індикаторів температури та вологості до початкового стану. Це дозволяє користувачу швидко перезапустити збір даних, починаючи нову сесію вимірювань без змішаних старих і нових значень. Такий механізм особливо корисний для точного аналізу окремих часових проміжків або змін середовища.

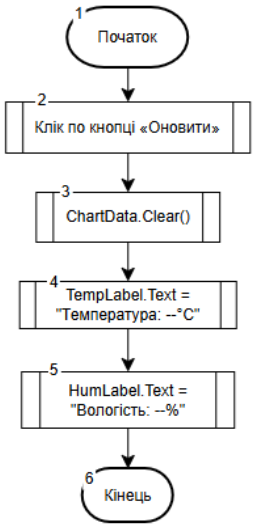

Рисунок 3.10 – Блок-схема кнопки оновлення та очищення даних

3.4 Висновки

У цьому розділі було обґрунтовано вибір середовища розробки .NET MAUI та реалізовано основні функціональні модулі мобільного застосунку для моніторингу температури і вологості. Реалізовано підключення до сенсора ESP32 через Bluetooth, зчитування кліматичних даних, оновлення інтерфейсу, виведення графіка[21] та порогові сповіщення.

Усі ключові етапи роботи описано у вигляді блок-схем, що дало змогу чітко структуризувати процеси. Застосунок забезпечує стабільний збір і візуалізацію даних у реальному часі, підтримує теми оформлення, експорт у файл і реагує на відхилення параметрів. Отримане рішення придатне для подальшого практичного застосування.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Основні етапи тестування мобільного застосунку

Тестуванням розробленого мобільного застосунку є всебічна перевірка правильності функціонування усіх його основних модулів, а також оцінка відповідності реалізованих функцій технічним і функціональним вимогам. Тестування орієнтувалося на забезпечення стабільності Bluetooth-з'єднання, коректності зчитування показників температури і вологості, безперервної візуалізації даних у реальному часі та правильності спрацьовування механізмів сповіщень про порушення заданих порогових меж.

На етапі тестування використовувалася модель передачі даних через Bluetooth, яка дозволила відтворити процес зчитування реальних екологічних показників. Це надало можливість комплексно перевірити механізм обробки отриманої інформації: правильність заповнення графічного відображення температури і вологості, оновлення текстових індикаторів на основному екрані застосунку, фіксацію перевищення встановлених порогових значень та формування відповідних сповіщень для користувача.

Окремо тестувалася стабільність роботи інтерфейсу при взаємодії з користувачем: запуск і зупинка збору даних через відповідні кнопки керування, очищення історії спостережень, експорт збережених даних у текстові файли формату TXT. Перевірка охоплювала також адаптивність графічних компонентів при зміні розміру екрана і тестування застосунку на фізичному пристрої для оцінки реальної чутливості елементів керування та коректності відображення в різних орієнтаціях.

Отже, тестування було спрямоване не лише на виявлення можливих помилок, а й на підтвердження повноцінної готовності програмного продукту до реального використання. Результати тестування дозволили зробити висновок про працездатність основних функцій застосунку, його стабільність, зручність взаємодії з користувачем та відповідність початково поставленим завданням.

4.2 Тестування системи

Після встановлення та запуску мобільного застосунку перевіряється коректність відображення стартового екрана (рис. 4.1). Інтерфейс має містити текстові індикатори температури та вологості, панель керування кнопками «Старт», «Стоп», «Оновити», «Експорт», «Історія», «Аналітика» і кнопку доступу до налаштувань. На цьому етапі важливо переконатися, що усі елементи відображаються правильно та інтерфейс адаптивно підлаштовується під екран мобільного пристрою.

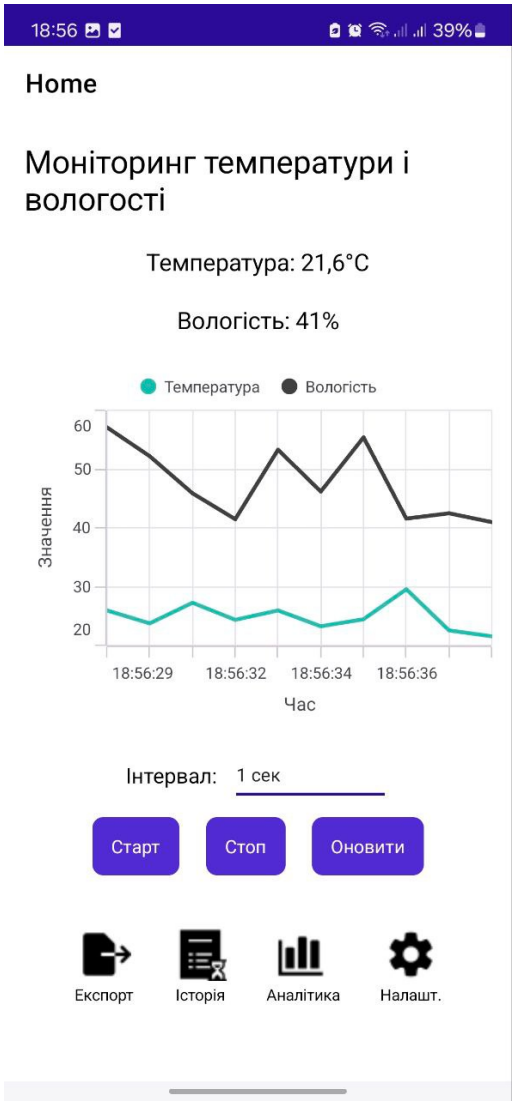

Рисунок 4.1 – Перевірка запуску застосунку та ініціалізації інтерфейсу

На другому етапі тестується функціонал запуску збору даних. Після натискання кнопки «Старт» має активуватись таймер збору показників температури та вологості (рис. 4.2). Текстові індикатори повинні почати оновлюватися через вибраний інтервал часу, а графік – поступово наповнюватися новими точками даних.

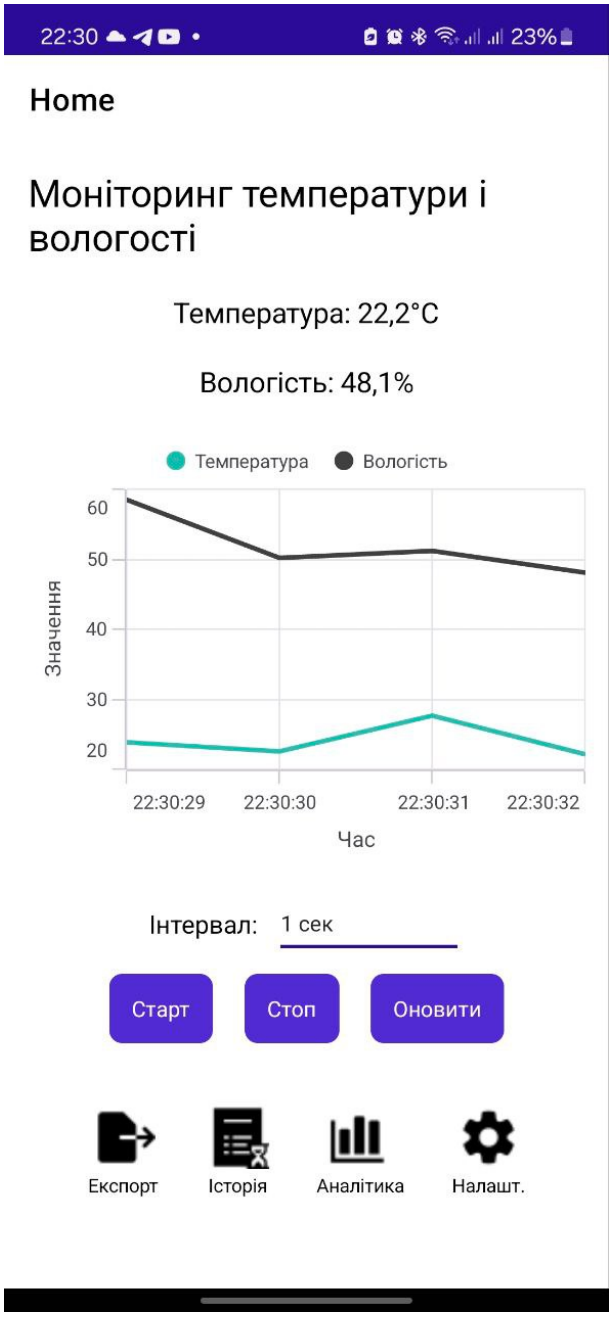

Рисунок 4.2 – Тестування роботи кнопки «Старт»

На цьому етапі перевіряється динамічне оновлення графіка. Зібрані дані температури і вологості мають додаватися до відповідних ліній графіка у реальному часі (рис. 4.3). Також важливо перевірити обмеження на кількість точок для забезпечення стабільної роботи застосунку.

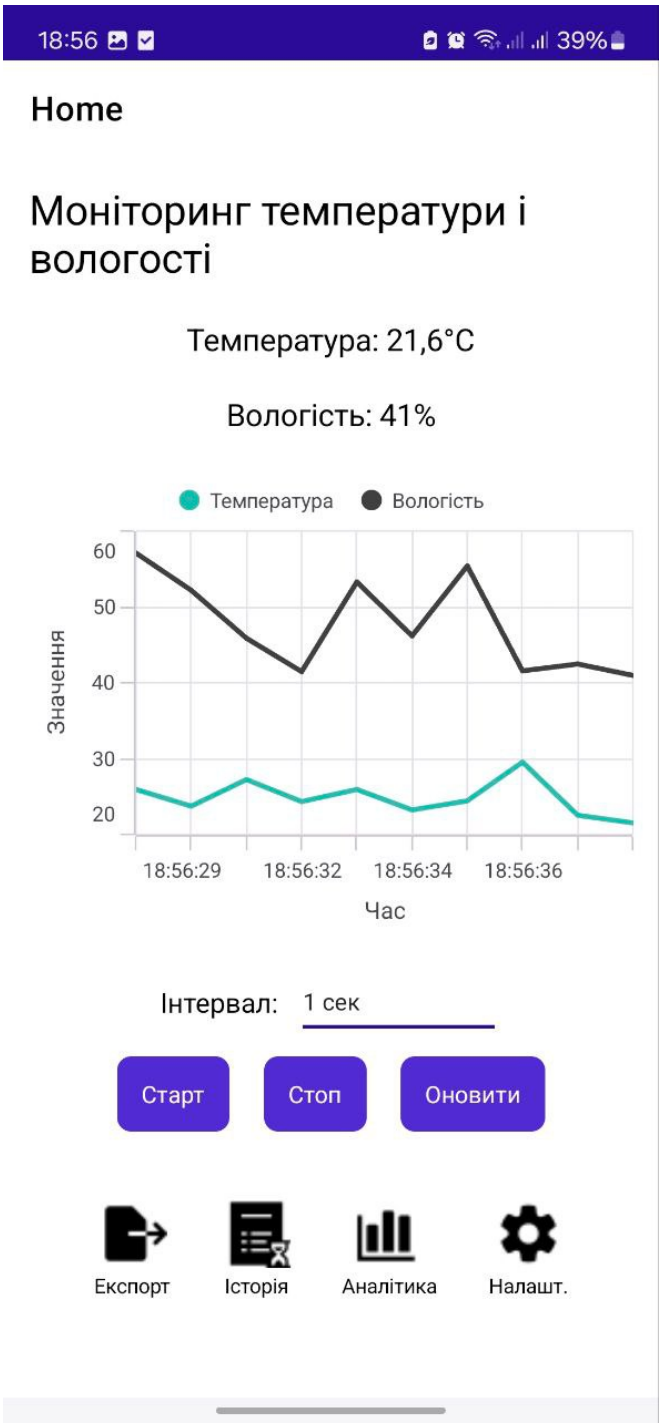

Рисунок 4.3 – Тестування роботи побудови графіка

Далі перевіряється система контролю порогових значень (рис. 4.4). При виході фактичної температури або вологості за встановлені межі має з'являтися відповідне сповіщення на екрані з попередженням користувача. Також перевіряється запис інформації про перевищення порогів у файл сповіщень.

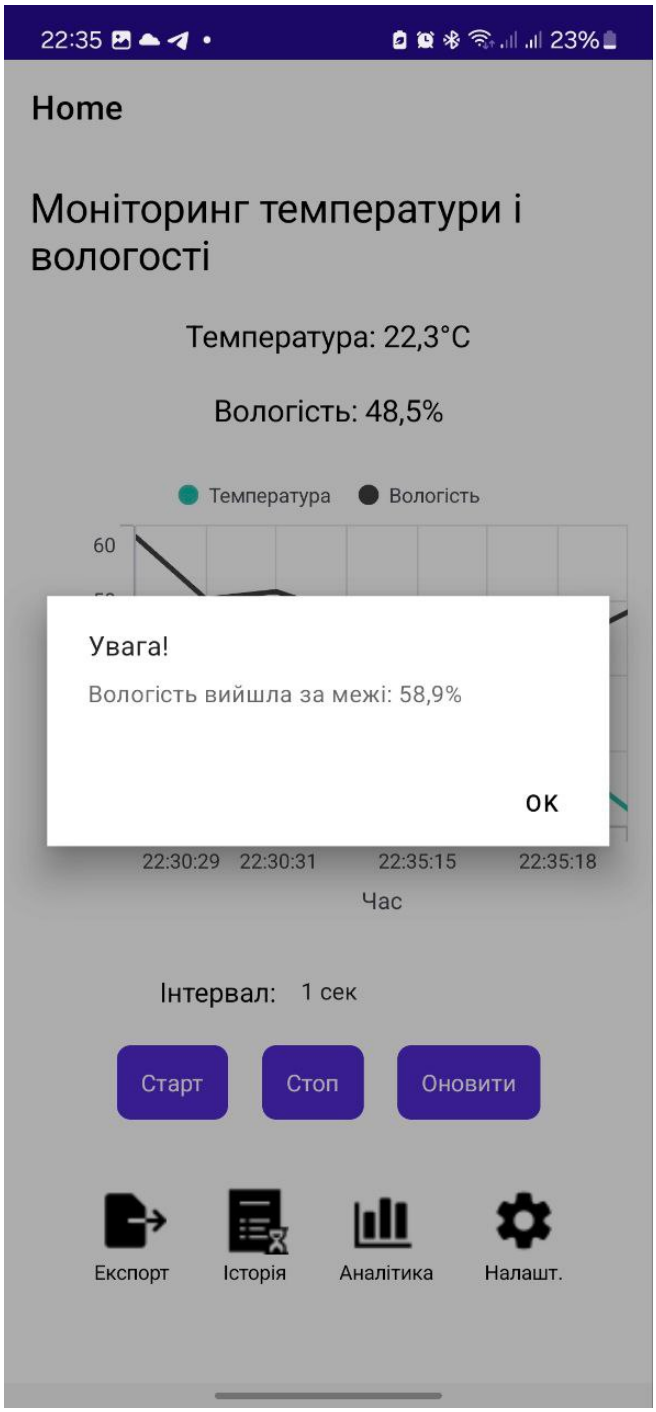

Рисунок 4.4 – Тестування роботи порогових сповіщень

Після натискання кнопки «Стоп» має зупинитися таймер збору даних, оновлення текстових індикаторів та побудова нових точок графіка має припинитися (рис. 4.5). Це дозволяє завершити сесію моніторингу без збоїв.

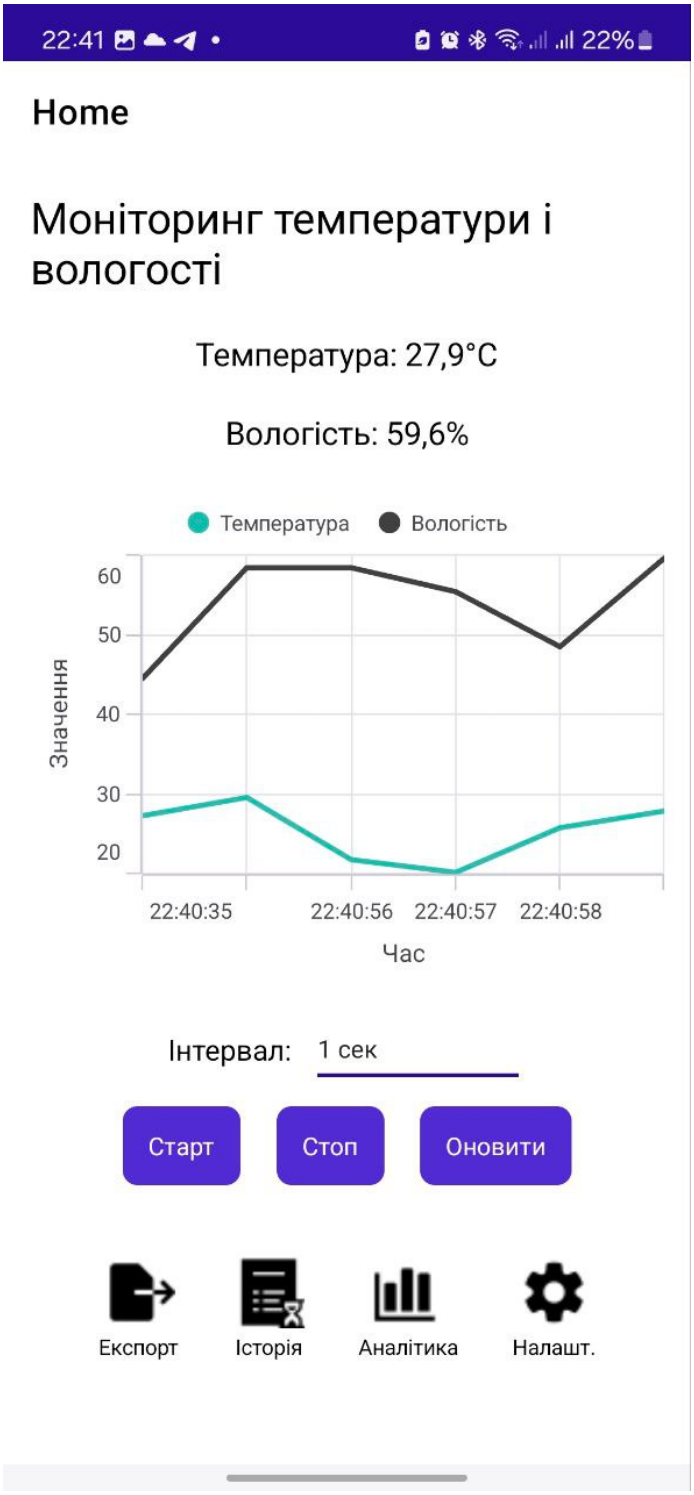

Рисунок 4.5 – Тестування роботи кнопки «Стоп»

Окремо тестується можливість перегляду історії спостережень. При натисканні на кнопку «Історія» має відкриватися окреме вікно або діалогове повідомлення (рис. 4.6) із відображенням попередніх записів вимірювань. Також перевіряється можливість очищення історії при бажанні користувача.

Рисунок 4.6 – Перевірка функціоналу перегляду історії

На цьому етапі перевіряється можливість зміни порогових значень температури і вологості через меню налаштувань (рис. 4.7). Після зміни порогів застосунок має зберігати нові значення і використовувати їх для подальших перевірок при зборі даних.

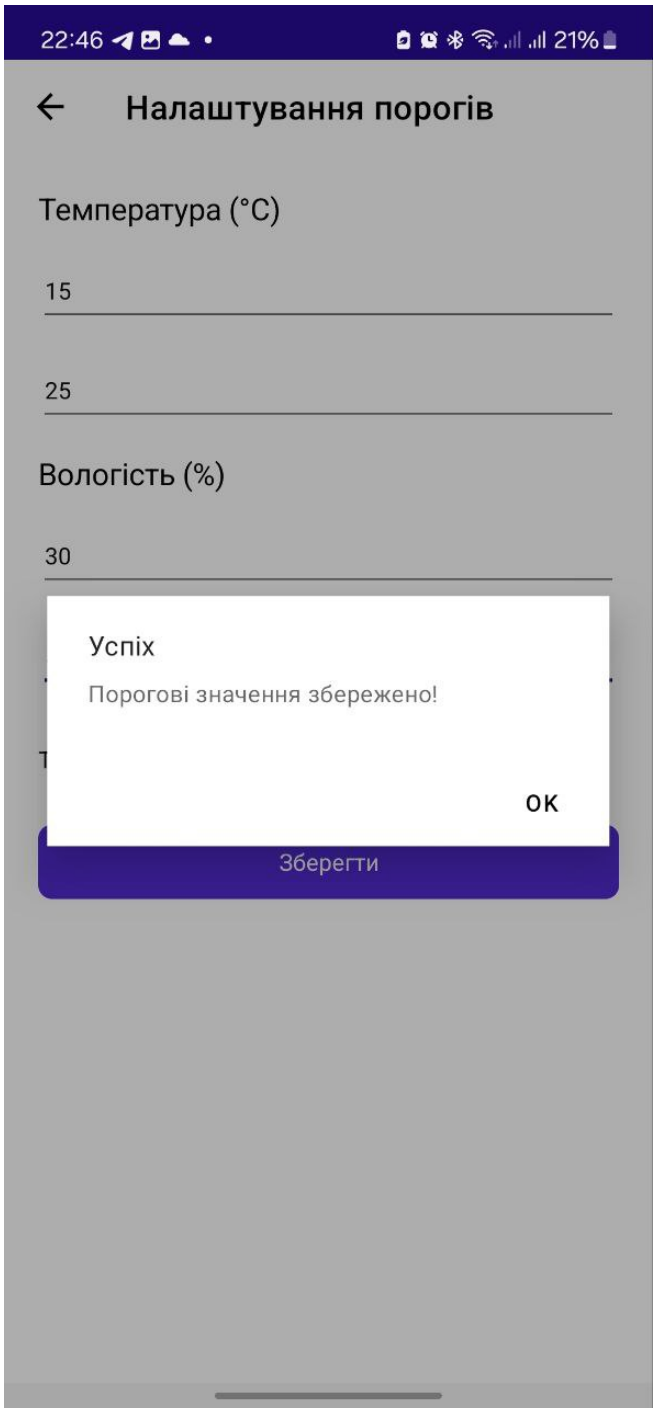

Рисунок 4.7 – Перевірка роботи налаштувань порогових значень

Отже, за результатами проведеного тестування було підтверджено працездатність усіх основних модулів мобільного застосунку. Застосунок стабільно здійснює зчитування даних, їх обробку, побудову графіка у реальному часі та генерацію сповіщень при перевищенні порогових значень. Також успішно працює функціонал експорту та перегляду історії. Інтерфейс є адаптивним і забезпечує зручну взаємодію користувача із системою, що свідчить про відповідність розробленого програмного забезпечення поставленим вимогам.

4.3 Розробка інструкції користувача

Для забезпечення ефективного використання мобільного застосунку для моніторингу температури і вологості приміщення було розроблено інструкцію користувача, що описує порядок взаємодії з основними функціями системи, а також визначає мінімальні системні вимоги до пристрою.

Застосунок розрахований на використання на сучасних мобільних пристроях із підтримкою Bluetooth-з’єднання. Мінімальні технічні характеристики, необхідні для стабільної роботи застосунку, наведено у таблиці 4.1.

Таблиця 4.1 – Системні вимоги до мобільного пристрою

Параметр	Мінімальні вимоги
Операційна система	Android 8.0 або новіша
Процесор	4-ядерний, частота не менше 1.4 ГГц
Оперативна пам’ять (RAM)	Від 2 ГБ
Вільне місце на накопичувачі	Від 100 МБ
Bluetooth	Наявність Bluetooth 4.0 або вище
Додатково	Дозволи на доступ до Bluetooth і файлів

Встановлення застосунку здійснюється стандартним способом через пакет встановлення або безпосередньо з середовища розробки при тестуванні. Після запуску користувач потрапляє на головний екран, де відображаються основні елементи керування: індикатори температури та вологості, кнопки для запуску

та зупинки збору даних, оновлення графіка, експорту інформації, перегляду історії та аналітики, а також доступ до налаштувань.

Щоб почати моніторинг, необхідно натиснути кнопку «Старт», після чого система почне зчитувати дані з сенсора через Bluetooth-з’єднання. Температура і вологість автоматично оновлюватимуться у вигляді текстових індикаторів та графіка в реальному часі. При натисканні кнопки «Стоп» обробка нових даних призупиняється, що дає змогу зберегти поточний стан вимірювань.

Кнопка «Оновити» очищає графік та текстові поля, дозволяючи користувачу почати нову сесію збору даних. Функція «Експорт» дає змогу зберегти отримані дані у вигляді текстового або CSV/JSON файлу для подальшого аналізу. За допомогою кнопки «Історія» можна переглядати попередні записи спостережень, а у розділі «Аналітика» представлено узагальнену статистику середніх значень температури і вологості за весь період спостереження.

Додаткові можливості передбачають зміну порогових значень для температури і вологості через розділ «Налаштування», а також вибір теми оформлення (світла або темна), що забезпечує комфортну роботу із застосунком у будь-який час доби. Інтерфейс програми оптимізовано для роботи на екранах різних розмірів, що гарантує зручність використання на широкому спектрі сучасних мобільних пристроїв.

4.4 Висновки

У ході проведеного тестування мобільного застосунку для моніторингу температури та вологості було підтверджено коректність роботи усіх розроблених модулів і функціональних компонентів. Застосунок стабільно здійснює зчитування даних із сенсора через Bluetooth-з’єднання, забезпечує їх візуалізацію у режимі реального часу, своєчасно генерує сповіщення при перевищенні заданих порогових меж та коректно обробляє дії користувача.

Особливу увагу приділено перевірці функцій керування збором даних, побудови графіків, експорту результатів моніторингу у текстові файли та перегляду історії спостережень. Інтерфейс програми продемонстрував високу адаптивність до розмірів екрана мобільного пристрою та забезпечив зручну і інтуїтивно зрозумілу взаємодію з користувачем.

Окремо було розроблено інструкцію користувача, що детально описує порядок роботи із застосунком, а також наведено системні вимоги до пристрою, що гарантують стабільність функціонування системи. Результати тестування свідчать про відповідність реалізованого застосунку початковим функціональним вимогам, що дозволяє вважати поставлені завдання успішно виконаними.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи на тему «Система моніторингу вологості і температури приміщення в режимі реального часу. Частина 2. Мобільний застосунок» було реалізовано повноцінний кросплатформенний мобільний застосунок, що забезпечує збір, обробку, візуалізацію та збереження кліматичних даних у реальному часі. Рішення поєднує апаратну частину на базі Arduino із сенсором DHT22 та модуль Bluetooth (ESP32) з програмною частиною, реалізованою на платформі .NET MAUI з використанням сучасних бібліотек.

У процесі розробки було здійснено глибокий аналіз предметної області, визначено актуальність створення мобільної системи контролю мікроклімату, проведено дослідження наявних програмно-апаратних рішень, їх переваг і недоліків. Це дозволило чітко сформулювати функціональні та нефункціональні вимоги до майбутнього застосунку, а також обґрунтувати вибір платформи .NET MAUI як сучасної кросплатформеної технології з широкими можливостями адаптації, зручного UI-розроблення, підтримки Bluetooth і активної розробницької екосистеми.

Було спроєктовано архітектуру застосунку з урахуванням модульності, підтримки масштабування та інтеграції з новими сенсорами або платформами. Реалізовано модулі збору даних через Bluetooth, обробки температури та вологості, відображення графіків з динамічним оновленням, збереження історії вимірювань у локальну пам’ять, експорт у формат текстового звіту, а також систему сповіщень про вихід параметрів за порогові межі. Важливою перевагою є адаптивний інтерфейс користувача з підтримкою темної та світлої тем, можливістю змінювати частоту оновлення, порогові значення та інші параметри.

Особливу увагу було приділено забезпеченню автономності застосунку – для основного функціоналу не потрібне підключення до Інтернету. Це дозволяє

використовувати систему в середовищах з обмеженим доступом до мережі, зокрема на складах, в теплицях, серверних кімнатах, лабораторіях тощо.

У ході тестування мобільний застосунок продемонстрував стабільність роботи Bluetooth-з’єднання, коректність зчитування та обробки кліматичних даних, ефективну візуалізацію інформації у графічному вигляді, своєчасну генерацію сповіщень, зручність експорту та надійність ведення історії спостережень. Інтерфейс показав високу адаптивність до різних типів пристроїв і розмірів екрана, що забезпечує зручність користування навіть на компактних смартфонах.

Отримані результати засвідчили, що поставлені у кваліфікаційній роботі завдання були реалізовані повністю. Система функціонує згідно технічного завдання, відповідає сучасним вимогам до мобільного програмного забезпечення, має високий рівень завершеності та практичної застосовності. Розроблений застосунок може бути використаний у реальному середовищі для контролю мікроклімату, інтегрований у більші IoT-системи або використаний як навчальний приклад під час вивчення дисциплін з програмування, IoT, систем реального часу та мобільної розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс] // DAN.IT education. – Режим доступу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/ – Назва з екрана.

2. Bluetooth Low Energy (BLE) Overview. Bluetooth.com. URL: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/low-energy/ (дата звернення: 12.03.2025).

3. Arduino Documentation. Arduino.cc. URL: https://docs.arduino.cc/ (дата звернення: 12.03.2025).

4. Govee Home App – Overview. Govee. URL: https://www.govee.com/pages/govee-home (дата звернення: 13.03.2025).

5. Mi Home App – Overview. Xiaomi. URL: https://www.mi.com/global/mihome/ (дата звернення: 13.03.2025).

6. SensorPush App – Overview. SensorPush. URL: https://www.sensorpush.com/pages/mobile-app (дата звернення: 13.03.2025).

7. Blynk IoT Platform Documentation. URL: https://docs.blynk.io/en/ (дата звернення: 19.03.2025).

8. Thingspeak Documentation. MathWorks. URL: https://thingspeak.com/docs (дата звернення: 19.03.2025).

9. Introduction to Data Logging. National Instruments. URL: https://www.ni.com/en-us/innovations/data-logging.html (дата звернення: 24.03.2025).

10. .NET MAUI documentation. Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/maui/ (дата звернення: 24.03.2025).

11. User Interface Design Basics. Nielsen Norman Group. URL: https://www.nngroup.com/articles/definition-user-interface-design/ (дата звернення: 24.03.2025).

12. Mobile UX Design Patterns. Smashing Magazine. URL: https://www.smashingmagazine.com/category/mobile/ (дата звернення: 29.03.2025).

13. Xamarin Essentials. Microsoft Learn. URL: https://learn.microsoft.com/en-us/xamarin/essentials/ (дата звернення: 05.04.2025).

14. Plugin.BLE – Cross-platform Bluetooth LE Plugin. URL: https://github.com/xabre/xamarin-bluetooth-le/ (дата звернення: 09.04.2025).

15. Visual Studio 2022 Documentation. Microsoft Docs. URL: https://learn.microsoft.com/en-us/visualstudio/releases/2022/release-notes (дата звернення: 15.04.2025).

16. MAUI Community Toolkit Documentation. URL: https://learn.microsoft.com/en-us/dotnet/communitytoolkit/maui/overview/ (дата звернення: 16.04.2025).

17. Syncfusion.Maui.Charts Documentation. Syncfusion. URL: https://help.syncfusion.com/maui/charts/getting-started (дата звернення: 18.04.2025).

18. Android Developers – Bluetooth Overview. Google Developers. URL: https://developer.android.com/guide/topics/connectivity/bluetooth (дата звернення: 18.04.2025).

19. C# Programming Guide. Microsoft Docs. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 22.04.2025).

20. Data Visualization Best Practices. Tableau. URL: https://www.tableau.com/learn/articles/data-visualization (дата звернення: 22.04.2025).

21. Романюк Олександр , Романюк Оксана , Чехместрук Роман. Комп`ютерна графіка . Романюк, О. Н. «Комп`ютерна графіка» : електронний навч. посіб. [Електронний ресурс] / О. Н. Романюк, О. В. Романюк, Р. Ю. Чехместрук. – Вінниця : ВНТУ, 2023. – 147 с.

ДОДАТКИ

63

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
« 25 » березня 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Система моніторингу вологості і температури приміщення в режимі реального часу. Частина 2. Мобільний застосунок» за спеціальністю
121 Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. О.В. Мельник
« 24 » березня 2025 р.
Виконав:
студент гр.5ІІІ-21б Д.С. Чістяков
« 24 » березня 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Система моніторингу вологості і температури приміщення в режимі реального часу. Частина 2. Мобільний застосунок».

Галузь застосування – приміщення.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 « 20 » березня 2025 р. по ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою роботи є розробка мобільного застосунку для моніторингу температури та вологості повітря в реальному часі з функцією візуалізації, збереження історії вимірювань та системою сповіщень.

Призначення роботи – створення адаптивного мобільного інструменту для контролю кліматичних параметрів у побутових і професійних умовах.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР.

1. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс] // DAN.IT education. – Режим доступу: https://dan-it.com.ua/uk/blog/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/.
2. .NET MAUI documentation. Microsoft Learn. – URL: https://learn.microsoft.com/en-us/dotnet/maui/.
3. Arduino Documentation. Arduino.cc. – URL: https://docs.arduino.cc/.

5. Технічні вимоги

Платформа розробки – .NET MAUI; мова програмування – C#; цільова операційна система – Android; тип підключення до сенсорів – Bluetooth; вхідні дані – температура повітря, відносна вологість, інтервал збору даних; елементи інтерфейсу – кнопки запуску, зупинки, експорту, перемикач темного/світлого режиму; графічний режим – виведення поточних значень температури та вологості на екран, побудова графіка в реальному часі з автоматичним оновленням; сповіщення – виведення повідомлень при перевищенні порогових значень, заданих користувачем; збереження історії вимірювань у внутрішню пам’ять пристрою; формат експорту даних – текстовий файл.

6. Конструктивні вимоги.

Інтерфейс застосунку має бути інтуїтивно зрозумілим, адаптованим для мобільних пристроїв з різними розмірами екранів.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до БКР;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз предметної області, визначення мети, завдань, обґрунтування вибору апаратної та програмної платформи	25.03.25 – 31.03.25
2	Проєктування архітектури застосунку, моделювання алгоритмів, побудова діаграм і блок-схем	01.04.25 – 07.04.25
3	Реалізація інтерфейсу, Bluetooth-з'єднання та збору даних з сенсор	08.04.25 – 14.04.25
4	Розробка модуля візуалізації, історії вимірювань, налаштування порогів і генерації сповіщень	15.04.25 – 30.04.25
5	Тестування системи, написання інструкції користувача,	01.05.25 – 30.05.25

10. Порядок контролю та прийняття.

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Захист бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: система моніторингу вологості і температури приміщення в режимі реального часу. Частина 2. Мобільний застосунок

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, ФІТКІ, 5ПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 4,8 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

(прізвище, ініціали, посада) (підпис)

(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник (підпис) Мельник О.В., к.т.н., доц. каф. ПЗ
(прізвище, ініціали, посада)

Здобувач (підпис) Чістяков Д.С.
(прізвище, ініціали)

Додаток В – Лістинг програми

Розмітка головного інтерфейсу застосунку (MainPage.xaml)

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:syncfusion="clr-
namespace:Syncfusion.Maui.Charts;assembly=Syncfusion.Maui.Charts"
             x:Class="TempHumidityMonitor.MainPage">

    <ScrollView>
        <VerticalStackLayout Padding="15" Spacing="20"
HorizontalOptions="Center">

            <!-- Назва -->
            <Label Text="Моніторинг температури і вологості"
                   FontSize="24"
                   HorizontalOptions="Center" />

            <!-- Поточні значення -->
            <Label x:Name="TempLabel"
                   Text="Температура: --°C"
                   FontSize="18"
                   HorizontalOptions="Center" />

            <Label x:Name="HumLabel"
                   Text="Вологість: --%"
                   FontSize="18"
                   HorizontalOptions="Center" />
```

```
<!-- Графік -->
<syncfusion:SfCartesianChart HeightRequest="280">
    <syncfusion:SfCartesianChart.Legend>
        <syncfusion:ChartLegend Placement="Top" />
    </syncfusion:SfCartesianChart.Legend>

    <syncfusion:SfCartesianChart.PaletteBrushes>
        <SolidColorBrush Color="Red" />
        <SolidColorBrush Color="#0000FF" />
    </syncfusion:SfCartesianChart.PaletteBrushes>

    <syncfusion:SfCartesianChart.XAxes>
        <syncfusion:CategoryAxis ShowMajorGridLines="True">
            <syncfusion:CategoryAxis.Title>
                <syncfusion:ChartAxisTitle Text="Час" />
            </syncfusion:CategoryAxis.Title>
        </syncfusion:CategoryAxis>
    </syncfusion:SfCartesianChart.XAxes>

    <syncfusion:SfCartesianChart.YAxes>
        <syncfusion:NumericalAxis ShowMajorGridLines="True">
            <syncfusion:NumericalAxis.Title>
                <syncfusion:ChartAxisTitle Text="Значення" />
            </syncfusion:NumericalAxis.Title>
        </syncfusion:NumericalAxis>
    </syncfusion:SfCartesianChart.YAxes>

    <syncfusion:SfCartesianChart.Series>
        <syncfusion:LineSeries
```

```
                ItemsSource="{Binding ChartData}"
                XBindingPath="Time"
                YBindingPath="Temperature"
                Label="Температура"
                EnableTooltip="True"
                ShowDataLabels="False"
                StrokeWidth="3" />

            <syncfusion:LineSeries
                ItemsSource="{Binding ChartData}"
                XBindingPath="Time"
                YBindingPath="Humidity"
                Label="Вологість"
                EnableTooltip="True"
                ShowDataLabels="False"
                StrokeWidth="3" />
        </syncfusion:SfCartesianChart.Series>
    </syncfusion:SfCartesianChart>

    <!-- Інтервал -->
    <HorizontalStackLayout HorizontalOptions="Center" Spacing="10"
Margin="0,-10,0,0">
        <Label Text="Інтервал:" FontSize="16" VerticalOptions="Center" />
        <Picker x:Name="intervalPicker"
                Title="Оберіть інтервал"
                WidthRequest="120"
                SelectedIndexChanged="OnIntervalChanged">
            <Picker.Items>
                <x:String>1 сек</x:String>
```

```
                <x:String>5 сек</x:String>
                <x:String>10 сек</x:String>
            </Picker.Items>
        </Picker>
    </HorizontalStackLayout>

    <!-- Кнопки керування -->
    <HorizontalStackLayout HorizontalOptions="Center" Spacing="20"
Margin="0,-10,0,0">
        <Button Text="Старт" Clicked="OnStartClicked" />
        <Button Text="Стоп" Clicked="OnStopClicked" />
        <Button Text="Оновити" Clicked="OnRefreshClicked" />
    </HorizontalStackLayout>

    <!-- Нижні іконки дій -->
    <HorizontalStackLayout HorizontalOptions="Center" Spacing="30"
Margin="0,20,0,10">
        <VerticalStackLayout Spacing="2" HorizontalOptions="Center">
            <ImageButton Source="export.png" WidthRequest="40"
HeightRequest="40" BackgroundColor="Transparent"
Clicked="OnExportImageClicked" />
            <Label Text="Експорт" FontSize="12" HorizontalOptions="Center" />
        </VerticalStackLayout>

        <VerticalStackLayout Spacing="2" HorizontalOptions="Center">
            <ImageButton Source="log.png" WidthRequest="40"
HeightRequest="40" BackgroundColor="Transparent" Clicked="OnLogClicked" />
            <Label Text="Історія" FontSize="12" HorizontalOptions="Center" />
        </VerticalStackLayout>
```

```
            <VerticalStackLayout Spacing="2" HorizontalOptions="Center">
                <ImageButton Source="analytics.png" WidthRequest="40"
HeightRequest="40" BackgroundColor="Transparent"
Clicked="OnAnalyticsClicked" />
                <Label Text="Аналітика" FontSize="12" HorizontalOptions="Center"
/>
            </VerticalStackLayout>

            <VerticalStackLayout Spacing="2" HorizontalOptions="Center">
                <ImageButton Source="settings.png" WidthRequest="40"
HeightRequest="40" BackgroundColor="Transparent" Clicked="OnSettingsClicked"
/>
                <Label Text="Налашт." FontSize="12" HorizontalOptions="Center" />
            </VerticalStackLayout>
        </HorizontalStackLayout>

    </VerticalStackLayout>
  </ScrollView>
</ContentPage>
```

Логіка обробки подій головної сторінки (MainPage.xaml.cs)

```
using System.Text;
using System.IO;
using Microsoft.Maui.Storage;
using Microsoft.Maui.ApplicationModel.DataTransfer;
using Microsoft.Maui.Dispatching;
using System.Collections.ObjectModel;
```

```
using System.Text.Json;

namespace TempHumidityMonitor;

public partial class MainPage : ContentPage
{
    public ObservableCollection<SensorData> ChartData { get; set; } = new();

    private IDispatcherTimer timer;
    private Random random = new();
    private bool tempWarningShown = false;
    private bool humWarningShown = false;


    public MainPage()
    {
        InitializeComponent();

        timer = Dispatcher.CreateTimer();

        this.BindingContext = this;

        intervalPicker.SelectedIndex = 1;

        TempLabel.Text = $"Температура: --°C";
        HumLabel.Text = $"Вологість: --%";

        timer.Interval = TimeSpan.FromSeconds(5);
        timer.Tick += Timer_Tick;
```

```
}

private void OnStartClicked(object sender, EventArgs e)
{
    if (!timer.IsRunning)
        timer.Start();
}

private void OnStopClicked(object sender, EventArgs e)
{
    if (timer.IsRunning)
        timer.Stop();
}

private void OnRefreshClicked(object sender, EventArgs e)
{
    ChartData.Clear();
    TempLabel.Text = "Температура: --°C";
    HumLabel.Text = "Вологість: --%";
}

private async void OnAnalyticsClicked(object sender, EventArgs e)
{
    await Navigation.PushAsync(new AnalyticsPage(ChartData));
}

private async void OnNotificationClicked(object sender, EventArgs e)
{
```

```
        await DisplayAlert("Сповіщення", "Тут буде виводитись список активних сповіщень.", "OK");
    }

    private async void OnLogClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new LogPage()); // або Show лог-діалог, залежно від реалізації
    }

    private async void Timer_Tick(object? sender, EventArgs e)
    {
        double temperature, humidity;
        string time = DateTime.Now.ToString("HH:mm:ss");

        try
        {
            var service = new SensorService();
            var data = await service.GetDataFromBluetoothAsync();
            temperature = data.Temperature;
            humidity = data.Humidity;
        }
        catch (Exception ex)
        {
            await DisplayAlert("Помилка", "Не вдалося зчитати дані з сенсора: " + ex.Message, "OK");
            return;
        }
```

```
        double tempMin = double.Parse(Preferences.Get("TempMin", "15"));
        double tempMax = double.Parse(Preferences.Get("TempMax", "30"));
        double humMin = double.Parse(Preferences.Get("HumMin", "30"));
        double humMax = double.Parse(Preferences.Get("HumMax", "70"));

        if ((temperature > tempMax || temperature < tempMin) &&
!tempWarningShown)
        {
            tempWarningShown = true;
            await DisplayAlert("Увага!", $"Температура вийшла за межі:
{temperature}°C", "OK");

            string alertText = $"[{time}] Перевищення температури: {temperature}°C";
            string alertPath = Path.Combine(FileSystem.AppDataDirectory, "alerts.txt");
            File.AppendAllText(alertPath, alertText + Environment.NewLine);
        }
        else if (temperature <= tempMax && temperature >= tempMin)
        {
            tempWarningShown = false;
        }

        if ((humidity > humMax || humidity < humMin) && !humWarningShown)
        {
            humWarningShown = true;
            await DisplayAlert("Увага!", $"Вологість вийшла за межі: {humidity}%",
"OK");

            string alertText = $"[{time}] Перевищення вологості: {humidity}%";
            string alertPath = Path.Combine(FileSystem.AppDataDirectory, "alerts.txt");
```

```
            File.AppendAllText(alertPath, alertText + Environment.NewLine);
        }
        else if (humidity <= humMax && humidity >= humMin)
        {
            humWarningShown = false;
        }

        TempLabel.Text = $"Температура: {temperature}°C";
        HumLabel.Text = $"Вологість: {humidity}%";

        ChartData.Add(new SensorData
        {
            Time = time,
            Temperature = temperature,
            Humidity = humidity
        });

        if (ChartData.Count > 20)
            ChartData.RemoveAt(0);

        string logLine = $"[{DateTime.Now:yyyy-MM-dd HH:mm:ss}] Температура:
{temperature}°C | Вологість: {humidity}%";
        string logPath = Path.Combine(FileSystem.AppDataDirectory, "log.txt");
        File.AppendAllText(logPath, logLine + Environment.NewLine);
    }


    private async void OnExportClicked(object sender, EventArgs e)
    {
```

```
        if (ChartData.Count == 0)
        {
            await DisplayAlert("Увага", "Немає даних для експорту.", "OK");
            return;
        }

        var text = new StringBuilder();
        text.AppendLine("Журнал спостереження:");
        text.AppendLine("--------------------------------------");

        foreach (var data in ChartData)
        {
            text.AppendLine($"Час: {data.Time} | Температура: {data.Temperature}°C |
Вологість: {data.Humidity}%");
        }

        string fileName = $"log_{DateTime.Now:yyyyMMdd_HHmmss}.txt";
        string filePath = Path.Combine(FileSystem.CacheDirectory, fileName);

        File.WriteAllText(filePath, text.ToString());

        await Share.RequestAsync(new ShareFileRequest
        {
            Title = "Поділитись TXT-файлом",
            File = new ShareFile(filePath)
        });
    }

    private async void OnExportImageClicked(object sender, EventArgs e)
```

```
    {
        var format = await DisplayActionSheet("Формат експорту", "Скасувати", null,
"CSV", "JSON");
        if (format == "CSV")
            await ExportCsv();
        else if (format == "JSON")
            await ExportJson();
    }

    private async Task ExportCsv()
    {
        if (ChartData.Count == 0)
        {
            await DisplayAlert("Увага", "Немає даних для експорту.", "OK");
            return;
        }

        var sb = new StringBuilder();
        sb.AppendLine("Час,Температура,Вологість");

        foreach (var d in ChartData)
            sb.AppendLine($"{d.Time},{d.Temperature},{d.Humidity}");

        var fileName = $"data_{DateTime.Now:yyyyMMdd_HHmmss}.csv";
        var path = Path.Combine(FileSystem.AppDataDirectory, fileName);

        File.WriteAllText(path, sb.ToString());

        await Share.RequestAsync(new ShareFileRequest
```

```
    {
        Title = "Поділитися CSV",
        File = new ShareFile(path)
    });
}

private async Task ExportJson()
{
    if (ChartData.Count == 0)
    {
        await DisplayAlert("Увага", "Немає даних для експорту.", "OK");
        return;
    }

    var json = JsonSerializer.Serialize(ChartData);
    var fileName = $"data_{DateTime.Now:yyyyMMdd_HHmmss}.json";
    var path = Path.Combine(FileSystem.AppDataDirectory, fileName);

    File.WriteAllText(path, json);

    await Share.RequestAsync(new ShareFileRequest
    {
        Title = "Поділитися JSON",
        File = new ShareFile(path)
    });
}

private void OnIntervalChanged(object sender, EventArgs e)
{
```

```
    if (timer == null || intervalPicker.SelectedIndex == -1)
        return;

    string? selected = intervalPicker.SelectedItem as string;
    if (string.IsNullOrEmpty(selected))
        return;

    int seconds = selected switch
    {
        "1 сек" => 1,
        "5 сек" => 5,
        "10 сек" => 10,
        _ => 5
    };

    bool wasRunning = timer.IsRunning;

    if (wasRunning)
        timer.Stop();

    timer.Interval = TimeSpan.FromSeconds(seconds);

    if (wasRunning)
        timer.Start();
}

private async void OnHistoryClicked(object sender, EventArgs e)
{
    string filePath = Path.Combine(FileSystem.AppDataDirectory, "history.txt");
```

```
        if (!File.Exists(filePath))
        {
            await DisplayAlert("Історія", "Файл історії порожній.", "OK");
            return;
        }

        string content = File.ReadAllText(filePath);

        bool clear = await DisplayAlert("Історія (останні записи)", content,
"Очистити", "Закрити");

        if (clear)
        {
            File.Delete(filePath);
            await DisplayAlert("Очищено", "Історію очищено успішно.", "OK");
        }
    }

    private async void OnBellClicked(object sender, EventArgs e)
    {
        string filePath = Path.Combine(FileSystem.AppDataDirectory, "alerts.txt");

        if (!File.Exists(filePath))
        {
            await DisplayAlert("Сповіщення", "Немає збережених сповіщень.",
"OK");
            return;
        }
```

```
        string content = File.ReadAllText(filePath);

        bool clear = await DisplayAlert("Сповіщення (останні)", content, "Очистити",
"Закрити");

        if (clear)
        {
            File.Delete(filePath);
            await DisplayAlert("Очищено", "Сповіщення очищено.", "OK");
        }
    }

    private async void OnSettingsClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new ThresholdPage());
    }

}

public class SensorData
{
    public string Time { get; set; } = "";
    public double Temperature { get; set; } = 0;
    public double Humidity { get; set; } = 0;
}
```

Модуль зчитування даних із сенсора (SensorService.cs)

```
using Plugin.BLE;
using Plugin.BLE.Abstractions.Contracts;
using Plugin.BLE.Abstractions.Exceptions;

namespace TempHumidityMonitor;

public class SensorService
{
    private readonly IBluetoothLE _bluetooth;
    private readonly IAdapter _adapter;

    public SensorService()
    {
        _bluetooth = CrossBluetoothLE.Current;
        _adapter = CrossBluetoothLE.Current.Adapter;
    }

    public async Task<(double Temperature, double Humidity)>
GetDataFromBluetoothAsync()
    {
        // Пошук пристрою ESP32
        await _adapter.StartScanningForDevicesAsync();
        var knownDevice = _adapter.DiscoveredDevices.FirstOrDefault(d => d.Name
!= null && d.Name.Contains("ESP32"));

        if (knownDevice == null)
            throw new Exception("Пристрій ESP32 не знайдено.");
```

```
        // Пошук сервісу і характеристик
        var service = await knownDevice.GetServiceAsync(Guid.Parse("0000181A-
0000-1000-8000-00805F9B34FB")); // Environmental Sensing

        var tempChar = await service.GetCharacteristicAsync(Guid.Parse("00002A6E-
0000-1000-8000-00805F9B34FB")); // Temperature
        var humChar = await service.GetCharacteristicAsync(Guid.Parse("00002A6F-
0000-1000-8000-00805F9B34FB")); // Humidity

        if (tempChar == null || humChar == null)
            throw new Exception("Характеристики не знайдено.");

        var (tempData, tempCode) = await tempChar.ReadAsync();
        var (humData, humCode) = await humChar.ReadAsync();

        if (tempData == null || tempData.Length < 2)
            throw new Exception("Дані температури не зчитано.");

        if (humData == null || humData.Length < 2)
            throw new Exception("Дані вологості не зчитано.");

        double temperature = BitConverter.ToInt16(tempData, 0) / 100.0;
        double humidity = BitConverter.ToInt16(humData, 0) / 100.0;

        return (temperature, humidity);
    }
}
```

Інтерфейс сторінки налаштування порогів (ThresholdPage.xaml)

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="TempHumidityMonitor.ThresholdPage"
             Title="Налаштування порогів">

    <ScrollView>
        <VerticalStackLayout Padding="20" Spacing="15">

            <Label Text="Температура (°C)" FontSize="18" />
            <Entry x:Name="entryTempMin" Placeholder="Мінімум"
Keyboard="Numeric"/>
            <Entry x:Name="entryTempMax" Placeholder="Максимум"
Keyboard="Numeric"/>

            <Label Text="Вологість (%)" FontSize="18" />
            <Entry x:Name="entryHumMin" Placeholder="Мінімум"
Keyboard="Numeric"/>
            <Entry x:Name="entryHumMax" Placeholder="Максимум"
Keyboard="Numeric"/>

            <HorizontalStackLayout Spacing="10">
                <Label Text="Темна тема" VerticalOptions="Center"/>
                <Switch x:Name="themeSwitch" Toggled="OnThemeToggled"/>
            </HorizontalStackLayout>

            <Button Text="Зберегти" Clicked="OnSaveClicked" />
        </VerticalStackLayout>
```

```
    </ScrollView>
</ContentPage>
```

Логіка сторінки налаштування порогів (ThresholdPage.xaml.cs)

```
using Microsoft.Maui.Storage;

namespace TempHumidityMonitor;

public partial class ThresholdPage : ContentPage
{
    public ThresholdPage()
    {
        InitializeComponent();

        themeSwitch.IsToggled = Preferences.Get("DarkThemeEnabled", false);

        // Завантаження збережених значень
        entryTempMin.Text = Preferences.Get("TempMin", "15");
        entryTempMax.Text = Preferences.Get("TempMax", "30");
        entryHumMin.Text = Preferences.Get("HumMin", "30");
        entryHumMax.Text = Preferences.Get("HumMax", "70");

    }

    private async void OnSaveClicked(object sender, EventArgs e)
    {
        try
        {
```

```
            // Перевірка введення
            _ = double.Parse(entryTempMin.Text);
            _ = double.Parse(entryTempMax.Text);
            _ = double.Parse(entryHumMin.Text);
            _ = double.Parse(entryHumMax.Text);

            // Збереження
            Preferences.Set("TempMin", entryTempMin.Text);
            Preferences.Set("TempMax", entryTempMax.Text);
            Preferences.Set("HumMin", entryHumMin.Text);
            Preferences.Set("HumMax", entryHumMax.Text);

            await DisplayAlert("Успіх", "Порогові значення збережено!", "OK");
            await Navigation.PopAsync();
        }
        catch
        {
            await DisplayAlert("Помилка", "Будь ласка, введіть числові значення.",
"OK");
        }
    }

    private void OnThemeToggled(object sender, ToggledEventArgs e)
    {
        Preferences.Set("DarkThemeEnabled", e.Value);

        App.Current.UserAppTheme = e.Value ? AppTheme.Dark : AppTheme.Light;
    }
}
```

Розмітка сторінки перегляду історії вимірювань (LogPage.xaml)

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="TempHumidityMonitor.LogPage"
             Title="Історія спостережень">

    <VerticalStackLayout Padding="15" Spacing="10">

        <!-- Верхній рядок з іконками -->
        <Grid ColumnDefinitions="*,Auto,Auto">
            <Label Text="Історія спостережень"
                   FontSize="20"
                   VerticalOptions="Center"
                   HorizontalOptions="Start" />
        </Grid>

        <!-- Вміст журналу -->
        <ScrollView>
            <Label x:Name="LogTextLabel"
                   FontSize="14"
                   TextColor="Black"
                   LineBreakMode="WordWrap" />
        </ScrollView>
        <!-- Кнопка очищення -->
        <Button Text="Очистити журнал"
                BackgroundColor="Crimson"
                TextColor="White"
                Clicked="OnClearLogClicked" />
```

```
    </VerticalStackLayout>
</ContentPage>
```

Логіка роботи сторінки перегляду історії вимірювань (LogPage.xaml.cs)

```
using System.IO;
using Microsoft.Maui.Storage;

namespace TempHumidityMonitor;

public partial class LogPage : ContentPage
{
    private readonly string logPath;

    public LogPage()
    {
        InitializeComponent();
        logPath = Path.Combine(FileSystem.AppDataDirectory, "log.txt");
        LoadLog();
    }

    private void LoadLog()
    {
        var label = this.FindByName<Label>("LogTextLabel");

        if (File.Exists(logPath))
            label.Text = File.ReadAllText(logPath);
        else
```

```
            label.Text = "Журнал порожній.";
    }

    private async void OnClearLogClicked(object sender, EventArgs e)
    {
        var label = this.FindByName<Label>("LogTextLabel");

        if (File.Exists(logPath))
            File.Delete(logPath);

        label.Text = "Журнал очищено.";
        await DisplayAlert("Успішно", "Історію очищено", "OK");
    }

    private async void OnNotificationClicked(object sender, EventArgs e)
    {
        await DisplayAlert("Сповіщення", "Наразі немає нових сповіщень", "OK");
    }

    private async void OnSettingsClicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new ThresholdPage());
    }
}
```

Додаток Г – Графічна частина

ГАФІЧНА ЧАСТИНА

СИСТЕМА МОНІТОРИНГУ ВОЛОГОСТІ І ТЕМПЕРАТУРИ ПРИМІЩЕННЯ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ. ЧАСТИНА 2. МОБІЛЬНИЙ ЗАСТОСУНОК

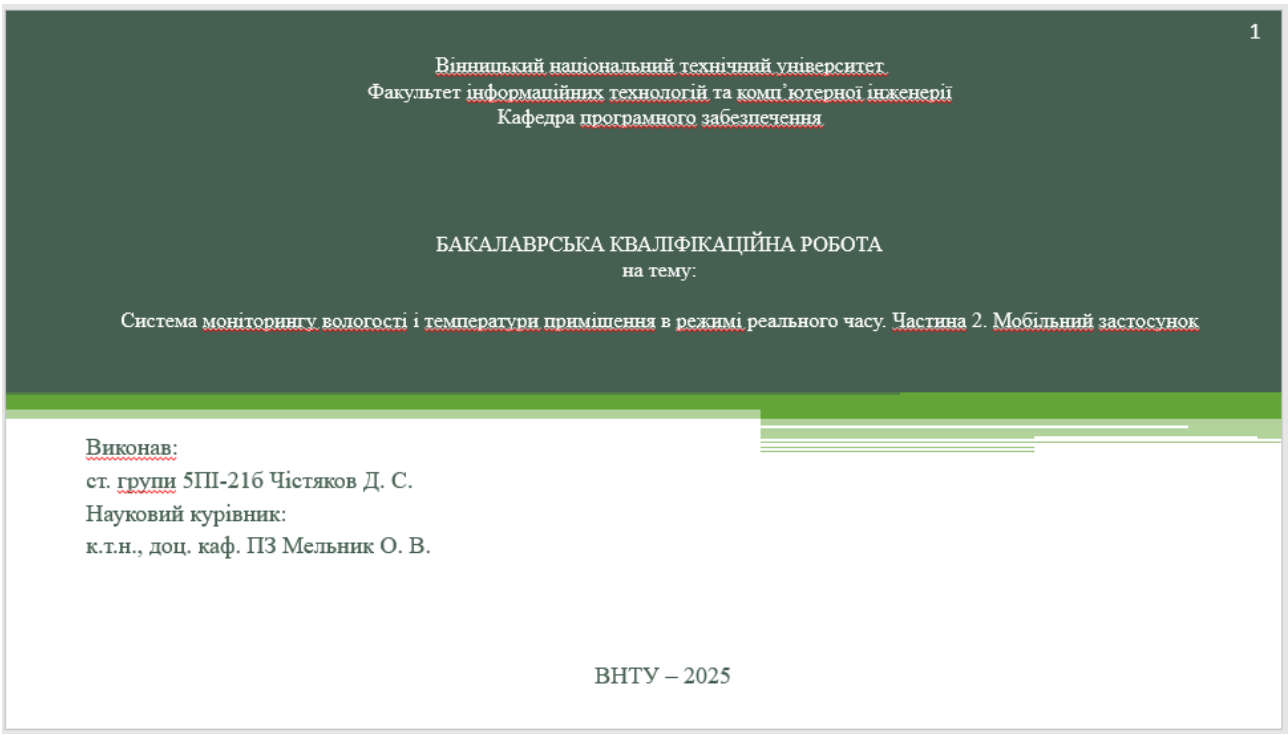

Рисунок Г.1 – Титульна сторінка

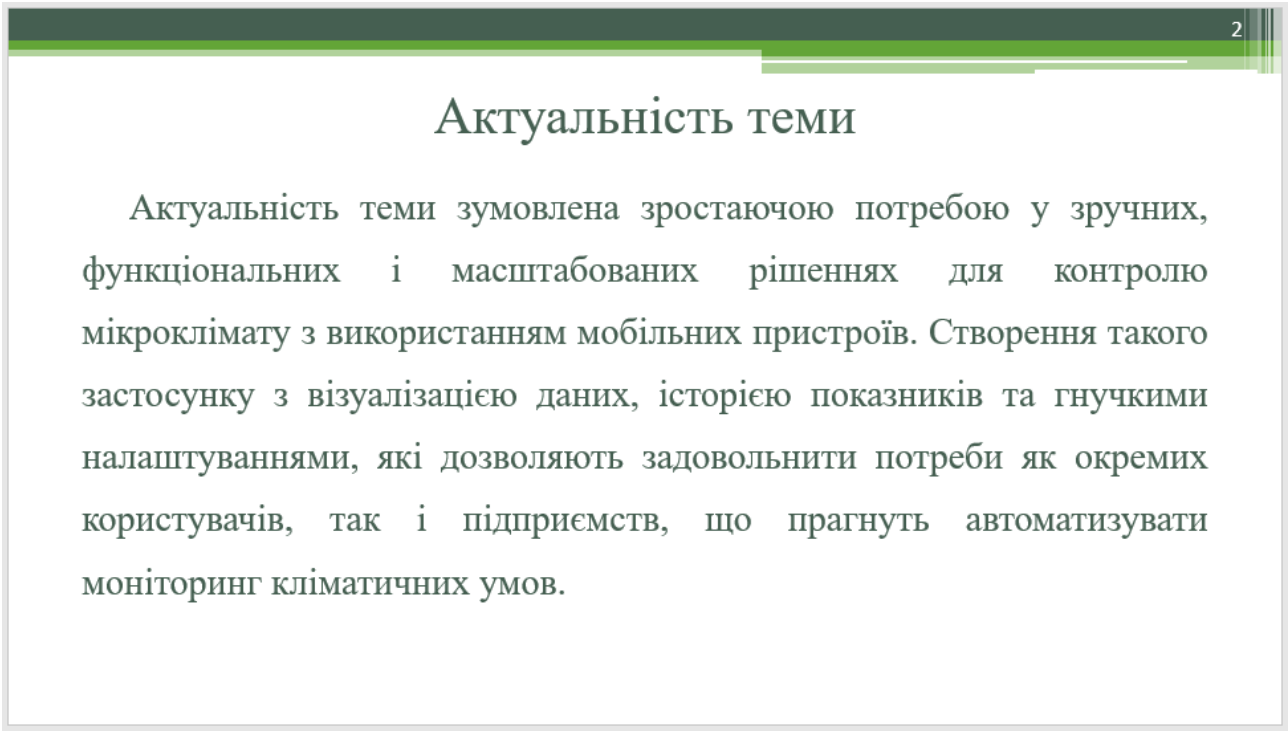

Рисунок Г.2 – Актуальність теми

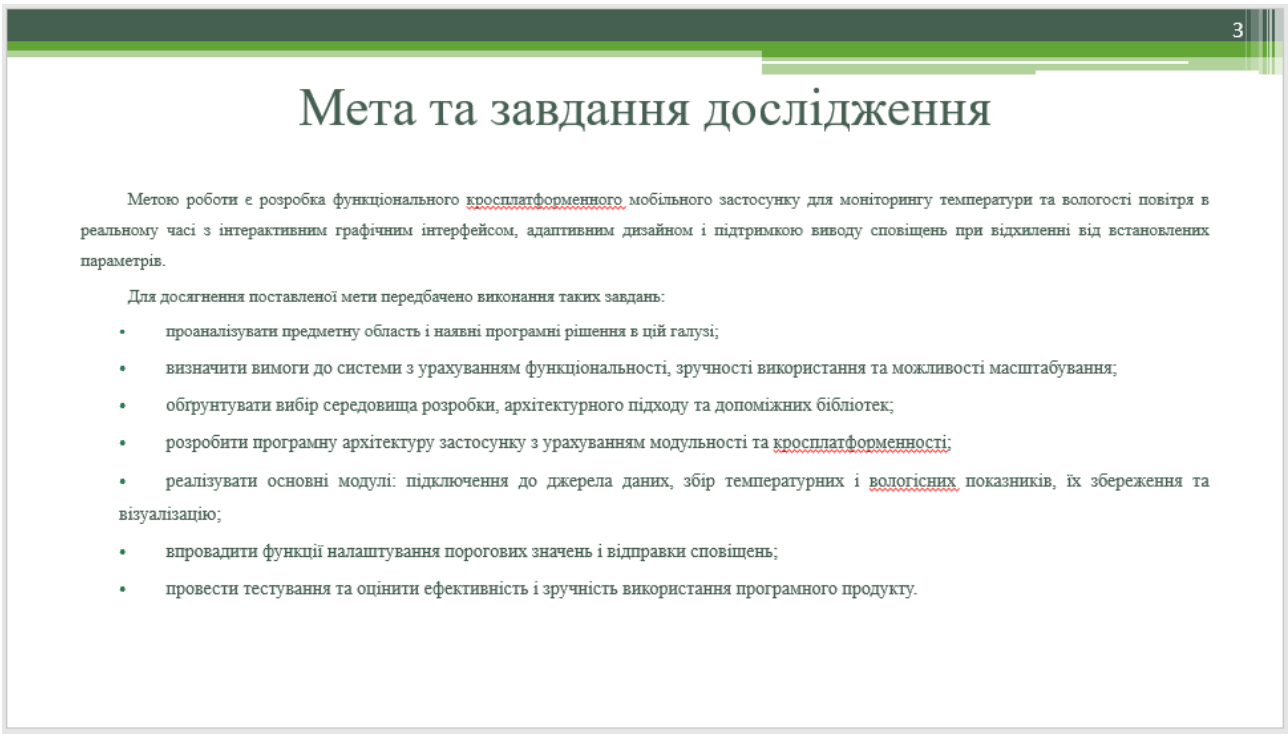

Рисунок Г.3 – Мета та завдання дослідження

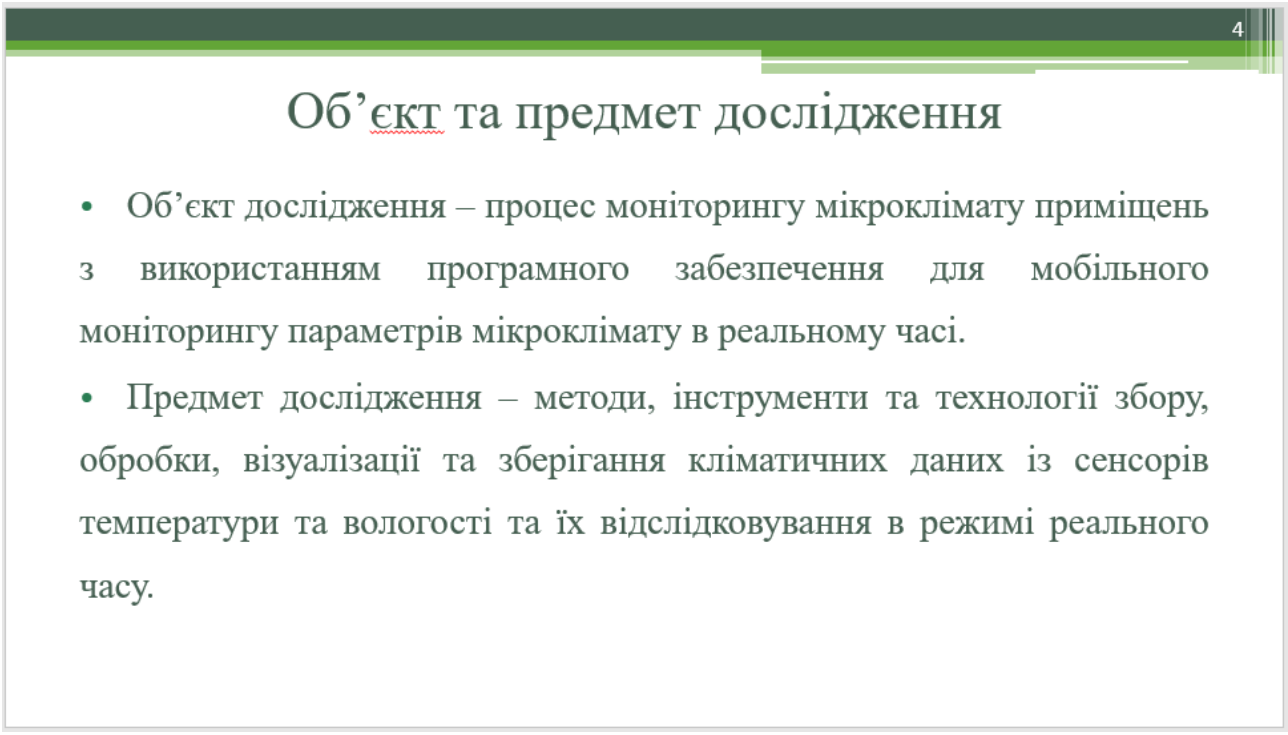

Рисунок Г.4 – Об’єкт та предмет дослідження

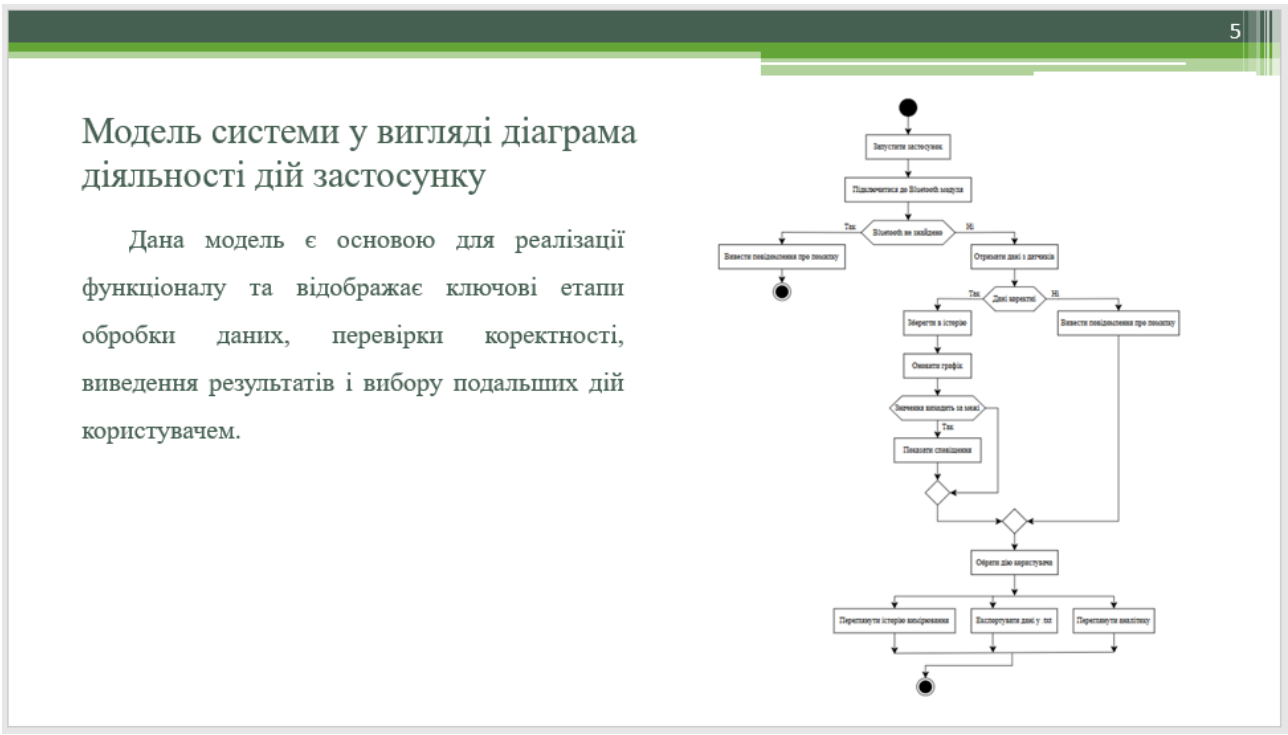

Рисунок Г.5 – Модель системи у вигляді діаграма діяльності дій застосунку

Рисунок Г.6 – Алгоритм роботи програмного застосунку

Рисунок Г.7 – Алгоритм функціонування системи моніторингу температури і вологості

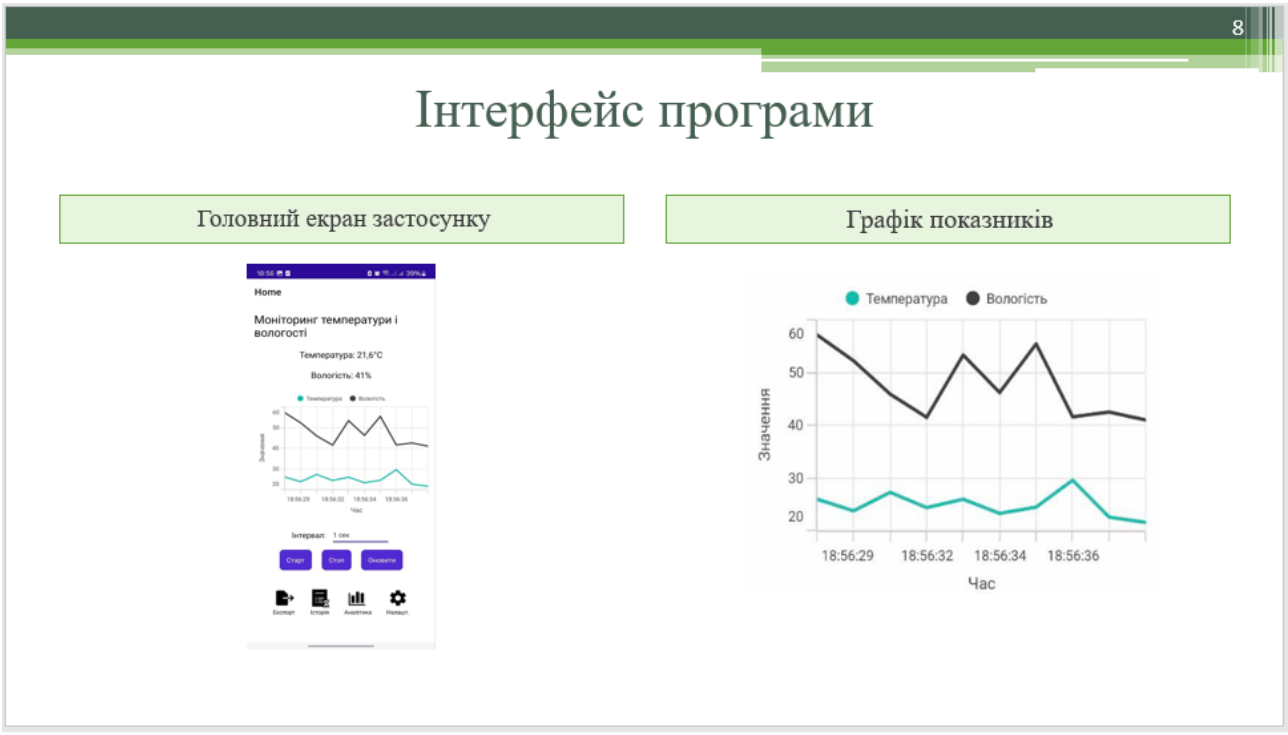

Рисунок Г.8 – Інтерфейс програми

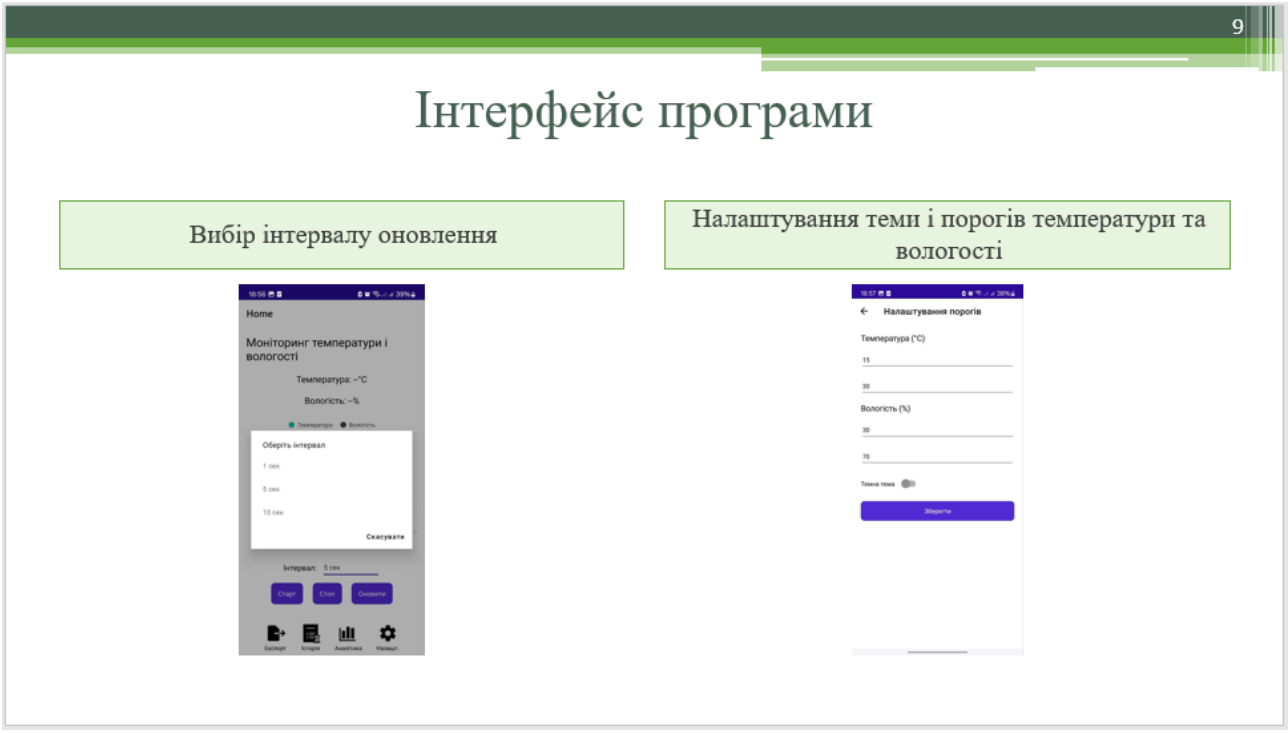

Рисунок Г.9 – Інтерфейс програми

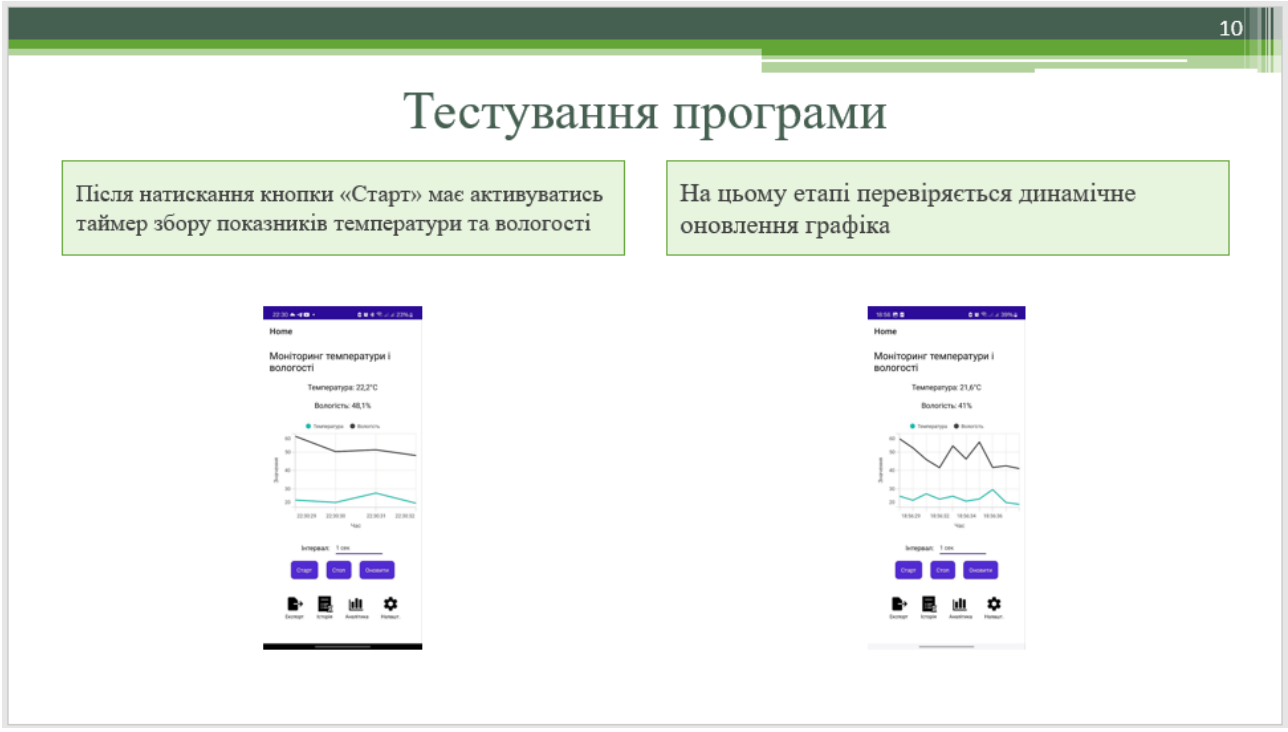

Рисунок Г.10 – Тестування програми

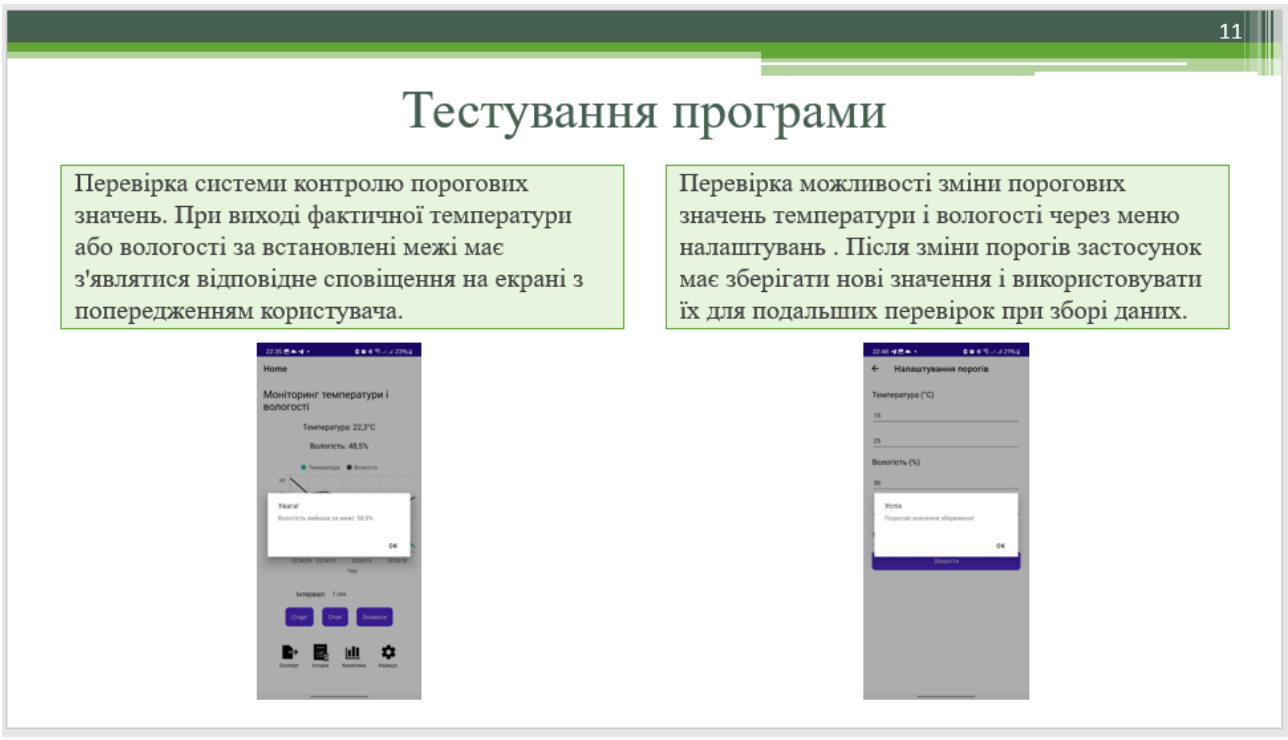

Рисунок Г.11 – Тестування програми

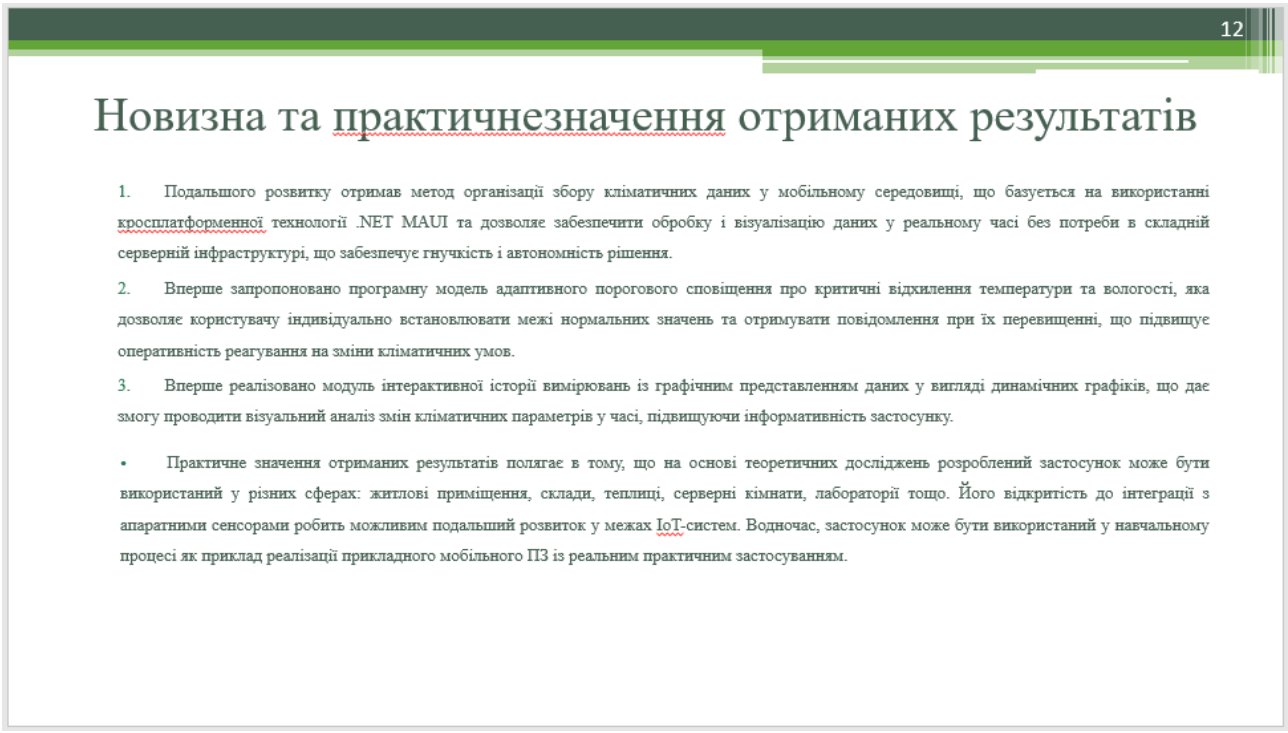

Рисунок Г.12 – Новизна та практичнезначення отриманих результатів

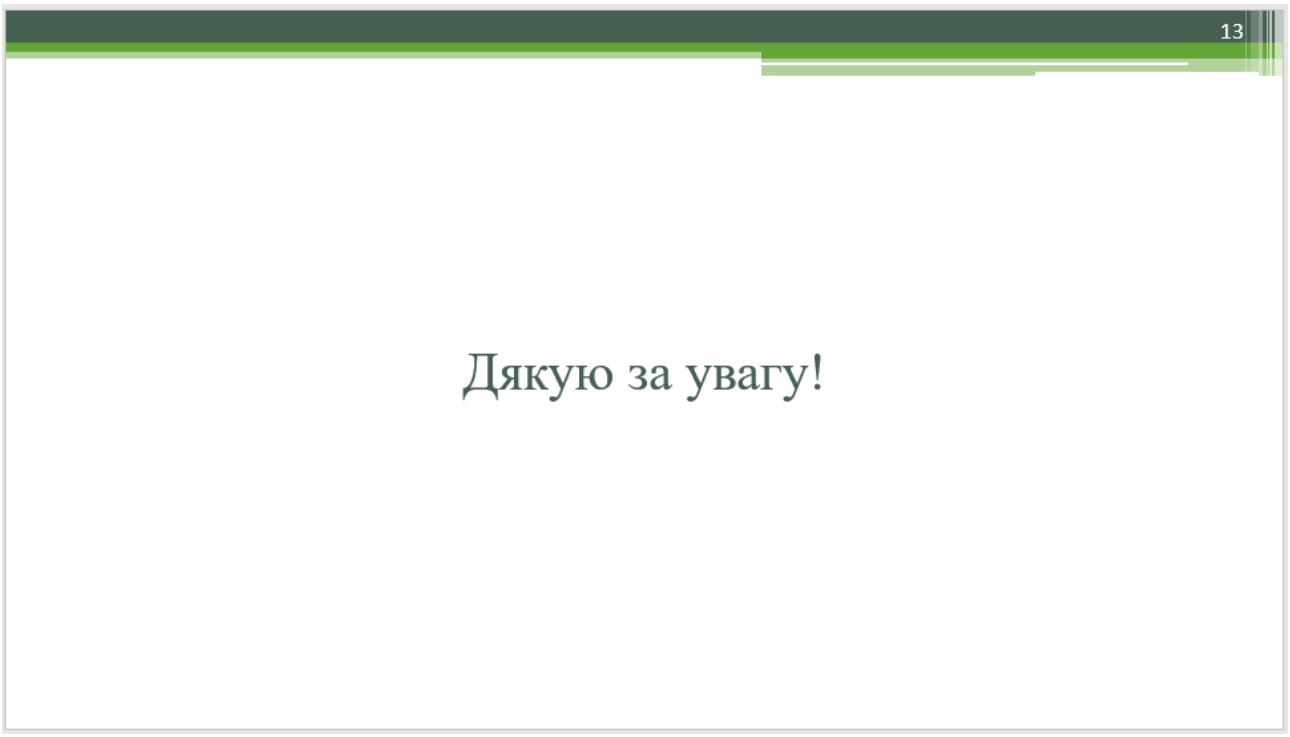

Рисунок Г.13 – Кінець презентації