Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

КУРСОВИЙ ПРОЕКТ

з дисципліни «Інтернет речей»

на тему: «Емулятор системи мікроклімату»

08-31.КП.ІР.009.00.000 ПЗ

Студента 4 курсу групи 1ІСТ-22б,
спеціальності 126 – Інформаційні системи та технології
Олексій ІЩЕНКО
Керівник:
к.т.н., доц, каф. АІІТ Ярослав КУЛИК

Кількість балів: ___Оцінка: ECTS

Члени комісії:

_____

_____

м. Вінниця 2025 рік
Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Зав. кафедри АІІТ, проф., д.т.н.

_____ Олег БІСІКАЛО
(підпис)
„____” _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсовий проект з дисципліни «Інтернет речей»
студенту Іщенку Олексію Руслановичу групи 1ІСТ-22б
ТЕМА: Емулятор системи мікроклімату

Постановка задачі: розробити програмний емулятор IoT-системи керування мікрокліматом приміщення та демонстраційну апаратну схему на базі Arduino.
Вимоги до функціональності:

- Змоделювати температуру, вологість, освітленість та присутність людини;
- Реалізувати автоматичне керування «обігрівачем», «кондиціонером» і «освітленням» за показами датчиків;
- Забезпечити вибір режиму роботи системи та візуалізацію станів у графічному інтерфейсі і на Arduino;

Зміст пояснювальної записки до курсової роботи:
Вступ
1. Опис предметної області
2. Системи аналоги
3. Технічні засоби
4. Аналіз методів вирішення поставленої задачі
5. Опис логічної структури
6. Перевірка роботи системи
7. Вхідні та вихідні дані
8. Інструкція користувача
Висновок
Перелік посилань
Додатки
Дата видачі «____» _____ 2025 р. Керівник _____ *(підпис)*

Завдання отримав _____ *(підпис)*

АНОТАЦІЯ

У курсовому проекту розроблено програмний емулятор IoT-системи керування мікрокліматом приміщення на базі технології WPF та мови C#. У роботі розглянуто предметну область систем «розумного дому» та обґрунтовано доцільність використання програмного емулятора для відлагодження алгоритмів без потреби у повному комплекті апаратних засобів. Емулятор моделює температуру, вологість, освітленість, віртуальний час доби та присутність людини, а також реалізує автоматичне керування «обігрівачем», «кондиціонером» і «освітленням» у кількох режимах роботи (Comfort, Eco, Away). Описано логічну структуру системи: модель середовища, віртуальні сенсори, виконавчі пристрої, контролер та графічний інтерфейс користувача з журналом подій. Для демонстрації можливої апаратної реалізації побудовано схему прототипу на Arduino Uno з датчиком температури TMP36 та трьома світлодіодами у середовищі Tinkercad, що імітують роботу виконавчих каналів. Проведено перевірку роботи програмної та апаратної частин, показано відповідність поведінки системи поставленим вимогам. Отримані результати можуть бути використані як навчальний стенд для вивчення основ Інтернету речей, принципів керування мікрокліматом та побудови систем «розумного дому».

Ключові слова: Інтернет речей, мікроклімат, розумний дім, емулятор, WPF, C#, Arduino.

ЗМІСТ

ВСТУП

Стрімкий розвиток технологій Інтернету речей (Internet of Things, IoT) призводить до появи все більшої кількості «розумних» пристроїв, здатних збирати дані з навколишнього середовища, обмінюватися ними через мережу та автоматично приймати керуючі рішення. Однією з практичних сфер застосування IoT є системи керування мікрокліматом приміщень, які дозволяють підтримувати комфортні умови для людини та одночасно підвищувати енергоефективність будівель. Такі системи поєднують датчики температури, вологості, освітленості, присутності та виконавчі пристрої, що керують опаленням, охолодженням та освітленням.

Разом з тим, розробка та тестування подібних IoT-рішень безпосередньо на фізичному обладнанні вимагає певних фінансових витрат, часу на монтаж апаратури та усунення апаратних помилок. Це ускладнює навчання та експерименти з алгоритмами керування, особливо в умовах навчального процесу. Тому актуальним є підхід, за якого логіка роботи системи, моделі сенсорів і виконавчих пристроїв спочатку відпрацьовуються у вигляді програмного емулятора, а апаратна частина розглядається у вигляді спрощеного прототипу.

У даній курсовій роботі розробляється програмний **емулятор системи мікроклімату приміщення** на базі технології WPF та мови C#. Емулятор моделює температуру, вологість, освітленість, віртуальний час доби та присутність людини, а також реалізує автоматичне керування «обігрівачем», «кондиціонером» і «освітленням» у кількох режимах роботи. Для демонстрації можливої апаратної реалізації системи розглядається прототип на базі Arduino Uno з датчиком температури TMP36 та трьома світлодіодами, який збирається та перевіряється у середовищі Tinkercad.

Метою роботи є створення програмного емулятора IoT-системи керування мікрокліматом та демонстраційної апаратної схеми, які разом

можуть використовуватися як навчальний стенд для вивчення основ IoT та принципів побудови систем «розумного дому». Для досягнення мети необхідно проаналізувати предметну область і системи-аналоги, обґрунтувати вибір технічних та програмних засобів, спроєктувати логічну структуру системи, реалізувати програмну частину емулятора, побудувати схему прототипу на Arduino та перевірити працездатність розробленого рішення [1].

Структурно пояснювальна записка містить опис предметної області та існуючих аналогів, перелік використаних технічних засобів, аналіз методів розв’язання задачі, опис логічної структури системи, результати перевірки роботи програмної та апаратної частини, характеристику вхідних та вихідних даних, інструкцію користувача, висновки та додатки з фрагментами коду й схемами.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

У цьому розділі розглядаються основні поняття та особливості предметної області, пов’язаної з керуванням мікрокліматом приміщення на основі технологій Інтернету речей. Також уточнюється роль програмних емуляторів у дослідженні та відлагодженні IoT-систем, що важливо для розуміння місця розробленого емульованого пристрою в загальному контексті.

1.1 Поняття мікроклімату приміщень

Мікроклімат приміщення – це сукупність параметрів внутрішнього середовища, до яких належать температура повітря, відносна вологість, рівень освітленості, а також рух повітря та фактор присутності людей. Від правильного поєднання цих параметрів залежать комфорт перебування у приміщенні, працездатність користувачів, збереження обладнання й будівельних конструкцій, а також обсяг енергоспоживання систем опалення, кондиціонування та освітлення.

Для житлових та офісних приміщень зазвичай задаються діапазони комфортної температури та вологості, а також мінімально допустимі рівні освітленості для роботи чи відпочинку. Вихід параметрів мікроклімату за межі цих діапазонів призводить до дискомфорту, погіршення самопочуття користувачів або нераціонального використання енергоресурсів [1][2][3].

1.2 Використання технологій Інтернету речей у системах мікроклімату

Технології Інтернету речей (IoT) активно застосовуються для автоматизації контролю й керування мікрокліматом. Типова IoT-система

такого класу включає:

- сенсори температури, вологості, освітленості та руху (присутності);
- контролер (мікроконтролер Arduino, плата ESP, мікрокомп’ютер тощо), який збирає дані з датчиків і приймає рішення;
- виконавчі пристрої – обігрівачі, кондиціонери, вентилятори, освітлювальні прилади;
- інтерфейс користувача (панель керування, веб- або десктопний застосунок, мобільний додаток).

На основі показів сенсорів контролер аналізує поточний стан середовища й за заданими правилами вмикає або вимикає відповідні виконавчі пристрої. У більш складних рішеннях використовуються різні сценарії та режими роботи, прив’язані до часу доби, графіка роботи користувачів або тарифів на електроенергію [2].

1.3 Баланс між комфортом та енергоефективністю

Одним із ключових завдань систем мікроклімату є пошук компромісу між комфортом користувача та енергоефективністю будівлі. Жорстке підтримання вузького діапазону температури призводить до збільшення споживання тепла або електроенергії, тоді як надто широкий діапазон може викликати дискомфорт. Тому в сучасних системах часто реалізують кілька режимів:

- комфортний режим – орієнтація на максимальний комфорт користувача;
- економний режим – допустиме ширше відхилення параметрів для зменшення витрат;
- режим відсутності – мінімальне підтримання параметрів, коли людей у приміщенні немає.

У межах даної курсової роботи ці режими моделюються в програмному емуляторі системи мікроклімату [2][3].

1.4 Роль програмних емуляторів у дослідженні IoT-систем

Розробка та тестування IoT-рішень безпосередньо на реальному обладнанні потребує закупівлі датчиків, виконавчих пристроїв, модулів зв’язку та додаткового часу на монтаж і налагодження. Це може бути надмірно витратно в умовах навчального проєкту. Тому важливою складовою предметної області є програмні емулятори, які дозволяють:

- моделювати зміну параметрів мікроклімату (температури, вологості, освітленості, присутності людей) у часі;
- відпрацьовувати алгоритми автоматичного керування без фізичних сенсорів;
- наочно демонструвати роботу системи через графічний інтерфейс;
- паралельно проєктувати спрощені апаратні прототипи (наприклад, на базі Arduino).

У цій роботі програмний емулятор системи мікроклімату виконує роль «цифрового двійника» реального IoT-пристрою, а схема на Arduino з датчиком температури та трьома світлодіодами в середовищі Tinkercad слугує прикладом можливої фізичної реалізації тієї ж логіки й надалі може бути пов’язана з літературними джерелами у переліку посилань.

2 СИСТЕМИ АНАЛОГИ

У даному підрозділі розглянуто приклади реальних IoT-систем та навчальних проєктів, які близькі за ідеєю до розробленого емулятора мікроклімату.

2.1 Комерційні системи керування кліматом

Одним з найвідоміших прикладів є Nest Learning Thermostat від Google. Це «розумний» термостат, який збирає інформацію з датчиків температури, вологості, руху та освітленості, навчається звичкам користувача, автоматично перемикає режими (у тому числі енергоощадний Eco) та може керувати опаленням і гарячою водою через застосунок чи голосових асистентів. Подібні функції – аналіз поведінки користувача, автоматичний вибір режиму, віддалене керування – концептуально співзвучні з алгоритмами, реалізованими в моєму емуляторі [4].

Рисунок 2.1 – Розумний термостат Nest як приклад комерційної IoT-системи керування мікрокліматом

Ще один клас аналогів – інші «розумні» термостати (Tado, Hive, Honeywell тощо), які так само дозволяють регулювати температуру з урахуванням присутності мешканців та часу доби, зменшуючи витрати на опалення на 10–25%.

2.2 Платформи «розумного дому»

Для побудови комплексних систем автоматизації широко використовуються платформи домашньої автоматизації, наприклад Home Assistant. Це відкрите програмне забезпечення для централізованого керування «розумним домом», яке підтримує велику кількість інтеграцій, у тому числі керування освітленням, кліматом, мультимедіа та іншими IoT-пристроями, забезпечуючи сценарії автоматизації та єдину панель керування. На відміну від мого проєкту, Home Assistant орієнтований на реальні пристрої та хмарні/локальні інтеграції, тоді як у курсовому проекту робиться акцент на локальному програмному емуляторі окремого пристрою мікроклімату [5].

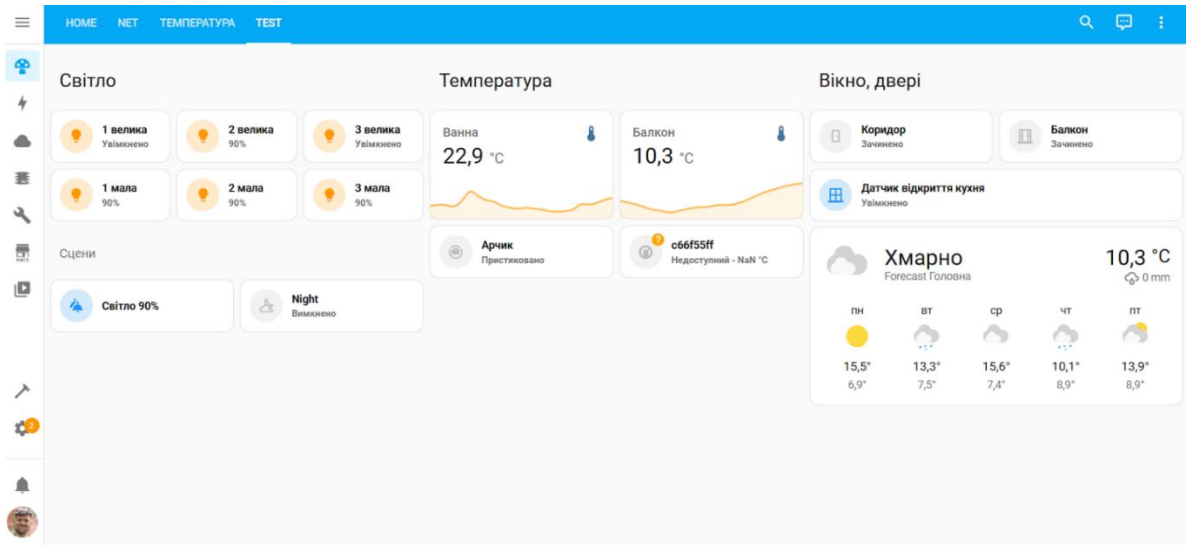

Рисунок 2.2 – Приклад дашборду системи Home Assistant для моніторингу та керування кліматом у «розумному домі»

2.3 Навчальні Arduino-проєкти для моніторингу мікроклімату

В освітніх цілях часто використовують прості Arduino-проєкти з датчиками температури та вологості (DHT11, DHT22) або аналоговим датчиком TMP36. Типовий приклад – проєкт «Temperature and humidity sensor with Arduino», де Arduino UNO опитує датчик DHT11 і відображає температуру та вологість, що може використовуватися як основа для домашньої системи контролю середовища [6]. Інші приклади демонструють підключення TMP36 та виведення температури на LCD-дисплей або роботу з TMP36 і світлодіодами в Tinkercad.

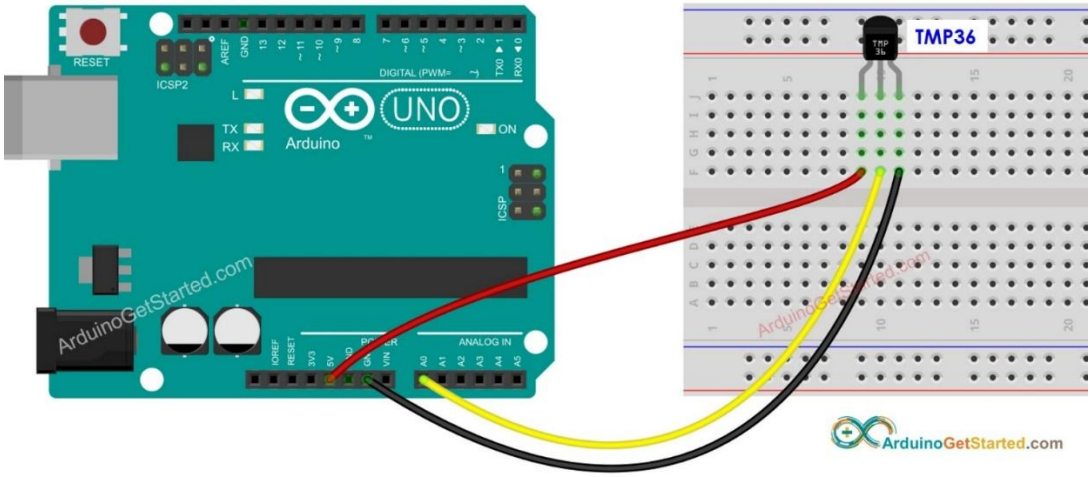

Рисунок 2.3 – Приклад навчального Arduino-проєкту з датчиком температури TMP36

Розроблений у цій роботі Arduino-прототип (TMP36 + 3 світлодіоди, що імітують виконавчі канали) логічно продовжує такі навчальні приклади, але пов’язаний із більш складним десктопним емулятором, який моделює всю систему мікроклімату, включаючи режими роботи, віртуальний час доби та вплив «зовнішніх» умов [7][8][9].

2.4 Порівняння з розробленою системою

Порівнюючи наведені аналоги з результатами курсової роботи, можна відзначити, що:

- комерційні термостати (Nest та інші) реалізують подібну ідею автоматичного керування кліматом, але є закритими продуктами, орієнтованими на кінцевого користувача, а не на навчання та експерименти;
- Home Assistant та подібні платформи вирішують задачу на рівні всього «розумного дому», тоді як у роботі моделюється окремий пристрій мікроклімату з прозорою внутрішньою логікою;
- навчальні Arduino-проєкти зазвичай обмежуються зчитуванням параметрів середовища та простими реакціями (увімкнення LED або сигналізації), тоді як розроблений емулятор поєднує модель середовища, алгоритми керування в різних режимах, графічний інтерфейс і апаратний прототип [4-7].

Таким чином, розроблена система займає проміжну нішу між простими лабораторними макетами та промисловими/комерційними рішеннями і може використовуватись як навчальний інструмент для вивчення принципів IoT та керування мікрокліматом.

2.5 Переваги та недоліки розглянутих систем-аналогів

Комерційні розумні термостати (на прикладі Nest Learning Thermostat).

Переваги:

- підтримка автоматичного навчання звичкам користувача та адаптація графіка опалення;
- наявність мобільного застосунку та віддаленого доступу через Інтернет;

- інтеграція з іншими сервісами Google та голосовими асистентами [4].

Недоліки:

- закритість платформи та обмежена можливість змінювати внутрішні алгоритми;
- висока вартість у порівнянні з навчальними або DIY-рішеннями;
- орієнтація на типові сценарії, що ускладнює використання в навчальних цілях як гнучкого стенду.

Платформи домашньої автоматизації (на прикладі Home Assistant).

Переваги:

- підтримка великої кількості протоколів та пристроїв, можливість об’єднати різні підсистеми «розумного дому» в єдиному інтерфейсі;
- потужні засоби автоматизації (scripts, automations, сцени), відкритий код та активна спільнота [5];
- можливість розгортання як повноцінної хаб-системи для житла чи офісу.

Недоліки:

- порівняно складне налаштування для початківців, потреба в окремому сервері або одноплатному комп’ютері;
- надлишковий функціонал для невеликого навчального проєкту з однією підсистемою мікроклімату;
- необхідність підтримувати оновлення та резервне копіювання конфігурацій.

Навчальні Arduino-проєкти контролю мікроклімату.

Переваги:

- простота схем на базі Arduino Uno, сенсорів DHT11/DHT22 або TMP36, доступність великої кількості прикладів і бібліотек [6–9];
- наочність роботи реальних сенсорів і виконавчих пристроїв (світлодіоди, реле, вентилятори);
- невисока вартість компонентів та придатність для лабораторних і курсових робіт.

Недоліки:

- обмежений графічний інтерфейс (часто лише послідовний порт або невеликі LCD-дисплеї);
- складність масштабування проєкту до повноцінної системи з багатим інтерфейсом користувача;
- відсутність єдиної програмної моделі «віртуального середовища», тому логіка керування частіше прив’язана до конкретного набору датчиків.

У порівнянні з розглянутими аналогами, розроблений у даному курсовому проекті емулятор займає проміжну позицію: він не є комерційним продуктом, але має зручний десктопний інтерфейс та модель віртуального середовища, якого зазвичай бракує простим Arduino-проєктам. Одночасно наявність демонстраційного прототипу на Arduino дозволяє зберегти зв’язок із реальною апаратною реалізацією системи мікроклімату.

3 ТЕХНІЧНІ ЗАСОБИ

У цьому розділі наведено перелік основних програмних та апаратних засобів, що були використані під час розробки емулятора системи мікроклімату та демонстраційного апаратного прототипу, а також коротко обґрунтовано вибір мови програмування й технологій.

3.1 Програмні засоби

Для реалізації програмної частини курсової роботи використовувалися такі засоби:

- Операційна система: Windows 10/11.
- Платформа розробки: .NET 8.
- Мова програмування: C#.
- Технологія побудови інтерфейсу: WPF (Windows Presentation Foundation) для створення десктопного застосунку з графічним інтерфейсом [10][11].
- Середовище розробки: JetBrains Rider (редагування коду, збірка, налагодження).
- Система контролю версій: Git для збереження проміжних версій проєкту.
- Онлайн-симулятор електроніки: Tinkercad Circuits для моделювання роботи Arduino-схеми без фізичного обладнання.

Цей набір дозволяє повністю розробити та протестувати програмний емулятор на ПК, а також перевірити коректність зібраної схеми в браузері. Загальний вигляд програмного проєкту у середовищі розробки наведено на рис. 3.1.

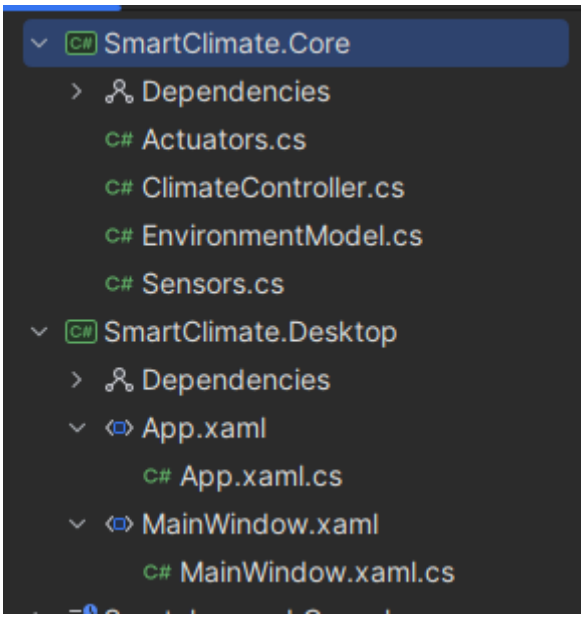

Рисунок 3.1 – Структура проєкту емулятора системи мікроклімату у середовищі JetBrains Rider

3.2 Апаратні засоби

Для демонстрації можливої апаратної реалізації IoT-пристрою мікроклімату використано простий стенд на базі [7][8][12]:

- Arduino Uno R3 – мікроконтролерна плата, що може виступати «мозком» IoT-пристрою;
- аналоговий датчик температури TMP36 – формує сигнал, пропорційний температурі [9];
- три світлодіоди з резисторами – імітують стан виконавчих пристроїв (обігрівач, кондиціонер, освітлення);
- макетна плата (breadboard) та набір з’єднувальних проводів – для швидкого монтажу без пайки.

У реальних умовах замість світлодіодів могли б використовуватися релейні модулі або драйвери для керування нагрівачами, вентиляторами та освітлювальними приладами, однак для навчального проєкту достатньо

індикації стану через LED. Схему апаратного прототипу, змодельовану в середовищі Tinkercad, наведено на рис. 3.2 та у додатку Б.

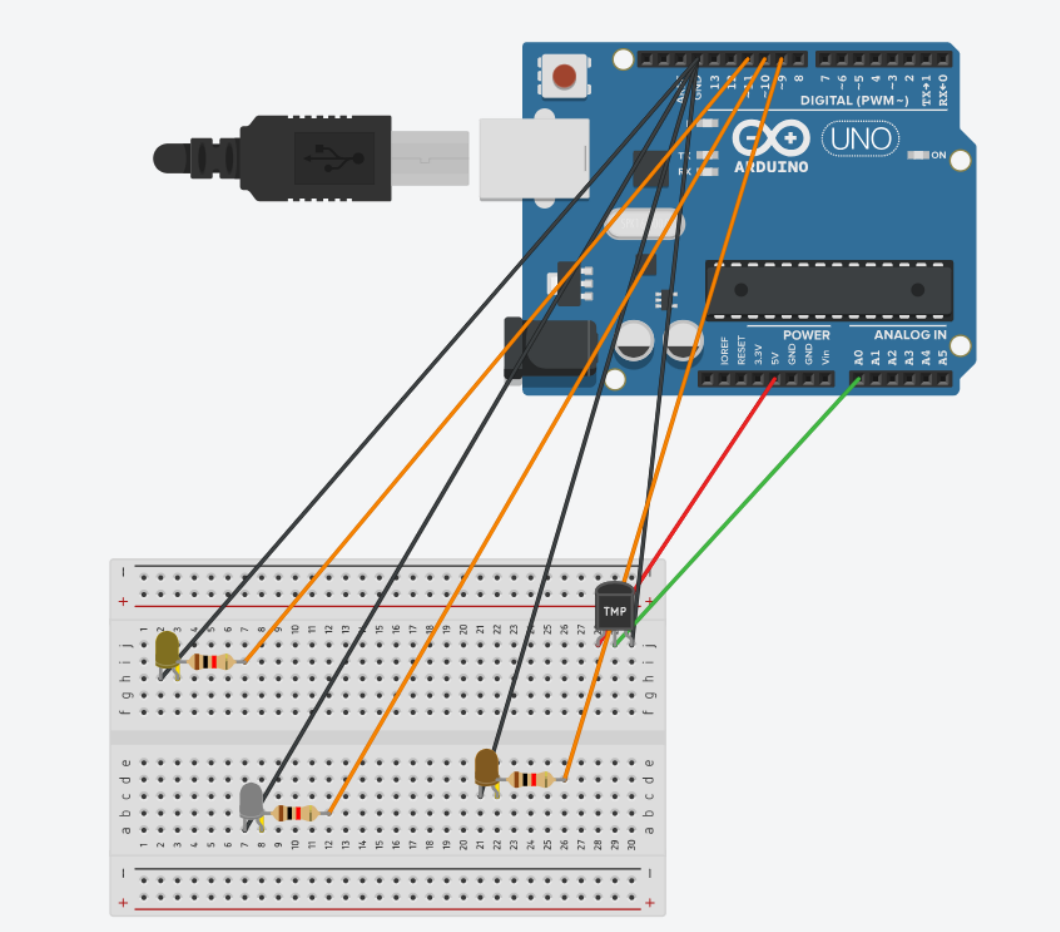

Рисунок 3.2 – Схема апаратного прототипу IoT-пристрою мікроклімату на Arduino Uno в середовищі Tinkercad

3.3. Обґрунтування вибору мови програмування та технологій

Для розробки програмного емулятора обрано мову C# та платформу .NET 8 з кількох причин:

- C# добре підходить для побудови десктопних застосунків з графічним інтерфейсом і має розвинену екосистему бібліотек;
- технологія WPF дозволяє зручно відокремити логіку застосунку від інтерфейсу, підтримує прив’язку даних (data binding), що спрощує відображення стану сенсорів та виконавчих пристроїв;

- .NET забезпечує хорошу продуктивність, кросверсійність і можливість подальшого розширення проєкту (наприклад, додавання Web API або інтеграції з хмарними сервісами).

В якості апаратної платформи для прототипу вибрано Arduino Uno, оскільки це одна з найпоширеніших і найпростіших для навчання мікроконтролерних плат із великою кількістю готових прикладів підключення датчиків та виконавців. Використання Tinkercad Circuits дозволило змоделювати роботу Arduino-схеми навіть без наявності реального обладнання, що зручно в освітньому контексті.

4 АНАЛІЗ МЕТОДІВ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

У даному розділі розглядаються можливі підходи до реалізації системи керування мікрокліматом на основі технологій Інтернету речей, а також обґрунтовується вибір середовища розробки та загальної архітектури рішення.

4.1. Вибір середовища розробки

Для програмної реалізації системи мікроклімату можливими були кілька варіантів:

- веб-застосунок (інтерфейс на HTML/CSS/JavaScript, логіка на Node.js або іншому серверному середовищі);
- настільний застосунок на Java (Swing/JavaFX);
- настільний застосунок на Python (Tkinter, PyQt тощо);
- настільний застосунок на .NET (Windows Forms, WPF, MAUI);

Веб-підхід дає переваги кросплатформенності та віддаленого доступу, але потребує побудови клієнт–серверної архітектури, налаштування веб-сервера та окремого сховища даних, що ускладнює навчальний проєкт. Рішення на Java або Python також є можливими, проте вимагають додаткових бібліотек для побудови сучасного інтерфейсу й не інтегруються настільки природно у вже наявну екосистему .NET.

У курсовому проекту як основне середовище розробки обрано платформу .NET 8, мову C# та технологію WPF. Такий вибір обумовлений наступним:

- WPF забезпечує зручне розділення логіки та інтерфейсу за допомогою XAML-розмітки та механізмів прив’язки даних (data binding), що спрощує відображення показів сенсорів і станів виконавчих пристроїв у реальному часі;

- C# є сучасною, строго типізованою мовою з розвиненою екосистемою та великою кількістю навчальних матеріалів;
- .NET 8 забезпечує високу продуктивність, підтримку сучасних можливостей платформи та потенціал подальшого розширення проєкту (додавання Web API, інтеграція з хмарними IoT-сервісами тощо);
- використання середовища JetBrains Rider дозволяє ефективно виконувати редагування, збірку та налагодження застосунку.

Таким чином, обране середовище розробки є компромісом між зручністю реалізації, можливостями графічного інтерфейсу та перспективою подальшого розвитку системи.

4.2. Підходи до моделювання IoT-системи мікроклімату

Для вирішення поставленої задачі можна було розглядати кілька загальних підходів:

1. Повністю апаратна реалізація.

Усі вимірювання виконуються реальними сенсорами (температури, вологості, освітленості, руху), логіка керування реалізується у мікроконтролері (Arduino, ESP32 тощо), а інтерфейс користувача обмежується світлодіодами, кнопками або простим дисплеєм. Перевагою є наближеність до реального пристрою, недоліком – більша складність налагодження та залежність від фізичної апаратури.

2. Хмарно-орієнтована IoT-система.

Сенсори та виконавчі пристрої підключаються до мережі й передають дані в хмарний сервіс (MQTT-брокер, REST API, платформи типу Azure IoT, ThingsBoard тощо). Користувач працює через веб-інтерфейс або мобільний застосунок. Такий підхід максимально наближений до промислових рішень, але істотно перевищує за складністю вимоги навчального проєкту [1][3].

3. Програмний емулятор з апаратним прототипом.

Основна логіка, модель середовища та моделювання сенсорів реалізуються у вигляді настільного застосунку, а апаратна частина представлена спрощеним прототипом (наприклад, Arduino + один датчик + LED як індикатори). Перевагою є можливість відлагодити алгоритми керування в зручному графічному інтерфейсі та одночасно продемонструвати зв’язок із реальною апаратурою.

У межах курсової роботи обрано саме третій підхід як найбільш збалансований за рівнем складності, наочністю та вимогами до обладнання.

4.3. Обґрунтування обраного методу реалізації

Обраний метод поєднання програмного емулятора та демонстраційного апаратного прототипу на Arduino має такі переваги для поставленої задачі:

- дає змогу детально промоделювати мікроклімат приміщення (температура, вологість, освітленість, віртуальний час доби, присутність людини) без потреби мати повний набір фізичних сенсорів;
- дозволяє реалізувати і наочно показати різні режими роботи контролера (Comfort, Eco, Away) та їх вплив на стан виконавчих пристроїв;
- забезпечує зручний графічний інтерфейс для моніторингу параметрів та журналу подій, що важливо для демонстрації і під час захисту роботи;
- демонстраційна схема на Arduino Uno з датчиком температури TMP36 та трьома світлодіодами показує, як програмна логіка може бути перенесена на реальне «залізо», не ускладнюючи проєкт великою кількістю компонентів;

- використання онлайн-симулятора Tinkercad Circuits дозволяє перевірити працездатність апаратної частини без обов'язкової наявності фізичної плати, що зручно в умовах навчання.

Таким чином, аналіз можливих підходів показав, що для навчальної курсової роботи найбільш доцільним є комбінований варіант: програмний емулятор на базі .NET/WPF та спрощений апаратний прототип на Arduino, які разом забезпечують як практичну, так і демонстраційну цінність проєкту.

4.4. Технічний, економічний та соціальний ефект від впровадження системи

Розроблена система може забезпечити низку позитивних ефектів у разі її подальшого розвитку та практичного використання.

Технічний ефект.

Застосування емулятора дозволяє відпрацьовувати алгоритми керування мікрокліматом, не маючи повного комплекту реальних сенсорів та виконавчих пристроїв. Це зменшує ризик помилок при подальшому перенесенні логіки на апаратну платформу та спрощує налагодження системи. Крім того, виділення окремих модулів (модель середовища, сенсори, виконавці, контролер, інтерфейс) полегшує розширення функціоналу та інтеграцію з іншими компонентами «розумного дому».

Економічний ефект.

Розумні системи керування мікрокліматом дозволяють зменшити витрати на опалення, кондиціонування та освітлення завдяки точнішому підтриманню параметрів середовища та використанню енергоощадних режимів (наприклад, Eco, режим відсутності) [2], [3], [4]. Навіть у навчальному варіанті відпрацювання таких алгоритмів на емуляторі дає можливість оцінити потенційні сценарії економії енергії та надалі адаптувати їх під реальні системи.

Окремо економічною перевагою є можливість проводити лабораторні та дослідні роботи без закупівлі великої кількості датчиків і контролерів: достатньо одного демонстраційного стенду та програмного емулятора.

Соціальний ефект.

Система спрямована на підвищення комфорту перебування людей у приміщенні за рахунок автоматичного підтримання прийнятних параметрів мікроклімату та адаптації до присутності/відсутності мешканців. Це позитивно впливає на самопочуття, працездатність і безпеку користувачів. З освітньої точки зору розроблений комплекс може використовуватися як навчальний стенд для студентів, які вивчають Інтернет речей, мікроконтролери та засоби розробки під .NET. Використання сучасних інструментів програмування та моделювання сприяє формуванню в студентів практичних навичок проєктування IoT-систем, що, у свою чергу, підвищує їхню конкурентоспроможність на ринку праці.

5 ОПИС ЛОГІЧНОЇ СТРУКТУИ

Метою даної курсової роботи є створення емулятора IoT-системи мікроклімату приміщення з віртуальними датчиками та виконавчими пристроями, а також демонстраційного апаратного прототипу на базі Arduino. Логічна структура системи побудована таким чином, щоб максимально наблизити програмну модель до реальної IoT-пристрою: виділено окремі рівні для моделі середовища, сенсорів, виконавчих пристроїв, контролера та інтерфейсу користувача. Структурну схему системи наведено на рис. 5.1.

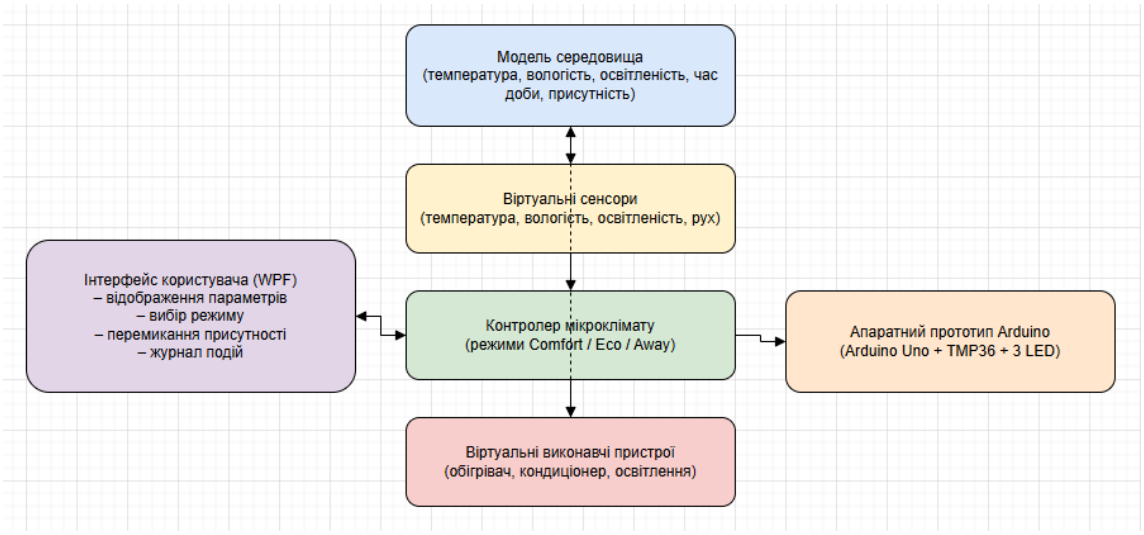

Рисунок 5.1 – Структурна схема системи емуляції мікроклімату приміщення

5.1 Загальна архітектура системи

На логічному рівні система складається з таких основних блоків:

- модель середовища – відповідає за збереження та оновлення параметрів мікроклімату (температура, вологість, освітленість, віртуальний час доби, присутність людини);
- сенсори – набір класів, які зчитують значення з моделі середовища

і надають їх контролеру в уніфікованому вигляді (датчик температури, вологості, освітленості, руху);

- виконавчі пристрої – програмні аналоги обігрівача, кондиціонера та освітлення, які впливають на модель середовища (нагрівають, охолоджують, змінюють освітленість) та зберігають свій поточний стан;
- контролер мікроклімату – модуль, що отримує дані із сенсорів і, залежно від вибраного режиму роботи, приймає рішення про ввімкнення або вимкнення виконавчих пристроїв;
- інтерфейс користувача – WPF-застосунок, який відображає параметри середовища, стани виконавців, дозволяє змінювати режим роботи та імітувати присутність людини, а також містить журнал подій.

У якості апаратного прототипу використовуються плата Arduino Uno з датчиком температури TMP36 та трьома світлодіодами. Вони логічно відповідають одному з програмних сенсорів (температура) та трьом виконавчим каналам (обігрівач, кондиціонер, освітлення), що демонструє можливість перенесення алгоритмів, відпрацьованих у емуляторі, на реальне апаратне забезпечення.

5.2 Структура програмного забезпечення

Програмна частина системи реалізована у вигляді двох основних проєктів: бібліотеки ядра (логіка моделювання) та десктопного застосунку (інтерфейс користувача).

До ядра входять такі ключові класи:

- EnvironmentModel – зберігає поточні значення температури, вологості, освітленості, віртуального часу доби та стан

присутності, а також реалізує метод оновлення середовища з урахуванням впливу «вуличних» умов і станів виконавців;

- класи сенсорів (TemperatureSensor, HumiditySensor, LightSensor, MotionSensor) – інкапсулюють доступ до моделі середовища та надають контролеру уніфікований інтерфейс читання даних;
- класи виконавців (HeaterActuator, CoolerActuator, LampActuator) – змінюють модель середовища відповідно до свого стану (увімкнено/вимкнено) та використовуються контролером як об’єкти керування;
- ClimateController – реалізує алгоритм керування мікрокліматом у декількох режимах (Comfort, Eco, Away), порівнює покази сенсорів із пороговими значеннями та приймає рішення про стан кожного виконавця.

Десктопний застосунок на WPF відповідає за створення головного вікна, прив’язку даних (data binding) до властивостей моделі та контролера, обробку подій користувача (зміна режиму, перемикання присутності) і відображення журналу подій. Основні фрагменти коду, що реалізують логіку ядра та інтерфейсу, наведені у додатку В.

5.3 Інтерфейс користувача

Інтерфейс користувача реалізовано у вигляді одного головного вікна WPF-застосунку, поділеного на кілька логічних блоків. У верхній частині відображається назва системи, короткий опис призначення та кнопка для імітації присутності людини в приміщенні.

Ліва частина вікна містить блок «Параметри середовища», де в текстовому вигляді показуються поточні значення температури, вологості, освітленості та стан присутності. Права частина вікна містить блок «Стан

виконавчих пристроїв» із випадаючим списком для вибору режиму роботи контролера, текстовим відображенням станів обігрівача, кондиціонера та освітлення, а також журналом подій, у який записуються всі зміни режимів і станів виконавців із часовими мітками. Загальний вигляд головного вікна програмного емулятора наведено на рис. 5.2.

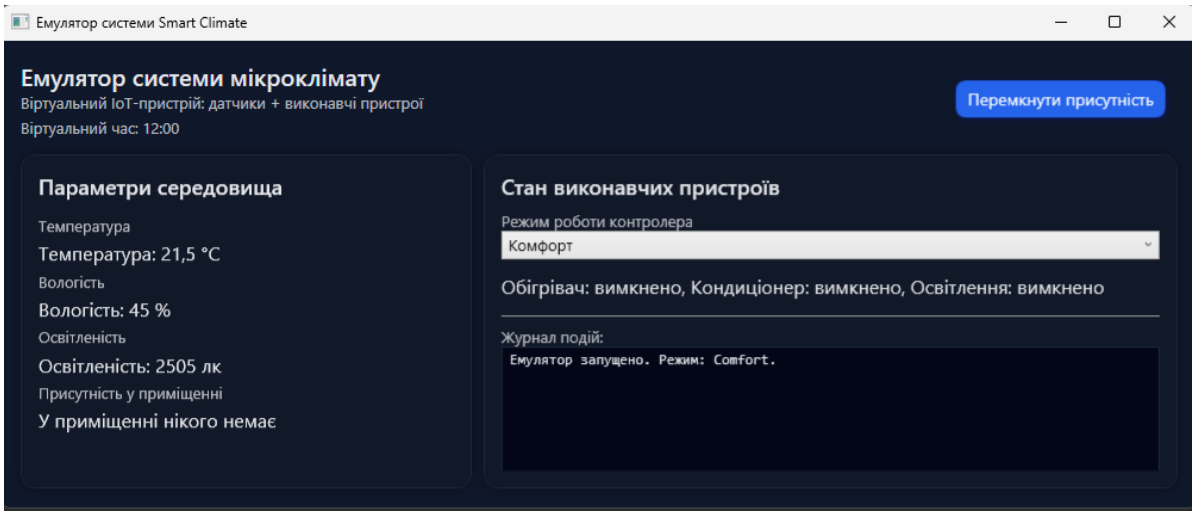

Рисунок 5.2 – Головне вікно програмного емулятора системи мікроклімату приміщення

Такий поділ інтерфейсу дозволяє користувачу одночасно спостерігати за динамікою параметрів мікроклімату, змінювати налаштування системи та контролювати, як саме реагують виконавчі пристрої на зміну умов та режимів роботи.

6 ПЕРЕВІРКА РОБОТИ СИСТЕМИ

У цьому розділі наведено результати перевірки роботи програмного емулятора системи мікроклімату та демонстраційного апаратного прототипу на базі Arduino. Метою перевірки є підтвердження коректності реалізації основних функцій: моделювання параметрів середовища, автоматичного керування виконавчими пристроями та відображення станів у інтерфейсі користувача.

6.1 Перевірка роботи програмного емулятора

Перевірка програмної частини виконувалася шляхом запуску WPF-застосунку «Емулятор системи мікроклімату» та послідовного відпрацювання кількох типових сценаріїв роботи.

На першому етапі було перевірено коректність відображення параметрів середовища. Після запуску застосунку в блоці «Параметри середовища» відображаються початкові значення температури, вологості, освітленості та стану присутності. У процесі роботи моделі значення параметрів змінюються в часі відповідно до логіки, закладеної у класі EnvironmentModel (імітація добового циклу, вплив виконавців). Було підтверджено, що всі відображувані значення оновлюються коректно та узгоджено між собою.

Далі перевірявся алгоритм керування виконавчими пристроями в різних режимах роботи контролера. Для кожного з режимів Comfort, Eco та Away задавався певний інтервал комфортної температури, після чого спостерігалося, як контролер вмикає й вимикає обігрівач та кондиціонер залежно від поточної температури. У режимі Comfort обігрівач вмикався раніше і вимикався пізніше, забезпечуючи вузький діапазон температури, тоді як у режимі Eco допускалися більші відхилення. Візуально стани виконавчих

пристроїв відображалися у відповідному блоці інтерфейсу, що дозволило підтвердити коректність логіки прийняття рішень.

Окремо був перевірений вплив присутності людини на роботу освітлення. При натисканні кнопки «Перемкнути присутність» у верхній панелі інтерфейсу змінювався стан змінної присутності в моделі середовища. У випадку, коли віртуальний час доби відповідав нічному або темному періоду, поява «присутності» призводила до автоматичного ввімкнення освітлення, а відсутність – до його вимкнення. Це підтверджує, що логіка керування освітленням коректно враховує як час доби, так і стан присутності.

Для кожної зміни режиму роботи контролера та станів виконавчих пристроїв у журналі подій створювався відповідний запис із часовою міткою. Перегляд журналу дозволив переконатися, що всі важливі події фіксуються, а послідовність записів відповідає фактичному перебігу роботи системи. Приклад роботи емулятора із відображенням станів і журналу подій наведено на рис. 6.1.

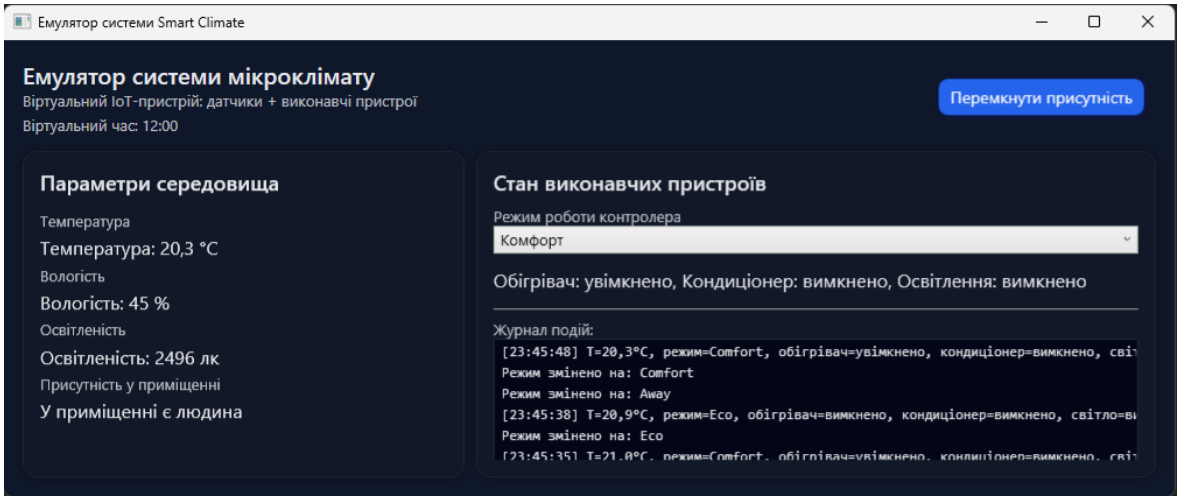

Рисунок 6.1 – Приклад роботи програмного емулятора системи мікроклімату

6.2 Перевірка роботи програмного емулятора

Перевірка апаратної частини здійснювалася у середовищі Tinkercad Circuits, де була зібрана схема на базі Arduino Uno, датчика температури TMP36 та трьох світлодіодів з резисторами, що імітують виконавчі пристрої (обігрівач, кондиціонер і освітлення).

Для тестування було завантажено простий скетч, який [7][9][12]:

- періодично зчитує значення з аналогового входу A0, до якого підключений TMP36;
- обчислює температуру за напругою датчика;
- виводить отримане значення температури у Serial Monitor;
- по черзі вмикає кожен зі світлодіодів, підключених до цифрових виходів D9, D10 та D11.

Під час запуску симуляції в Tinkercad було підтверджено:

- при зміні температури повзунком у властивостях TMP36 змінюється показ температури в Serial Monitor, що свідчить про коректність підключення датчика та роботи аналогового входу Arduino;
- кожен зі світлодіодів (Heater, AC, Lamp) послідовно вмикається й вимикається під керуванням програми, що підтверджує правильність з’єднання цифрових виходів із резисторами та LED.

Таким чином, апаратний прототип коректно реагує на зміну вхідного сигналу (температури) та здатен керувати виконавчими каналами, що логічно відповідає функціям, реалізованим у програмному емуляторі. Загальний вигляд схеми та вікна симуляції в Tinkercad наведено на рис. 6.2.

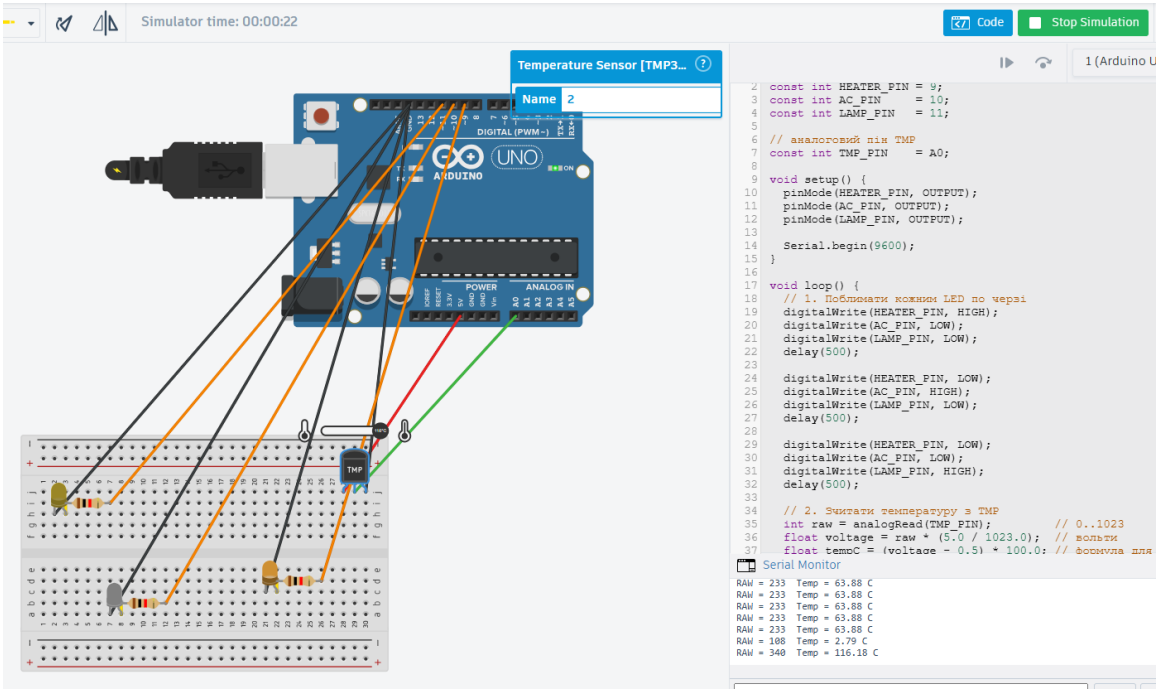

Рисунок 6.2 – Перевірка роботи апаратного прототипу в середовищі Tinkercad

Підсумовуючи, обидві частини системи – програмний емулятор та апаратний прототип – продемонстрували коректну роботу, а поведінка системи відповідає поставленим вимогам до функціоналу.

7 ВХІДНІ ТА ВИХІДНІ ДАНІ

У даному розділі описано основні вхідні та вихідні дані, які використовуються в програмному емуляторі системи мікроклімату та в демонстраційному апаратному прототипі на базі Arduino.

7.1 Вхідні дані

До вхідних даних системи належать як користувацькі налаштування, так і модельовані параметри середовища.

Основні вхідні дані програмного емулятора:

- Режим роботи системи
- Обирається користувачем із випадаючого списку (Comfort, Eco, Away). Від режиму залежать порогові значення температури та поведінка контролера.
- Стан присутності людини
- Імітується натисканням кнопки «Перемкнути присутність». Впливає насамперед на роботу освітлення (вмикання/вимикання світла при певних умовах освітленості та часу доби).
- Параметри зовнішнього середовища (модельовані)
- У моделі EnvironmentModel враховується віртуальний час доби та умовна «зовнішня» температура/освітленість, які впливають на зміну внутрішніх параметрів. Ці значення задаються алгоритмічно (емулюються), а не вводяться користувачем вручну.
- Поточний стан виконавчих пристроїв
- При кожному кроці моделювання контролер враховує попередній стан обігрівача, кондиціонера та освітлення, що також виступає вхідними даними для оновлення моделі середовища (наприклад,

якщо обігрівач був увімкнений, температура зростає швидше).

Для апаратного прототипу на Arduino основними вхідними даними є:

- Аналоговий сигнал із датчика температури TMP36, який перетворюється в значення температури в коді Arduino;
- Програма (скетч), прошита в контролер, що визначає порядок опитування датчика та керування світлодіодами.

7.2 Вихідні дані

Вихідні дані системи – це результати роботи алгоритмів моделювання та керування, які відображаються користувачу або подаються на виконавчі пристрої.

Основні вихідні дані програмного емулятора:

- Поточні значення параметрів мікроклімату
  - температура повітря;
  - відносна вологість;
  - рівень освітленості;
  - віртуальний час доби;
  - стан присутності людини.
  - Ці дані відображаються у відповідному блоці інтерфейсу у текстовому вигляді та постійно оновлюються в процесі моделювання.

- Стани виконавчих пристроїв
  - «Обігрівач увімкнено/вимкнено»;
  - «Кондиціонер увімкнено/вимкнено»;
  - «Освітлення увімкнено/вимкнено».
  - Вони показуються у блоці «Стан виконавчих пристроїв» і залежать від поточних показів сенсорів і вибраного режиму.
- Журнал подій
  - Список текстових записів, у яких фіксуються:
  - зміни режимів роботи контролера;
  - вмикання та вимикання виконавчих пристроїв;
  - інші важливі події (наприклад, зміна стану присутності).
  - Журнал дозволяє простежити динаміку роботи системи в часі.

Для апаратного прототипу вихідними даними є:

- Стан світлодіодів, підключених до пінів D9, D10, D11, які імітують вмикання обігрівача, кондиціонера та освітлення;
- Вивід значення температури у Serial Monitor у середовищі Tinkercad або Arduino IDE, що підтверджує коректність перетворення аналогового сигналу датчика TMP36 в цифрове значення.

Таким чином, вхідні та вихідні дані системи охоплюють як модельовані параметри мікроклімату та налаштування користувача, так і стани виконавчих пристроїв, що відображаються у графічному інтерфейсі та на демонстраційному апаратному прототипі.

8 ІНСТРУКЦІЯ КОРИСТУВАЧА

У цьому розділі наведено коротку інструкцію з використання програмного емулятора системи мікроклімату та демонстраційного апаратного прототипу на базі Arduino.

8.1. Вимоги та запуск програмного емулятора

Для роботи програмного емулятора необхідно:

- операційна система: Windows 10/11;
- встановлене середовище виконання .NET 8 (за потреби);
- наявність зібраного виконуваного файлу застосунку (наприклад, SmartClimate.Desktop.exe).

Порядок запуску:

1. Відкрити каталог, у якому розміщено виконуваний файл застосунку.
2. Запустити файл подвійним кліком.
3. Дочекатися завантаження головного вікна емулятора системи мікроклімату.

8.2. Робота з головним вікном емулятора

Головне вікно застосунку складається з кількох функціональних областей:

- верхня панель – назва системи та кнопка «Перемкнути присутність», яка імітує появу або зникнення людини в приміщенні;

- ліва частина – блок «Параметри середовища», де відображаються:
  - поточна температура повітря;
  - відносна вологість;
  - рівень освітленості;
  - віртуальний час доби;
  - стан присутності (є/немає);
- права частина – блок «Стан виконавчих пристроїв», який містить:
  - випадаючий список для вибору режиму роботи контролера (Comfort, Eco, Away);
  - текстові індикатори станів обігрівача, кондиціонера та освітлення;
  - журнал подій, де фіксуються зміни режимів та станів виконавчих пристроїв.

Користувач може:

1. Обрати режим роботи системи в комбобоксі. Після зміни режиму контролер почне по-іншому реагувати на зміну температури (звуження/розширення діапазону комфортної температури).
2. Натискати кнопку «Перемкнути присутність», імітуючи прихід чи вихід людини з приміщення. При певних умовах (темний час доби) це призводить до автоматичного вмикання або вимикання освітлення.
3. Спостерігати за зміною значень температури, вологості та освітленості в блоці «Параметри середовища» у процесі моделювання.
4. Аналізувати роботу системи за допомогою журналу подій, де відображається історія роботи контролера.

8.3. Типові сценарії використання емулятора

Рекомендується перевіряти роботу системи на таких сценаріях:

- Зміна режиму роботи контролера
  - Увімкнути режим Comfort та спостерігати, як часто вмикається обігрівач/кондиціонер.
  - Перемкнути режим на Eco та порівняти частоту реакцій системи.
  - У режимі Away переконатися, що система підтримує більш економний режим роботи.

- Імітація присутності людини
  - У темний (віртуально нічний) період натиснути кнопку «Перемкнути присутність».
  - Переконатися, що освітлення вмикається при появі «людини» та вимикається при її відсутності.
- Аналіз журналу подій
  - Змінювати режими, імітувати присутність і спостерігати, як це відображається в журналі подій.
  - Переконатися, що час та послідовність подій відповідають реальним діям користувача і роботі контролера.

8.4. Використання демонстраційного апаратного прототипу

Демонстраційний прототип реалізовано в середовищі Tinkercad Circuits. Схема містить плату Arduino Uno, датчик температури TMP36 та три світлодіоди з резисторами.

1. Порядок роботи з прототипом у Tinkercad:
2. Відкрити проєкт із зібраною схемою в Tinkercad.
3. Перевірити, що до Arduino завантажено тестовий скетч, який:
    - зчитує значення з датчика TMP36;
    - виводить обчислену температуру у Serial Monitor;
    - вмикає/вимикає світлодіоди на пінових D9, D10 та D11.
4. Натиснути кнопку Start Simulation.
5. Відкрити Serial Monitor та спостерігати за показами температури.
6. Змінювати температуру повзунком у властивостях TMP36 і перевіряти зміну значень у Serial Monitor.
7. Спостерігати за роботою світлодіодів, які імітують вмикання обігрівача, кондиціонера та освітлення.

Таким чином, користувач може ознайомитися як із програмною реалізацією системи мікроклімату у вигляді WPF-емулятора, так і з її можливою апаратною реалізацією на основі Arduino, що підвищує наочність та навчальну цінність розробленої системи.

ВИСНОВКИ

У курсовому проекті було розроблено та досліджено емулятор IoT-системи керування мікрокліматом приміщення, а також створено демонстраційний апаратний прототип на базі Arduino. Метою проєкту було змоделювати роботу «розумного» пристрою, який на основі показів сенсорів (температури, вологості, освітленості, присутності людини) автоматично керує виконавчими пристроями (обігрівачем, кондиціонером, освітленням) у різних режимах роботи. Поставлена мета була досягнута в повному обсязі.

У теоретичній частині було проаналізовано предметну область керування мікрокліматом приміщень, розглянуто роль технологій Інтернету речей у системах «розумного дому», а також наведено приклади існуючих систем-аналогів – комерційних розумних термостатів, платформ домашньої автоматизації та навчальних Arduino-проєктів. Окрему увагу приділено питанню балансу між комфортом користувача та енергоефективністю, що безпосередньо впливає на вибір режимів роботи системи мікроклімату.

У практичній частині було обґрунтовано вибір платформи .NET 8, мови C# та технології WPF для реалізації програмного емулятора. Запропоновано логічну структуру системи, яка включає модель середовища, віртуальні сенсори, виконавчі пристрої, контролер мікроклімату та графічний інтерфейс користувача. Реалізовано кілька режимів роботи контролера (Comfort, Eco, Away), що відрізняються параметрами комфортної температури та поведінкою виконавчих пристроїв. Інтерфейс користувача дозволяє спостерігати за поточними параметрами мікроклімату, перемикати режими роботи, імітувати присутність людини та переглядати журнал подій.

Розроблена система має відчутний технічний, економічний та соціальний ефект. Технічний ефект полягає у можливості відпрацьовувати алгоритми керування мікрокліматом на програмному емуляторі без потреби в повному комплекті апаратури, що спрощує налагодження і подальше перенесення логіки на реальні пристрої. Економічний ефект пов’язаний з

потенційною економією енергоресурсів завдяки використанню енергоощадних режимів роботи (Eco, режим відсутності), а також зі зменшенням витрат на навчальне обладнання за рахунок комбінування одного демонстраційного стенду з програмною моделлю. Соціальний ефект проявляється у підвищенні комфорту та безпеки перебування людей у приміщенні, а також у зростанні якості підготовки студентів, які отримують практичні навички проєктування та дослідження IoT-систем.

Для демонстрації можливої апаратної реалізації було спроєктовано спрощений прототип на Arduino Uno з використанням аналогового датчика температури TMP36 та трьох світлодіодів, які імітують роботу обігрівача, кондиціонера та освітлення. Схему зібрано та перевірено в середовищі Tinkercad Circuits, що дозволило підтвердити коректність підключення компонентів і працездатність базового алгоритму керування виконавчими каналами.

Проведена перевірка роботи системи показала, що програмний емулятор коректно моделює зміну параметрів мікроклімату та реакцію виконавчих пристроїв на зміну умов і режимів, а апаратний прототип відтворює базову логіку на рівні мікроконтролера. Це дає підстави стверджувати, що розроблений комплекс (емулятор + прототип) може використовуватися як навчальний стенд для ознайомлення з принципами побудови IoT-систем і структурою систем «розумного дому».

Подальший розвиток проєкту може включати розширення набору реальних сенсорів та виконавчих пристроїв, організацію обміну даними між емулятором і Arduino в реальному часі, інтеграцію з веб-інтерфейсом або мобільним застосунком, а також підключення до хмарних IoT-платформ для віддаленого моніторингу й керування. Це дозволить наблизити розроблену систему до повноцінного практичного рішення в області «розумного дому» та енергоефективних будівель.

Розкриття факту делегування завдань генеративному ШІ

Автор заявляє про використання генеративного ШІ у процесі дослідження та підготовки рукопису. Відповідно до таксономії GAIDeT (2025) [14], наведені нижче завдання були делеговані інструментам генеративного ШІ за повного людського нагляду:

- Генерування ідей
- Аналіз ринкових трендів та/або патентного середовища
- Оцінювання новизни дослідження та виявлення прогалин
- Оптимізація коду
- Вичитування та редагування
- Переклад

Використаний інструмент генеративного ШІ: GPT-5.1 Thinking.

Повну відповідальність за зміст та результати курсового проекту несе автор.

Інструменти генеративного ШІ не зазначаються як автори та не несуть відповідальності за кінцеві результати.

Декларацію подав(ла): Олексій Іщенко

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. An overview of the Internet of Things (IoT) in building smart homes [Електронний ресурс]. – Frontline Professionals Journal, 26.04.2025. – Режим доступу: https://frontlineprofessionalsjournal.info/an-overview-of-the-internet-of-things-iot-in-building-smart-homes/ (дата звернення 12.11.2025)
2. Olenych I. Internet of Things Based System for Smart Home Climate Control on Fuzzy Logic / I. Olenych // Electronics and Information Technologies. – 2019. – №11(2). – С. 94–101. – Режим доступу: (дата звернення 04.12.2025)https://publications.lnu.edu.ua/collections/index.php/electronics/article/view/3572
3. Internet of Things (IoT)-Based System for Smart Home Heating and Cooling Control [Електронний ресурс] // ResearchGate. – Режим доступу: https://www.researchgate.net/publication/362810401_Internet_of_Things_IoT-Based_System_for_Smart_Home_Heating_and_Cooling_Control (дата звернення 28.11.2025)
4. Google Nest Help. Beginner’s guide to the Nest thermostats [Електронний ресурс]. – Режим доступу: https://support.google.com/googlenest/answer/9248184 (дата звернення 18.11.2025)
5. Home Assistant. Open source home automation that puts local control and privacy first [Електронний ресурс]. – Режим доступу: https://www.home-assistant.io/ (дата звернення 02.12.2025)
6. Complete Guide for DHT11/DHT22 Humidity and Temperature Sensor With Arduino [Електронний ресурс] // Random Nerd Tutorials. – Режим

доступу: https://randomnerdtutorials.com/complete-guide-for-dht11dht22-humidity-and-temperature-sensor-with-arduino/ (дата звернення 18.11.2025)

7. Arduino UNO R3. Офіційна документація [Електронний ресурс] // Arduino Documentation. – Режим доступу: https://docs.arduino.cc/hardware/uno-rev3
8. Arduino UNO Board Anatomy [Електронний ресурс] // Arduino Documentation. – Режим доступу: https://www.arduino.cc/en/Guide/BoardAnatomy (дата звернення 15.11.2025)
9. TMP36 Temperature Sensor. Datasheet / Adafruit Industries [Електронний ресурс]. – Режим доступу: https://cdn-learn.adafruit.com/downloads/pdf/tmp36-temperature-sensor.pdf (дата звернення 26.11.2025)
10. Windows Presentation Foundation (WPF) documentation [Електронний ресурс] // Microsoft Learn. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/ (дата звернення 20.11.2025)
11. WPF – Windows Presentation Foundation for .NET [Електронний ресурс] // GitHub, .NET Foundation. – Режим доступу: https://github.com/dotnet/wpf (дата звернення 20.11.2025)
12. Arduino Documentation [Електронний ресурс]. – Режим доступу: https://docs.arduino.cc/ (дата звернення 28.11.2025)
13. OpenAI. ChatGPT: Large Language Model for Natural Language Processing [Електронний ресурс]. – Режим доступу: https://openai.com/chatgpt (дата звернення 12.11.2025)
14. Генератор декларації GAIDeT [Електронний ресурс]. – Режим доступу: https://panbibliotekar.github.io/gaidet-declaration/index-uk.html (дата звернення 12.11.2025)

ДОДАТКИ

Додаток А (обов'язковий)

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ

Зав. кафедри АІІТ, проф., д.т.н.

_____ Олег БІСІКАЛО

(підпис)

„____" _____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на курсовий проект

«Емулятор системи мікроклімату»

08-31.ІР.009.00.000 ПЗ

Керівник курсового проекту

канд. техн. наук, доц. каф. АІІТ,

Ярослав КУЛИК_____

«___»_____2025р.

Виконавець ст. гр. 1ІСТ-22б,

Олексій ІЩЕНКО_____

«___»_____2025р.

Вінниця 2025

1. Назва та галузь застосування

1.1. Назва – Розробка програмного емулятора IoT-системи керування мікрокліматом приміщення та демонстраційного апаратного прототипу на базі Arduino.

1.2. Галузь застосування – Інтелектуальні інформаційні технології та автоматизація; системи «розумного дому»; навчальні та демонстраційні IoT-стенди.

2. Підстава для проведення розробки

Розробку курсового проєкту виконувати на підставі протоколу кафедри АІІТ протокол №__ від «__» _____ 2025 р. та індивідуального завдання, складеного і затвердженого кафедрою АІІТ ВНТУ.

3. Мета та призначення розробки

Метою курсового проєкту є створення програмного емулятора IoT-системи мікроклімату, який:

3.1. моделює ключові параметри середовища (температуру, вологість, освітленість, присутність людини);

3.2. реалізує автоматичне керування виконавчими каналами «обігрівач», «кондиціонер», «освітлення» за показами датчиків;

3.3. забезпечує вибір режимів роботи та візуалізацію станів у графічному інтерфейсі;

3.4. демонструє принцип роботи на спрощеному апаратному прототипі на базі Arduino (симуляція/перевірка в середовищі Tinkercad).

4. Вихідні дані для проведення розробки

Курсовий проєкт виконується вперше. В ході розробки повинні використовуватись/враховуватись такі джерела та матеріали:

4.1. Microsoft документація: Windows Presentation Foundation (WPF) documentation (офіційні матеріали).

4.2. Arduino Documentation (офіційні матеріали) та документація по Arduino UNO Rev3.

4.3. Онлайн-симулятор електроніки: Tinkercad Circuits (для моделювання та перевірки апаратного прототипу).

4.4. Довідкові матеріали по датчиках температури (зокрема TMP36) та приклади підключення/зчитування.

4.5. Оглядові/довідкові матеріали з IoT-систем «розумного дому» та термостатів (для аналізу аналогів і вимог).

5. Елементна база

5.1. Програмна частина (емулятор)

ПК/ноутбук під керуванням Windows.

Середовище розробки: .NET / C#; графічний інтерфейс: WPF.

Логічні модулі: моделі середовища, моделі сенсорів, модуль прийняття рішень (контролер), UI-візуалізація.

5.2. Апаратна частина (демонстраційний прототип)

Макетна плата (breadboard).

Мікроконтролер Arduino Uno R3 (або еквівалент у симуляції).

Датчик температури TMP36 (в симуляції Tinkercad).

Світлодіоди (3 шт.) з резисторами (індикація каналів: обігрівач / кондиціонер / освітлення).

Провідники/з’єднувальні дроти.

6. Кліматичні умови

Для програмної частини забезпечити стабільну роботу в умовах експлуатації ПК (типові офісні/домашні умови).

Для апаратного прототипу (навчального стенду) забезпечити працездатність у температурному діапазоні +5ºС…+40ºС і відносній вологості повітря не більше 75%.

7. Конструктивні вимоги

Програмна частина повинна бути реалізована як WPF-застосунок з інтуїтивним керуванням режимами, візуалізацією значень сенсорів і станів виконавчих каналів.

Апаратний прототип виконується у вигляді макетної схеми на breadboard (або її повного аналога в Tinkercad Circuits) з індикацією станів виконавчих каналів.

Взаємодія користувача: зміна умов (параметрів середовища) та спостереження реакції системи.

8. Стадії та етапи розробки

8.1. Пояснювальна записка

Вступ — __.__.2025 р.

1 Опис предметної області — __.__.2025 р.

2 Системи аналоги — __.__.2025 р.

3 Технічні засоби — __.__.2025 р.

4 Аналіз методів вирішення поставленої задачі — __.__.2025 р.

5 Опис логічної структури — __.__.2025 р.

6 Перевірка роботи системи — __.__.2025 р.

7 Вхідні та вихідні дані — __.__.2025 р.

8 Інструкція користувача — __.__.2025 р.

Висновки — __.__.2025 р.

Додатки (фрагменти коду, схеми, скріншоти/вікна симуляції) — __.__.2025 р.

8.2. Графічні матеріали

- Блок-схема/структурна схема логіки керування мікрокліматом.
- UML-діаграма варіантів використання (Use Case) взаємодії користувача із системою.
- UML-діаграма активності/станів для режимів роботи системи (за наявності).

- Схема апаратного прототипу в Tinkercad Circuits та матеріали перевірки (скріншоти/результати симуляції).

9. Порядок контролю і приймання

9.1. Хід виконання курсового проєкту контролюється керівником. Рубіжний контроль провести до .____.2025 р.

9.2. Захист курсового проєкту провести в період з _____.2025 р. по .__.2025 р.

Дата видачі «____» _____ 2025 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Додаток Б (обов’язковий)

Схема програми

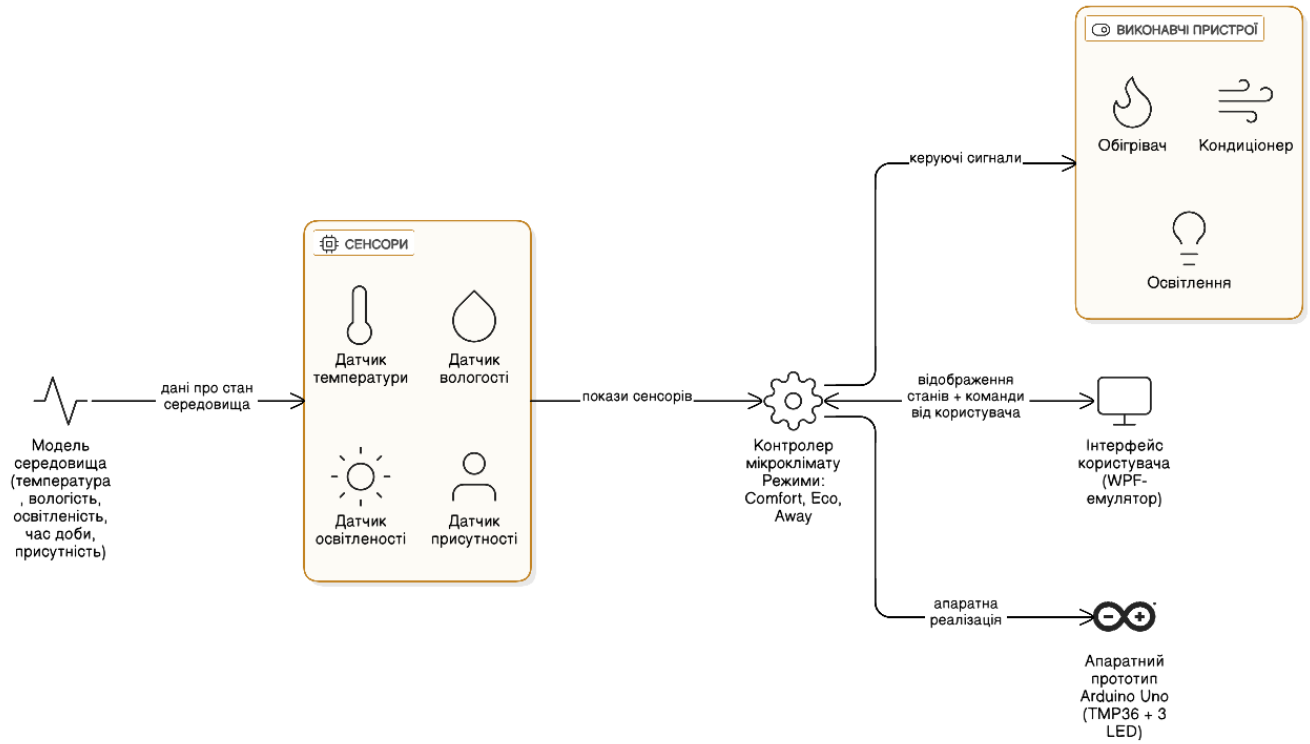

Рисунок А.1 - Структурна схема системи мікроклімату

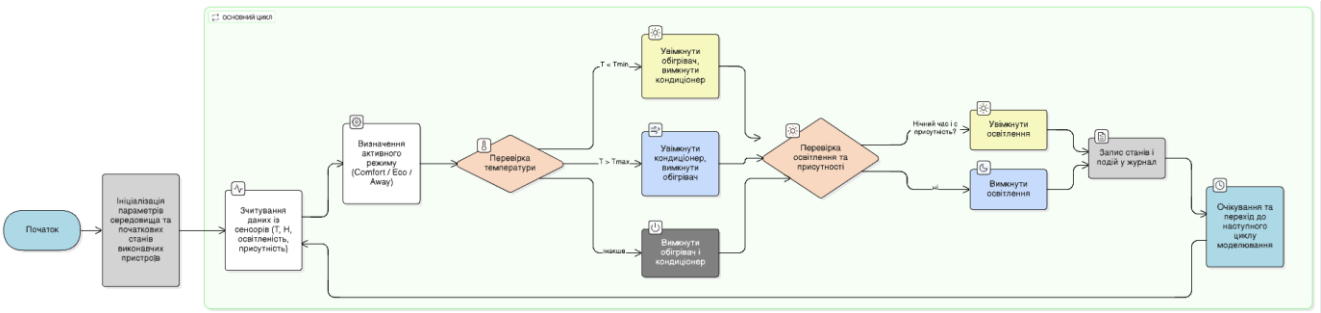

Рисунок А.2 - Блок-схема алгоритму роботи контролера

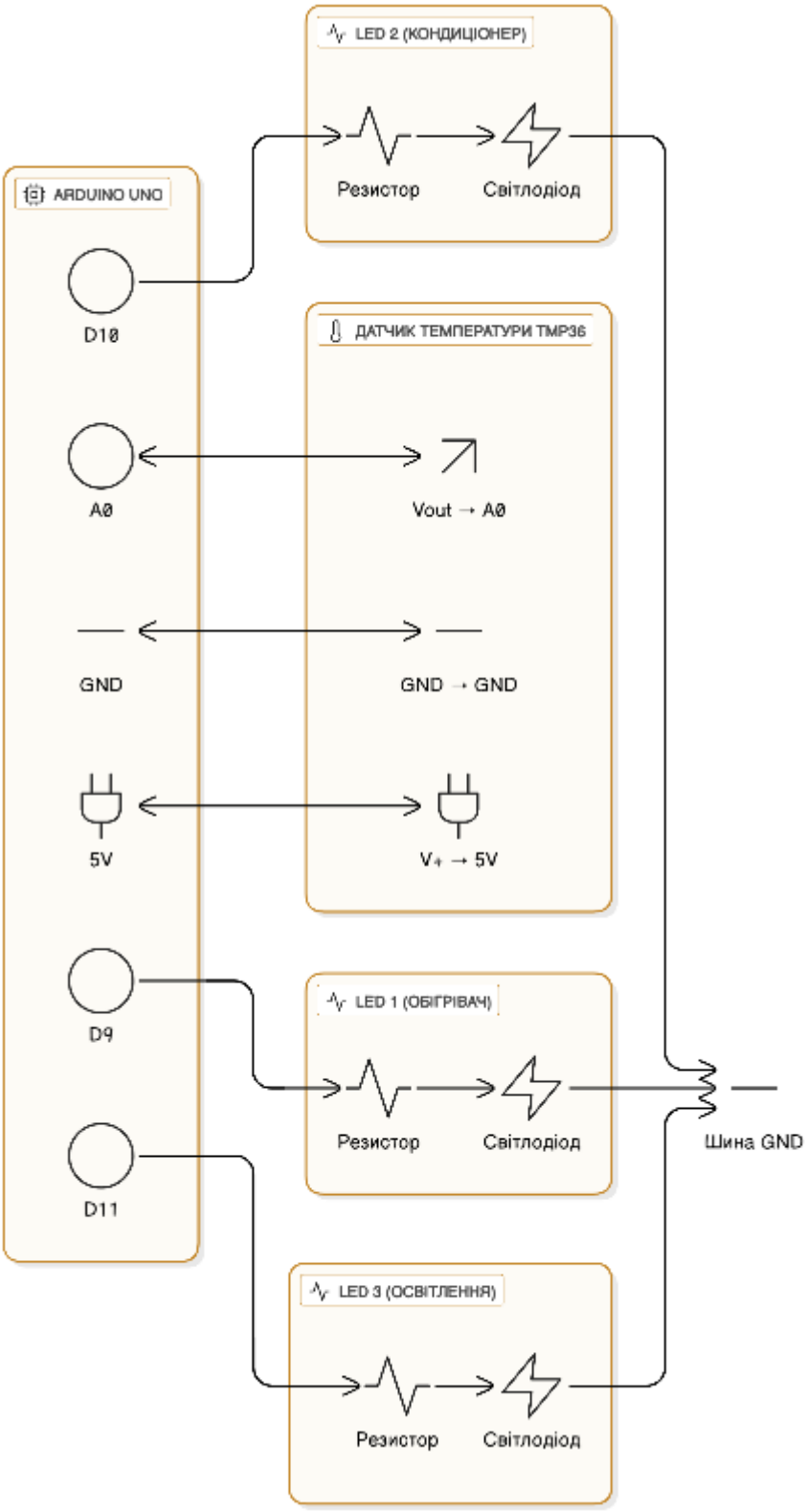

Рисунок А.3 - Схема підключення TMP36 та LED до Arduino Uno

Додаток В (обов’язковий)

Лістинг програми

ClimatController.cs:

```
namespace SmartClimate.Core;

public enum ClimateMode
{
    Comfort,
    Eco,
    Away
}

public class ClimateController
{
    private readonly TemperatureSensor _temp;
    private readonly LightSensor _light;
    private readonly MotionSensor _motion;
    private readonly HeaterActuator _heater;
    private readonly CoolerActuator _cooler;
    private readonly LampActuator _lamp;

    public ClimateMode Mode { get; set; } =
ClimateMode.Comfort;

    public ClimateController(
        TemperatureSensor temp,
        LightSensor light,
        MotionSensor motion,
        HeaterActuator heater,
        CoolerActuator cooler,
        LampActuator lamp)
    {
        _temp = temp;
        _light = light;
        _motion = motion;
        _heater = heater;
        _cooler = cooler;
        _lamp = lamp;
    }

    public void Step()
    {
        var t = _temp.ReadValue();
        var hasPerson = _motion.ReadValue();
        var light = _light.ReadValue();
```

```
        double minT;
        double maxT;
        double darkThreshold;

        switch (Mode)
        {
            case ClimateMode.Eco:
                minT = 19;
                maxT = 23;
                darkThreshold = 120;
                break;

            case ClimateMode.Away:
                // ширший діапазон – економія
                minT = 16;
                maxT = 28;
                darkThreshold = 80;
                break;

            default: // Comfort
                minT = 21;
                maxT = 24;
                darkThreshold = 200;
                break;
        }

        // керування обігрівачем/кондиціонером
        if (t < minT)
        {
            _heater.SetState(true);
            _cooler.SetState(false);
        }
        else if (t > maxT)
        {
            _heater.SetState(false);
            _cooler.SetState(true);
        }
        else
        {
            _heater.SetState(false);
            _cooler.SetState(false);
        }

        // світло: у режимі Away лампу не вмикаємо
        var wantLight = hasPerson && Mode != ClimateMode.Away
&& light < darkThreshold;
        _lamp.SetState(wantLight);
```

```
    }
}
```

EnviromentModel.cs:

```
public class EnvironmentModel
{
    public double TemperatureC { get; private set; } = 22.0;
    public double HumidityPercent { get; private set; } = 45.0;
    public double LightLux { get; private set; } = 150.0;
    public bool HasOccupant { get; private set; }

    // віртуальний час доби
    public TimeSpan TimeOfDay { get; private set; } =
TimeSpan.FromHours(12);

    // "вуличні" умови
    public double OutdoorTemperatureC { get; private set; } =
18.0;
    public double OutdoorLightLux { get; private set; } =
5000.0;

    // вплив виконавчих пристроїв
    internal double HeatingPower { get; set; }
    internal double CoolingPower { get; set; }
    internal double LightPower { get; set; }

    private readonly Random _rnd = new();

    public void ToggleOccupant() => HasOccupant = !HasOccupant;

    public void Update(TimeSpan dt)
    {
        var seconds = dt.TotalSeconds;

        // 1) крутимо віртуальний час
        TimeOfDay = TimeOfDay.Add(dt);
        if (TimeOfDay >= TimeSpan.FromDays(1))
            TimeOfDay -= TimeSpan.FromDays(1);

        var hours = TimeOfDay.TotalHours;

        // 2) модель вуличної температури (мін ~12° вночі, макс
~30° вдень)
        OutdoorTemperatureC = 20 + 8 * Math.Sin(2 * Math.PI *
(hours - 15) / 24);
        OutdoorTemperatureC = Math.Clamp(OutdoorTemperatureC,
12, 30);
```

```
        // 3) модель освітленості на вулиці (0 вночі, пік
вдень)
        var daylightFactor = Math.Max(0, Math.Sin(Math.PI *
(hours - 6) / 12)); // >0 між 6:00 та 18:00
        OutdoorLightLux = 25000 * daylightFactor;

        // 4) температура в кімнаті тягнеться до вуличної
        var driftToOutdoor = (OutdoorTemperatureC -
TemperatureC) * 0.01 * seconds;
        TemperatureC += driftToOutdoor;

        // невеликий шум
        TemperatureC += (0.01 * _rnd.NextDouble() - 0.005) *
seconds;

        // вплив обігрівача/кондиціонера
        TemperatureC += HeatingPower * 0.06 * seconds;
        TemperatureC -= CoolingPower * 0.06 * seconds;

        // 5) вологість - людина дає більше "пари"
        var humidityBase = HasOccupant ? 0.01 : 0.003;
        HumidityPercent += humidityBase * _rnd.NextDouble() *
seconds;
        HumidityPercent = Math.Clamp(HumidityPercent, 20, 80);

        // 6) освітленість усередині - частина вуличної + лампа
        LightLux = OutdoorLightLux * 0.1 + LightPower * 300 +
_rnd.Next(-10, 11);
        LightLux = Math.Clamp(LightLux, 0, 30000);
    }
}
```

Actuators.cs:

```
namespace SmartClimate.Core;

public abstract class Actuator
{
    protected readonly EnvironmentModel Environment;

    protected Actuator(EnvironmentModel env) => Environment =
env;

    public abstract string Name { get; }

    public abstract bool IsOn { get; protected set; }

    public abstract void SetState(bool on);
}
```

```
public class HeaterActuator : Actuator
{
    public HeaterActuator(EnvironmentModel env) : base(env) { }

    public override string Name => "Heater";

    public override bool IsOn { get; protected set; }

    public override void SetState(bool on)
    {
        IsOn = on;
        Environment.HeatingPower = on ? 1.0 : 0.0;
    }
}

public class CoolerActuator : Actuator
{
    public CoolerActuator(EnvironmentModel env) : base(env) { }

    public override string Name => "AC";

    public override bool IsOn { get; protected set; }

    public override void SetState(bool on)
    {
        IsOn = on;
        Environment.CoolingPower = on ? 1.0 : 0.0;
    }
}

public class LampActuator : Actuator
{
    public LampActuator(EnvironmentModel env) : base(env) { }

    public override string Name => "Lamp";

    public override bool IsOn { get; protected set; }

    public override void SetState(bool on)
    {
        IsOn = on;
        Environment.LightPower = on ? 1.0 : 0.0;
    }
}
```

Sensorss.cs:

```
namespace SmartClimate.Core;

public abstract class Sensor<T>
{
    protected readonly EnvironmentModel Environment;

    protected Sensor(EnvironmentModel env) => Environment =
env;

    public abstract string Name { get; }

    public abstract T ReadValue();
}

public class TemperatureSensor : Sensor<double>
{
    public TemperatureSensor(EnvironmentModel env) : base(env)
{ }

    public override string Name => "TempSensor";

    public override double ReadValue() =>
Environment.TemperatureC;
}

public class HumiditySensor : Sensor<double>
{
    public HumiditySensor(EnvironmentModel env) : base(env) { }

    public override string Name => "HumiditySensor";

    public override double ReadValue() =>
Environment.HumidityPercent;
}

public class LightSensor : Sensor<double>
{
    public LightSensor(EnvironmentModel env) : base(env) { }

    public override string Name => "LightSensor";

    public override double ReadValue() => Environment.LightLux;
}

public class MotionSensor : Sensor<bool>
{
    public MotionSensor(EnvironmentModel env) : base(env) { }
```

```
    public override string Name => "MotionSensor";

    public override bool ReadValue() =>
Environment.HasOccupant;
}
```