КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Електронні системи розумного будинку»

Скаля Єгор Владиславович
(прізвище, ім’я та по батькові студента повністю)

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
д.т.н., професор Цюцюра С.В.

„___” _____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Електронні системи розумного будинку»

Виконав: студент 4-го курсу, групи КН-20-1

Спеціальності: 122 «Комп`ютерні науки

Спеціалізація: «Інформаційні управляючі системи і технології»
(шифр і назва напряму підготовки, спеціальності)

Скаля. Є.В
(прізвище та ініціали)

Керівник доц. Саченко І.А.
(прізвище та ініціали)

Рецензент к.т.н., доц. Шабала Є.Є.
(прізвище та ініціали)

Київ, 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Кафедра: ____інформаційних технологій
Освітній рівень: «бакалавр» за ОП
Спеціальність: 122 «Комп`ютерні науки»
Спеціалізація: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
к.т.н., професор Гончаренко Т.А.

_____
„___” _____2024 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ АТЕСТАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

Скаля Єгор Владиславович
Тема роботи: Електронні системи розумного будинку
затверджена наказом ректора КНУБА № _____ від « « лютого 2024 р.
2. Керівник роботи: Саченко Ілля Анатолійович, доцент
кафедри інформаційних технологій проектування та прикладної математики
3. Строк подання студентом роботи до захисту:
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз предметної області та постановка задачі
Р.2. Алгоритмічне та математичне забезпечення
Р.3. Розробка програмного забезпечення
Р.4. Ергономіка інформаційних технологій
5. Інформаційні слайди:

С.1. Принцип роботи методу Бройдена-Флетчера-Гольдфарба-Шанно
С.2. Блок-схеми основних алгоритмів
С.3. Датасет поведінкових патернів
С.4. Модель бази даних
С.5. Діаграми класів трьохрівневої архітектури
С.6. Тестові приклади

6. Календарний план виконання атестаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Січень 2024 р.
Р. 2. Алгоритмічне та математичне забезпечення	Лютий 2024 р.
Р. 3. Розробка програмного забезпечення	Березень 2024 р.
Тестовий приклад програми	Квітень 2024 р.
Р. 4. Ергономіка інформаційних технологій	Травень 2024 р.
Остаточне оформлення роботи	Травень 2024 р.
Направлення роботи на рецензування	Червень 2024 р.
Попередній захист роботи на кафедрі	Червень 2024 р.

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Ергономіка інформаційних технологій	д.т.н. проф. Терентьєв О.О.		
Прийом програмного продукту	к.т.н., доц. Шабала Є.Є.		

8. Дата видачі завдання: 09 грудня 2023 р.

Керівник _____ Саченко І.А.
(підпис) (прізвище та ініціали)

Бакалавр _____ Скаля Є.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка електронної системи розумного будинку з використанням технологій штучного інтелекту.

Атестаційна випускна робота бакалавра за спеціальністю: 122 «Комп’ютерні науки», спеціалізація: «Інформаційні управляючі системи і технології». – Київський національний університет будівництва та архітектури. – Київ, 2020.

Робота присвячена розробці системи «Розумний будинок», що використовує технології Інтернету речей (IoT) та методи машинного навчання для автоматизації управління домогосподарством. Інструментом реалізації розроблюваної системи виступає платформа ML.NET від Microsoft, яка дозволяє втілити складні алгоритми машинного навчання для аналізу та прогнозування поведінки системи «Розумний будинок».

Ключові слова: розумний будинок, IoT, машинне навчання, архітектура системи, безпека, алгоритм Бройдена-Флетчера-Гольдфарба-Шанно, ML.NET.

SUMMARY

Development of an electronic system of a smart house using artificial intelligence technologies.

Attestation graduation thesis of a bachelor in the specialty: 122 "Computer science", specialization: "Information management systems and technologies". - Kyiv National University of Construction and Architecture. - Kyiv, 2020.

The work is devoted to the development of a "Smart House" system that uses Internet of Things (IoT) technologies and machine learning methods to automate household management. The tool for implementing the developed system is the ML.NET platform from Microsoft, which allows you to implement complex machine learning algorithms for analyzing and predicting the behavior of the "Smart House" system.

Keywords: smart home, IoT, machine learning, system architecture, security, Broyden-Fletcher-Goldfarb-Shannot algorithm, ML.NET.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – бази даних

БФГШ – алгоритм Бройдена-Флетчера-Гольдфарба-Шанно

МВП – метод випадкового пошуку

МК – мікроконтролери

МН – машинне навчання

РБ – розумний будинок

СГС – стохастичний градієнтний спуск

ШІ – штучний інтелект

API – програмний інтерфейс застосувань (Application Programming Interface)

IDE – інтегроване середовище розробки (Integrated Development Environment)

IoT – Інтернет речей (Internet of Things)

ML.NET – фреймворк для машинного навчання від Microsoft

AI – алгоритми інтелекту

ВСТУП

У сучасному світі, що стрімко розвивається, технології займають дедалі важливіше місце у повсякденному житті людини. Одним з найбільш прогресивних напрямків є розробка та впровадження електронних систем розумного будинку, що є втіленням ідей Інтернету речей (IoT). Ці системи спрямовані на підвищення комфорту, економії енергії, безпеки та ефективності управління домогосподарствами. Вони дозволяють інтегрувати різноманітні пристрої та апаратуру в єдину мережу, якою можна керувати автоматично або дистанційно. Сутність розумного будинку полягає не лише в автоматизації побутових процесів, але й у створенні адаптивного та інтерактивного середовища, що реагує на потреби його мешканців.

Станом на сьогодні проблематика розумних будинків набуває особливої значущості через швидке зростання міських агломерацій, підвищення вартості енергоресурсів та необхідність забезпечення високого рівня життя. Розвиток цих систем є важливим не лише з точки зору покращення якості життя, але й у контексті енергозбереження, екології та безпеки. Підставами для розробки теми електронних систем розумного будинку є стрімкий розвиток цифрових технологій, зростаючі вимоги до комфорту та безпеки житлового середовища, а також потреба у ефективному управлінні ресурсами.

Актуальність дослідження електронних систем розумного будинку обумовлена їх здатністю радикально трансформувати звичайне житлове середовище, зробивши його більш зручним, безпечним та економічно ефективним. Крім того, у контексті України, ця тема є особливо важливою з огляду на потребу в модернізації житлового фонду, підвищення енергоефективності та впровадження інноваційних технологій у побутовий сектор. Враховуючи глобальні тенденції до сталого розвитку та екологічної відповідальності, розумні будинки можуть зіграти ключову роль у досягненні цих цілей на національному рівні.

Шляхом критичного аналізу та порівняння з відомими рішеннями можна виявити, що багато існуючих систем мають обмеження у плані інтеграції,

масштабування та індивідуалізації. Тому розробка нових, більш гнучких та вдосконалених систем розумного будинку, які б враховували специфіку українського ринку та потреби споживачів, є актуальним та перспективним напрямком досліджень. Значущість цієї роботи полягає у створенні основи для подальшого розвитку інтелектуальних житлових середовищ, що зможуть адаптуватися до змінюваних умов та потреб мешканців, сприяючи підвищенню якості життя та екологічної стійкості.

Мета дослідження: розробка інтегрованої системи розумного будинку, яка забезпечує підвищення безпеки, комфорту та енергоефективності житлового простору шляхом використання сучасних технологій Інтернету речей та методів машинного навчання.

Для досягнення поставленої мети було визначено наступні основні задачі:

– аналіз предметної області та існуючих рішень у сфері розумних будинків;

– вибір відповідних технологій та методів для розробки інтелектуальної системи;

– розробка архітектури системи та математичної моделі для автоматизації побутових процесів;

– реалізація прототипу системи з використанням платформи ML.NET;

– проведення експериментів та аналіз отриманих результатів для оцінки ефективності розробленої системи.

Реалізація цих задач дозволить створити ефективну та надійну систему розумного будинку, що відповідає сучасним вимогам та очікуванням, і стане вагомим внеском у розвиток технологій Інтернету речей та штучного інтелекту в Україні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області

Електронні системи розумного будинку представляють собою комплексні інтегровані рішення, які використовують авангардні технології для управління різними процесами та пристроями в домашньому господарстві. Ці системи об'єднують в собі елементи автоматизації, інтернету речей (IoT), штучного інтелекту та машинного навчання, щоб зробити життя більш комфортним, безпечним та енергоефективним.

Основною метою електронних систем розумного будинку є спрощення повсякденного життя людей шляхом автоматизації рутинних задач. Вони дозволяють контролювати освітлення, температуру, мультимедіа системи, вентиляцію, а також інші побутові прилади та системи безпеки з одного централізованого інтерфейсу, яким може бути смартфон, планшет або стаціонарний пульт керування. Це не тільки сприяє збільшенню комфорту та зручності, але й допомагає знизити споживання енергії через оптимізацію використання ресурсів.

Безпека домівок є одним з ключових аспектів електронних систем розумного будинку. Завдяки інтеграції охоронних систем, відеокамер, датчиків руху, диму, витоку газу та води, такі системи забезпечують високий рівень захисту для мешканців. Вони можуть автоматично сповіщати власників та відповідні служби про спроби несанкціонованого доступу, пожежі або інші аварійні ситуації, що значно знижує ризик зломів, крадіжок та інших небезпек.

Окрім того, системи розумного будинку можуть бути налаштовані на автоматичне відключення електроприладів або водопостачання у випадку виявлення витоку, що запобігає можливим матеріальним збиткам [1]. Використання елементів штучного інтелекту дозволяє системі «вчитися» на поведінці мешканців, автоматично адаптуючи режими роботи для максимального комфорту та безпеки.

Архітектура «розумного будинку» може бути втілена через дві основні структурні концепції: централізовану та децентралізовану. У рамках

централізованої моделі, ключові компоненти системи, включаючи управлінські механізми, головний обчислювач та виконавчі елементи, зібрані в єдину телекомунікаційну мережу (рис. 1.1). Цей підхід забезпечує можливість здійснювати організоване та синхронізоване керування, гарантуючи надійну трансляцію команд по всьому об'єднанню.

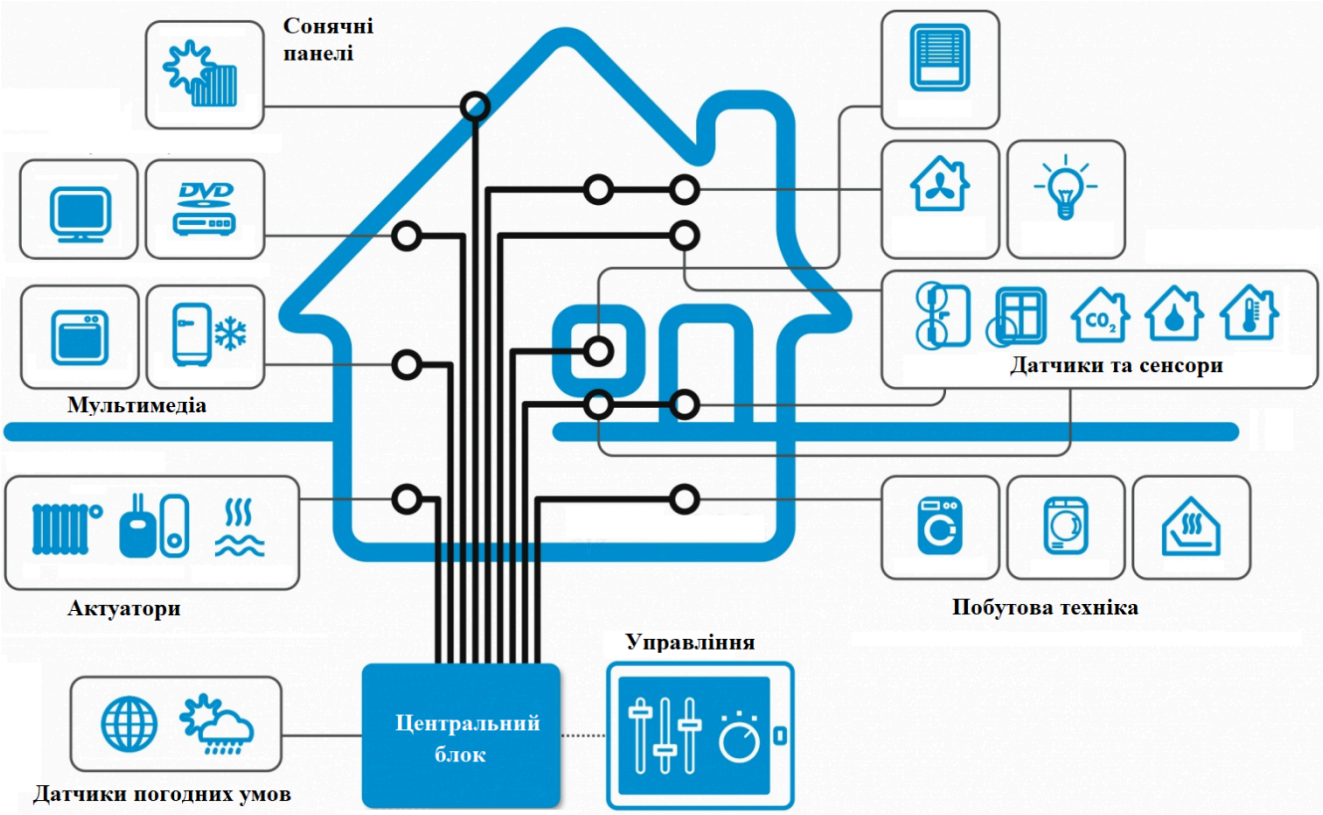

Рисунок 1.1. Концепція централізованої архітектури

Інтерфейси керування слугують зв'язною ланкою між користувачем і системою, дозволяючи управляти функціями «розумного будинку» за допомогою різноманітних пристроїв, починаючи з традиційних пультів дистанційного керування і закінчуючи сучасними сенсорними панелями та мобільними додатками. Автоматизація надає системі здатність автономно реагувати на зміни у навколишньому середовищі завдяки використанню вбудованих датчиків, котрі моніторять освітленість, наявність людей, температуру, вологість та інші важливі параметри.

Центральний процесор виконує роль мозку системи, відповідаючи за приймання, аналіз та обробку інформації, отриманої від інтерфейсів управління та датчиків. Він зберігає сценарії дій, задані користувачем, і може автоматично

створювати виконавчі команди, базуючись на алгоритмах штучного інтелекту [2]. Такий контролер координує діяльність усіх підсистем «розумного будинку», відправляючи оперативні інструкції до виконавчих пристроїв для виконання встановлених завдань.

Переваги централізованої архітектури «розумного будинку»:

– єдине керування та синхронізація. Централізована система дозволяє управляти всіма підключеними пристроями та системами з одного місця, забезпечуючи високий рівень синхронізації та координації дій;

– спрощення інсталяції та обслуговування. Маючи єдиний центр управління, система легше монтується та обслуговується, оскільки всі зміни та оновлення зосереджені в одному місці;

– ефективність передачі даних. Централізована структура забезпечує високу ефективність та надійність у передачі команд і даних між різними компонентами системи, мінімізуючи затримки та помилки;

– вищий рівень безпеки. Централізоване управління дозволяє більш ефективно контролювати доступ до системи та керувати заходами безпеки, оскільки всі критичні функції зосереджені в одному контролері.

Недоліки централізованої архітектури:

– точка єдиного збою. Централізована система має недолік у вигляді точки єдиного збою. Якщо центральний контролер виходить з ладу, це може призвести до відмови всієї системи;

– масштабованість. Можуть виникати складнощі з масштабуванням системи, особливо при спробі інтегрувати нові технології або значно розширити кількість підключених пристроїв;

– висока вартість. Централізовані системи часто вимагають значних вкладень на етапі інсталяції через необхідність придбання потужного центрального контролера та спеціалізованого обладнання;

– залежність від центрального контролера. Усі рішення приймаються та виконуються через центральний контролер, що може створювати затримки у

відгуку системи на зміни в навколишньому середовищі або команди користувача, особливо в великих системах [3].

Централізована архітектура пропонує зручне та ефективне управління «розумним будинком», але в той же час вона має певні обмеження, а саме: високу вартість та залежність від центрального контролера .

В архітектурі децентралізованих інтелектуальних систем управління домогосподарством перевагу надають розподіленому управлінню замість традиційного централізованого підходу. Тут кожен компонент, чи то пристрій, чи модуль, обладнаний власним механізмом виконання дій (рис. 1.2). Це дозволяє датчикам, які реєструють зміни або події в середовищі проживання, безпосередньо активувати виконавчі механізми, наприклад, реле або електромеханічні засоби.

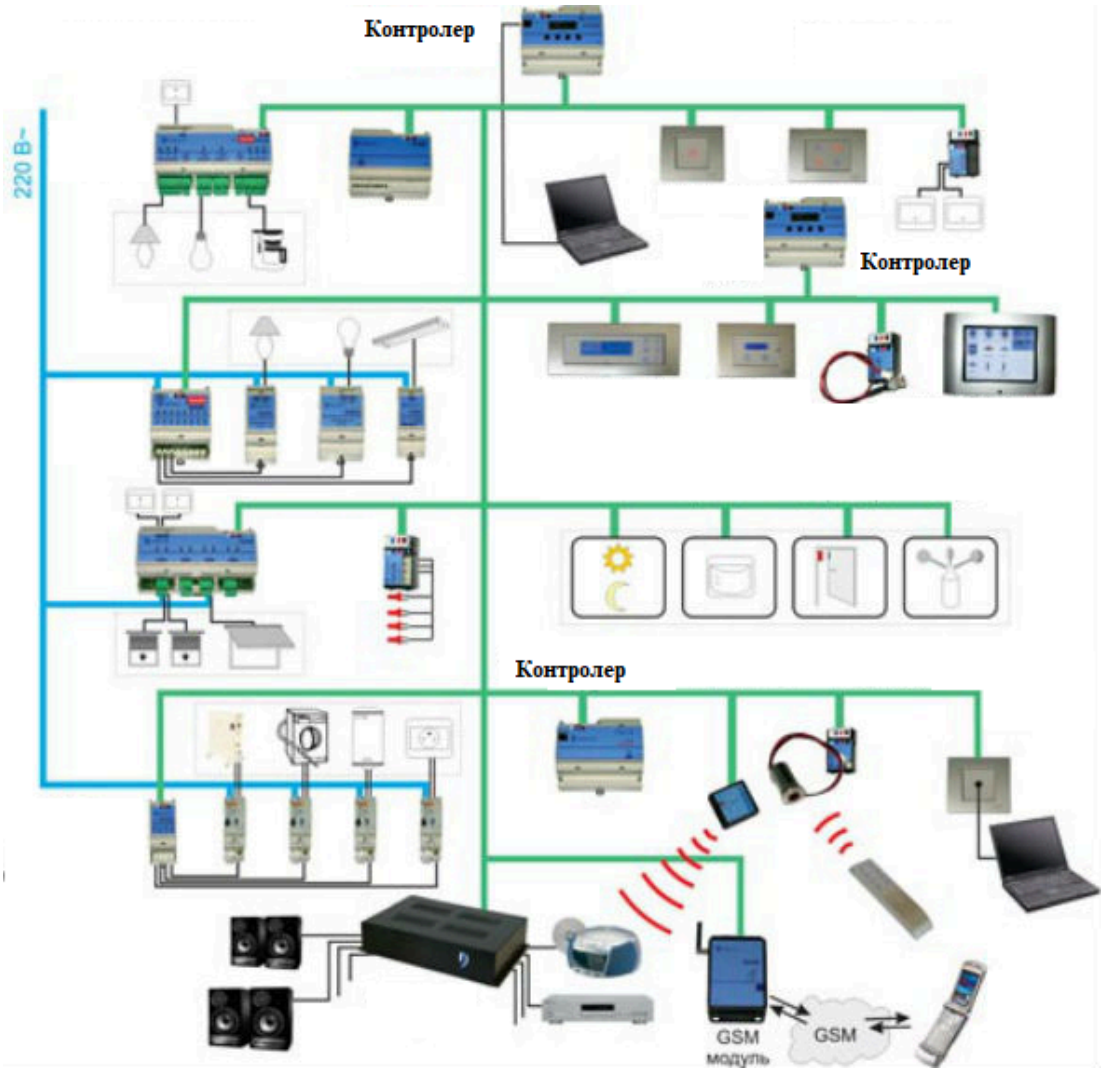

Рисунок 1.2. Концепція децентралізованої архітектури

Така архітектура наділяє систему можливістю самостійно регулювати процеси на локальному рівні, дозволяючи кожному елементу автоматично адаптуватися до змін умов середовища. Це, у свою чергу, збільшує гнучкість та адаптивність системи, яку можна налаштувати для задоволення індивідуальних потреб та побажань користувачів.

Проте, потреба в синхронізації та координації між численними розподіленими компонентами може ускладнити керування та моніторинг системи в цілому. Такий підхід може вимагати вдосконаленої інфраструктури для забезпечення ефективної взаємодії між елементами, а також створює додаткові виклики для централізованого обслуговування та діагностики. Отже, хоча децентралізована архітектура і забезпечує компонентам більшу незалежність, вона також вимагає складніших методів інтеграції та управління, аби згуртувати ці елементи в єдину функціональну систему.

Переваги децентралізованої архітектури «розумного будинку» [4]:

– автономність компонентів. Кожен елемент системи має власну логіку роботи, що дозволяє йому самостійно реагувати на зміни в оточенні без необхідності команд з центрального контролера;

– підвищена надійність. Система менш схильна до повної відмови, оскільки збій одного компонента не веде до зупинки всієї системи, а лише впливає на обмежену функціональність;

– гнучкість та масштабованість. Легко додавати, видаляти або замінювати окремі компоненти системи без необхідності перепроектування або пере-налаштовування всієї системи;

– адаптивність до умов середовища. Компоненти можуть незалежно адаптуватися до змін умов, забезпечуючи оптимальну роботу системи відповідно до поточних потреб.

Недоліки децентралізованої архітектури «розумного будинку»:

– складність управління та моніторингу. Координація та синхронізація між різними автономними компонентами може бути складною, особливо в великих системах;

– потреба в розширеній інфраструктурі. Ефективна взаємодія між розподіленими компонентами вимагає розвиненої мережевої інфраструктури та протоколів комунікації;

– виклики в інтеграції. Інтегрування нових компонентів або технологій вимагає забезпечення їх сумісності з існуючими елементами системи, що може бути непростим завданням;

– складнощі в діагностиці та обслуговуванні. Виявлення та усунення несправностей може бути ускладнене через децентралізовану природу системи, оскільки проблеми потрібно ідентифікувати та вирішувати на рівні окремих компонентів [5].

Отже, децентралізована архітектура може бути більш вигідною в умовах, де потрібна висока автономність окремих зон чи пристроїв, а також там, де важлива швидка реакція на зміни умов.

Таким чином, електронні системи розумного будинку не тільки спрощують управління домашнім господарством і підвищують комфорт життя, але й забезпечують комплексний захист домівок від різноманітних загроз. Вони відіграють ключову роль у створенні безпечного, енергоефективного та інтелектуального житлового простору, що відповідає вимогам сучасного життя.

1.2 Аналіз об’єкту дослідження

Електронні системи розумного будинку становлять інноваційний напрям у технологіях автоматизації житла, орієнтований на створення інтелігентного середовища, що здатне оптимізувати різні аспекти повсякденного життя. Такі системи інтегрують в себе різноманітні пристрої та алгоритми для керування освітленням, кліматом, медіа, безпекою та іншими елементами домашнього комфорту, роблячи їх взаємопов'язаними та автоматично регульованими.

В основі електронних систем розумного будинку лежить взаємодія між різними технологічними компонентами, які разом створюють інтегроване середовище для автоматизації та оптимізації домашнього життя. Серцевиною цієї системи є сенсори та датчики, розроблені для моніторингу найрізноманітніших

аспектів умов проживання. Вони здатні виявляти коливання температури, зміни в рівні освітленості, вологості, а також фіксувати рух чи присутність людей у приміщенні. Ці пристрої збирають важливі дані, які стають основою для подальших дій системи.

Виконавчі пристрої, що включають в себе різноманітні механізми та апарати, відіграють ключову роль у реалізації команд системи. Вони активуються відповідно до отриманих вказівок, запускаючи певні дії, такі як включення чи вимикання освітлення, коригування температури або вентиляції, забезпечуючи тим самим комфорт та ефективність енергоспоживання.

Керувальні пристрої слугують засобом взаємодії між системою та її користувачами, дозволяючи встановлювати необхідні параметри роботи або модифікувати налаштування системи відповідно до особистих переваг. Це можуть бути спеціалізовані панелі управління, смартфони або планшети з відповідним програмним забезпеченням, що надають доступ до широкого спектру функцій системи розумного будинку.

Центральний процесор системи розумного будинку виконує роль керівного вузла, що забезпечує оркестрацію та взаємодію між усіма компонентами. Обробляючи інформацію, яка надходить від сенсорів, враховуючи вказівки користувачів, процесор використовує складні алгоритми для аналізу зібраних даних. Це дозволяє йому ухвалювати обдумані рішення, оптимізуючи роботу системи з метою задоволення потреб мешканців і адаптації до змінних умов середовища. Процесор, ефективно керуючи виконавчими механізмами, гарантує інтелектуальне управління ресурсами домогосподарства, що сприяє створенню адаптивного і комфортного простору для життя [6].

Системи розумного будинку об'єднують в собі великий арсенал технологій для створення комфортного, безпечного та енергоефективного житлового середовища. Вони реалізують автоматизацію побутових процесів, що охоплює контроль за освітленням, температурою повітря, вологістю та іншими параметрами, які впливають на комфорт проживання. Завдяки цьому, мешканці

можуть насолоджуватися ідеальною атмосферою вдома з мінімальними зусиллями.

Безпека в системах розумного будинку займає особливе місце. З використанням відеокамер, сенсорів руху, датчиків відкриття дверей та вікон, а також систем сповіщення про аварійні ситуації, як-от детектори диму чи витоку води, створюється надійний захист для житла. Це дозволяє виявляти потенційні загрози на ранніх етапах та вживати необхідних заходів для їх нейтралізації.

Енергозбереження є важливою частиною концепції розумного будинку, де через інтелектуальне керування опаленням, охолодженням, вентиляцією та іншими системами мінімізується витрата енергії [7]. Система здатна аналізувати патерни споживання та автоматично регулювати роботу обладнання для оптимального використання ресурсів.

Також, комфорт і зручність, які пропонують системи розумного будинку, перетворюють домашнє середовище на ідеально адаптований до потреб мешканців простір. Від автоматичного регулювання освітлення до створення сценаріїв для різних життєвих ситуацій, таких як перегляд фільмів або вечеря в колі сім'ї, системи розумного будинку забезпечують неперевершену зручність та адаптацію до індивідуального стилю життя.

1.3 Опис предмету дослідження

Предмет дослідження у контексті електронних систем розумного будинку зосереджується на вивченні та аналізі функціональності, архітектури, інтеграції та управління цих систем. Цей предмет включає в себе розгляд комплексу технологій, що дозволяють автоматизувати ряд домашніх процесів, покращити енергоефективність, безпеку, комфорт і зручність проживання в житлових просторах. Він охоплює детальний аналіз таких аспектів:

– технологічна інфраструктура. Дослідження технологій, на яких базуються системи розумного будинку, включно з бездротовими та дротовими комунікаційними стандартами, протоколами передачі даних, а також апаратним та

програмним забезпеченням, що забезпечує інтеграцію та взаємодію між різними компонентами системи;

– автоматизація та контроль. Вивчення механізмів автоматизації домашніх процесів, як-от управління освітленням, кліматом, безпекою, мультимедіа та іншими системами. Аналіз способів контролю та інтерфейсів для взаємодії користувачів з системою, включаючи сенсорні панелі, мобільні додатки та голосові команди;

– безпека та моніторинг. Детальний огляд рішень для забезпечення безпеки домівок, що включає системи спостереження, сигналізації, контролю доступу та аварійного реагування. Розгляд технологій детекції та попередження небезпечних ситуацій, таких як пожежі, витоки газу або води;

– енергоефективність та екологічність. Аналіз способів, за допомогою яких системи розумного будинку сприяють зниженню споживання енергії та водних ресурсів. Вивчення інтелектуального управління енергопостачанням, оптимізації роботи побутових приладів і систем опалення, вентиляції та кондиціонування;

– комфорт та адаптивність. Оцінка можливостей систем розумного будинку створювати адаптивне житлове середовище, яке відповідає на індивідуальні потреби та уподобання мешканців, забезпечуючи високий рівень комфорту та зручності.

Предмет дослідження систем розумного будинку вимагає комплексного підходу, що включає технічні, функціональні та соціальні аспекти, спрямовані на створення інтелектуального, безпечного, енергоефективного та комфортабельного домашнього середовища.

1.4 Аналіз актуальності

Актуальність впровадження електронних систем розумного будинку в сучасному суспільстві є незаперечною, враховуючи сучасні тенденції та потреби. Значний прогрес у сфері інформаційних технологій, зокрема в розвитку Інтернету речей, штучного інтелекту та машинного навчання, створює умови для

радикального трансформування житлових просторів. Відповідно до даних дослідницьких агентств, ринок розумного будинку продемонструє значне зростання протягом наступних кількох років, відображаючи зростаючий інтерес споживачів до інноваційних рішень у сфері житлового комфорту та енергоефективності.

Однією з ключових переваг систем розумного будинку є їхня здатність забезпечити високий рівень безпеки та контролю за житловим простором, що є важливим фактором у сучасному динамічному світі. Зростаюча кількість випадків злому та несанкціонованого доступу до приватних помешкань спонукає до пошуку ефективних рішень, здатних забезпечити захист майна та особистої безпеки мешканців.

Водночас, важливим аспектом є стрімке зростання вартості енергоресурсів та необхідність їх раціонального використання. Системи розумного будинку дозволяють оптимізувати енергоспоживання, автоматично регулюючи роботу опалення, освітлення та інших систем відповідно до актуальних потреб мешканців та зовнішніх умов.

Крім того, глобалізація та пандемія COVID-19 виявили потребу в адаптації житлового простору до нових реалій, де домівка стає не лише місцем проживання, але й робочим простором. Системи розумного будинку пропонують рішення, що забезпечують комфортні та ергономічні умови для дистанційної роботи та навчання.

Необхідно зазначити, що наявність віддалених та недостатньо обслуговуваних регіонів, де доступ до новітніх технологій обмежений, вимагає розробки та адаптації систем розумного будинку, здатних функціонувати в різноманітних умовах. Можливість дистанційного управління та моніторингу житлових систем через Інтернет відкриває доступ до сучасних технологічних рішень для ширшого кола користувачів.

1.5 Аналіз існуючих рішень

Аналіз існуючих рішень у сфері систем розумного будинку є ключовим етапом у розробці та впровадженні нових програмних продуктів. Вивчення та порівняння наявних технологій дозволяють не лише визначити поточний стан ринку, але й ідентифікувати потенційні можливості для інновацій та поліпшень. У цьому контексті, ретельний огляд аналогічних програмних розробок є важливим для забезпечення конкурентоспроможності, адаптації до змінних потреб користувачів та врахування останніх тенденцій у технологічному прогресі.

Apple HomeKit є частиною екосистеми Apple, розробленою з метою надання користувачам зручного та безпечного способу керування розумними пристроями в їхніх домівках за допомогою iPhone, iPad, Apple Watch або Mac. Система призначена для інтеграції широкого спектру розумних пристроїв – від освітлення та систем опалення до жалюзі, безпеки та різноманітних сенсорів — у єдину мережу, що дозволяє користувачам легко керувати ними через інтуїтивно зрозумілий інтерфейс.

На рис. 1.3 представлено приклад інтерфейсу для роботи користувача із системою.

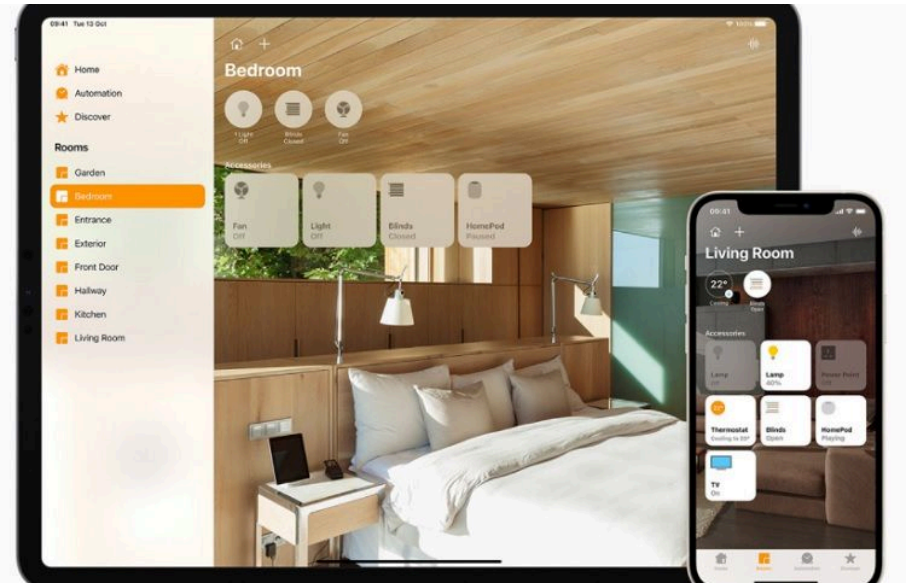

Рисунок 1.3. Приклад інтерфейсу системи «Apple HomeKit»

Архітектура HomeKit орієнтована на безпеку та приватність, використовуючи шифрування для забезпечення захисту даних користувачів під час передачі команд між пристроями та центральними вузлами управління. Система

дозволяє створювати сценарії для автоматизації різних процесів (наприклад, «Я вдома» або «Я вийшов»), що спрощує повсякденне використання різних пристроїв у домі. Основою архітектури HomeKit є концепція «моста», який з'єднує всі сумісні пристрої з Apple екосистемою, дозволяючи їм спілкуватися через локальну мережу або через хмару. Це забезпечує високу гнучкість у керуванні домашніми пристроями, надаючи можливість доступу до них з будь-якої точки світу, де є доступ до інтернету.

Завдяки тісній інтеграції з іншими продуктами Apple, HomeKit також пропонує користувачам переваги голосового управління через Siri, що додає додатковий рівень зручності та інтерактивності у взаємодії з розумним будинком.

Переваги Apple HomeKit:

– інтеграція з Apple екосистемою. HomeKit ідеально інтегрований з іншими продуктами Apple, забезпечуючи плавне керування розумними пристроями через iPhone, iPad, Apple Watch, або Mac. Це створює зручний і однорідний користувацький досвід;

– висока безпека та приватність. Apple відома своєю увагою до безпеки та приватності користувачів. HomeKit використовує суворе шифрування для захисту даних користувачів та команд, що передаються між пристроями;

– голосове управління через Siri. Можливість керувати пристроями за допомогою голосових команд Siri додає додаткову зручність та робить інтеракцію з системою розумного будинку більш інтуїтивною;

– автоматизація та сценарії. HomeKit дозволяє створювати складні сценарії та автоматизації, які можуть бути налаштовані під індивідуальні потреби користувача, підвищуючи ефективність та комфорт використання розумного будинку.

Недоліки Apple HomeKit:

– обмежена сумісність з пристроями. Не всі розумні пристрої сумісні з HomeKit, що може обмежити вибір для користувачів, які бажають інтегрувати існуючі пристрої в систему розумного будинку;

– висока вартість. Продукти, що підтримують HomeKit, часто мають вищу ціну порівняно з аналогами, що не є сумісними з HomeKit. Це може збільшити загальні витрати на створення системи розумного будинку;

– залежність від екосистеми Apple. Хоча інтеграція з продуктами Apple є перевагою, вона також створює залежність від екосистеми Apple, обмежуючи користувачів, які використовують пристрої на інших платформах;

– комплексність налаштування. Для користувачів, не знайомих з технологіями Apple або новачків у сфері розумного будинку, налаштування та керування системою HomeKit може виявитися складним, вимагаючи певного часу на освоєння [8].

Отже, Apple HomeKit представляє собою потужну систему управління розумним будинком, що вирізняється тісною інтеграцією з широким спектром продуктів Apple, високим рівнем безпеки та приватності, а також зручністю голосового керування через Siri. Ця система забезпечує великі можливості для автоматизації та персоналізації домашнього середовища, роблячи повсякденне життя більш комфортним та ефективним. Водночас, обмежена сумісність з пристроями та вища вартість продуктів, сумісних з HomeKit, а також залежність від екосистеми Apple та потенційні складнощі з налаштуванням системи можуть стати перепонами для деяких користувачів. Незважаючи на це, для прихильників продукції Apple, які прагнуть максимальної інтеграції своїх пристроїв та високого рівня автоматизації житла, HomeKit є одним із кращих рішень на ринку систем розумного будинку.

Wink Hub є універсальним центром для інтеграції та управління розумними пристроями в домі. Його головна мета полягає в тому, щоб забезпечити користувачам зручний спосіб керування різними розумними пристроями — від освітлення та термостатів до замків та жалюзі — через єдиний інтерфейс. Wink Hub підтримує широкий спектр бездротових технологій зв'язку, включаючи Wi-Fi, Bluetooth, Z-Wave та Zigbee, що дозволяє йому з'єднуватися з великою кількістю розумних пристроїв різних виробників. На рис. 1.4 представлено приклад інтерфейсу для взаємодії користувача із системою.

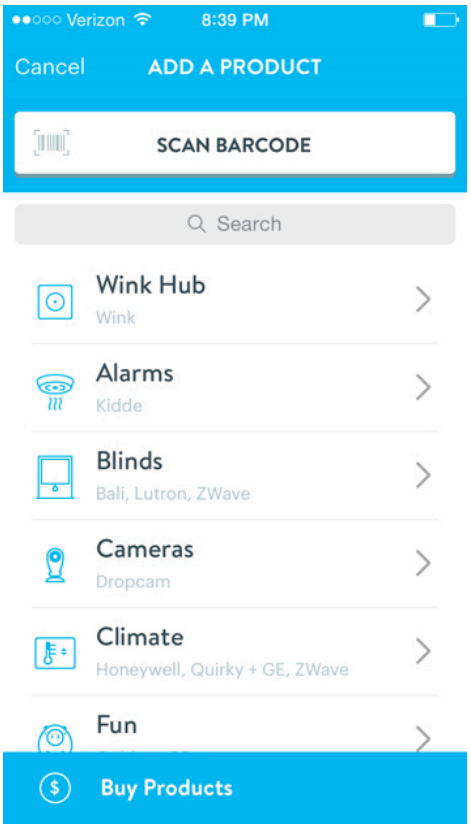

Рисунок 1.4. Приклад інтерфейсу системи «Wink Hub»

Архітектура Wink Hub орієнтована на забезпечення високої сумісності та гнучкості. Вона включає в себе центральний хаб, який виступає як міст між розумними пристроями в домі та мобільним додатком Wink, доступним для iOS та Android пристроїв. Цей хаб збирає дані з підключених пристроїв та передає команди від користувачів, отримані через мобільний додаток або інтегровані голосові асистенти, такі як Amazon Alexa або Google Assistant.

Використання Wink Hub дозволяє користувачам створити централізовану систему розумного будинку, яка може бути легко налаштована та керована з одного додатку, незалежно від брендів або технологій пристроїв, що використовуються. Це робить Wink Hub особливо привабливим рішенням для тих, хто шукає універсальне рішення для інтеграції та управління різноманітними розумними пристроями в домашньому середовищі.

Переваги Wink Hub:

– сумісність з різними протоколами. Wink Hub підтримує множину бездротових технологій, включаючи Wi-Fi, Bluetooth, Z-Wave, та Zigbee, що

дозволяє інтегрувати велику кількість розумних пристроїв від різних виробників у єдину систему;

– централізоване управління. Завдяки центральному хабу, користувачі можуть керувати всіма своїми розумними пристроями через один мобільний додаток, що спрощує управління домашнім середовищем;

– голосове керування. Інтеграція з голосовими асистентами, такими як Amazon Alexa та Google Assistant, надає додаткову зручність у керуванні розумними пристроями;

– автоматизація та сценарії. Wink Hub дозволяє створювати автоматизовані сценарії для розумних пристроїв, що покращує зручність використання та ефективність енергоспоживання.

Недоліки Wink Hub:

– залежність від інтернет-з'єднання. Для функціонування системи необхідне стабільне інтернет-з'єднання, що може стати проблемою при його відсутності або нестабільності;

– вартість. Вартість хаба та окремих розумних пристроїв, сумісних з Wink, може бути високою, що збільшує загальні витрати на створення системи розумного будинку;

– обмеження функціоналу додатку. Деякі користувачі відзначають обмеження в функціоналі мобільного додатку Wink, що може ускладнювати деякі аспекти керування системою;

– потреба в постійних оновленнях. Система вимагає регулярних оновлень для підтримки стабільності та безпеки, що може вимагати додаткового часу та уваги з боку користувачів [9].

Отже, Wink Hub представляє собою універсальне рішення для інтеграції та централізованого управління різноманітними розумними пристроями в домі, пропонуючи високий рівень сумісності з широким спектром бездротових технологій. Завдяки здатності до інтеграції з голосовими асистентами та можливості створення автоматизованих сценаріїв, система значно спрощує процес управління розумним будинком, роблячи його більш зручним та ефективним.

Водночас, висока вартість, залежність від інтернет-з'єднання та деякі обмеження функціоналу мобільного додатку можуть стати перешкодами для деяких користувачів. Незважаючи на це, Wink Hub залишається привабливим варіантом для тих, хто шукає гнучке та ефективне рішення для створення інтегрованої системи розумного будинку.

Samsung SmartThings є однією з провідних платформ розумного будинку, яка забезпечує користувачам універсальне рішення для інтеграції та управління різноманітними розумними пристроями в їхньому домогосподарстві. Призначена для забезпечення більшого комфорту, безпеки та енергоефективності, платформа дозволяє користувачам легко контролювати освітлення, термостати, камери безпеки, електронні замки та багато іншого через єдиний мобільний додаток або через інтеграцію з голосовими помічниками. На рис. 1.5 представлено приклад інтерфейсу користувача для роботи із системою.

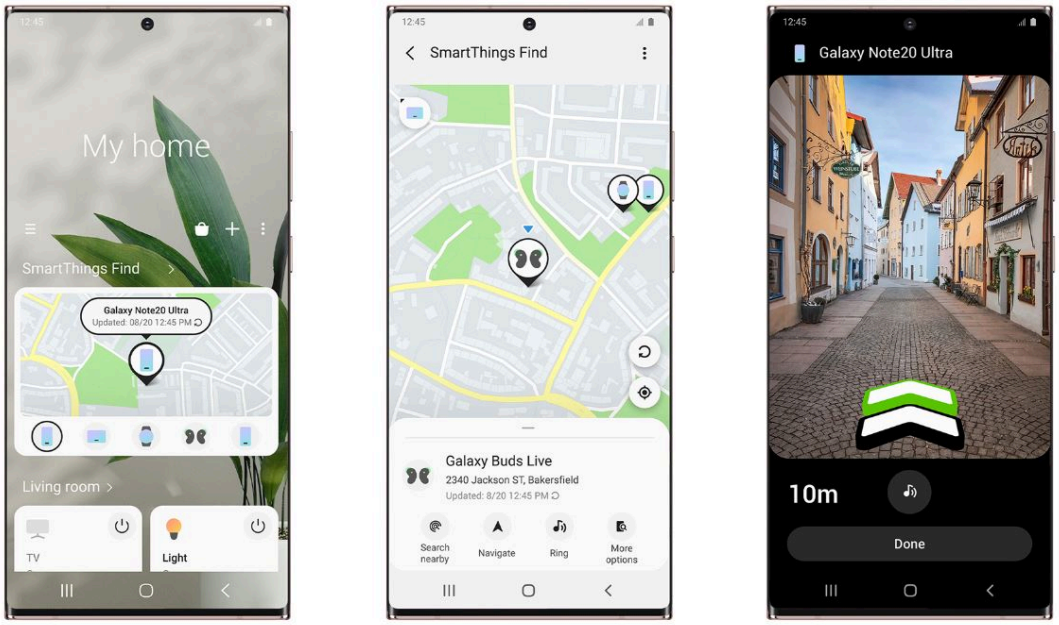

Рисунок 1.5. Приклад інтерфейсу системи «Samsung SmartThings»

Архітектура Samsung SmartThings базується на центральному хабі, який підключається до домашньої мережі Wi-Fi і слугує як основний вузол для зв'язку між всіма підключеними розумними пристроями. Цей хаб спрощує процес підключення нових пристроїв, автоматично виявляючи та інтегруючи їх у систему. Користувачі можуть додавати пристрої різних брендів, що підтримують протоколи

Z-Wave, Zigbee, а також Wi-Fi, забезпечуючи широку сумісність та гнучкість у виборі компонентів для своєї системи розумного будинку.

Система SmartThings не обмежується лише автоматизацією та управлінням пристроями; вона також пропонує розширені можливості для створення сценаріїв та автоматизованих дій. Наприклад, користувачі можуть налаштувати систему таким чином, щоб світло вдома автоматично вимикалося, коли всі члени сім'ї виходять з дому, або щоб термостат знижував температуру вночі для економії енергії.

Переваги Samsung SmartThings:

– широка сумісність. Samsung SmartThings підтримує широкий спектр протоколів зв'язку, включаючи Z-Wave, Zigbee та Wi-Fi, що дозволяє інтегрувати пристрої від різних виробників у єдину систему;

– централізоване управління. Єдиний мобільний додаток для управління всіма розумними пристроями забезпечує зручність та легкість у використанні, дозволяючи користувачам керувати своїм домом з будь-якого місця;

– гнучкість та автоматизація. Платформа пропонує розширені можливості для створення сценаріїв та автоматизованих дій, що дозволяє користувачам налаштувати систему згідно зі своїми потребами та перевагами;

– інтеграція з голосовими помічниками. Сумісність з Amazon Alexa, Google Assistant та іншими голосовими асистентами надає додаткову зручність у керуванні системою за допомогою голосових команд.

Недоліки Samsung SmartThings:

– залежність від інтернету. Для нормальної роботи системи необхідне стабільне інтернет-з'єднання, що може стати проблемою при його відсутності або перебоях;

– складність початкової настройки. Для новачків процес інтеграції та налаштування системи може здатися складним через велику кількість налаштувань та опцій;

– вартість. Початкова вартість хаба та додаткових компонентів може бути досить високою, особливо якщо планується масштабування системи;

– проблеми з оновленнями. Користувачі іноді стикаються з проблемами після оновлення програмного забезпечення, які можуть вимагати додаткових налаштувань або виправлень для повернення системи до нормального функціонування [10].

Отже, Samsung SmartThings є високофункціональною платформою для розумного будинку, яка вирізняється своєю широкою сумісністю з різноманітними протоколами та пристроями, забезпечуючи користувачам гнучкість у створенні інтегрованої системи управління домогосподарством. Завдяки централізованому управлінню через мобільний додаток та можливості інтеграції з популярними голосовими асистентами, система пропонує зручне та інтуїтивно зрозуміле керування розумними пристроями. При цьому, автоматизація та налаштування сценаріїв додатково підвищують комфорт та ефективність використання ресурсів. Незважаючи на високу вартість ініціалізації та потенційні складнощі з налаштуванням та оновленнями, Samsung SmartThings залишається одним з лідерів на ринку рішень для розумного будинку, пропонуючи користувачам надійну та розширювану платформу для автоматизації їхнього домашнього середовища.

1.6 Визначення цілей дослідження та постановка задачі

Основною метою даної кваліфікаційної роботи є створення інтегрованої системи розумного будинку, що об'єднує в собі передові технології для забезпечення безпеки, комфорту та енергоефективності житлового простору. Ця система має на меті вдосконалити існуючі рішення, пропонуючи комплексний захист від різноманітних загроз, а також відповідати сучасним вимогам користувачів до розумного дому. Для реалізації цієї мети вирішення наступних завдань є критично важливим:

– детальний аналіз потреб користувачів і виявлення потенційних загроз для розумних будинків є фундаментом для розробки ефективної системи;

– огляд наявних на ринку систем розумного будинку дозволить виявити їхні слабкі сторони та переваги;

– визначення ключових недоліків існуючих систем і встановлення потреби в нових підходах є необхідним для покращення функціональності та ефективності;

– розробка концепції інтелектуальної системи , яка відповідає сучасним стандартам;

– розробка алгоритмів та протоколів для забезпечення безперебійної взаємодії між компонентами системи є ключовим для її стабільності та надійності;

– реалізація та тестування прототипу системи дозволить перевірити її ефективність у реальних умовах.

Розроблена інтелектуальна система безпеки розумного будинку повинна бути гнучкою та адаптивною, забезпечувати комплексний захист від різних типів загроз, гарантувати високий рівень кібербезпеки та приватності даних користувачів. Використання інтелектуальних алгоритмів та автоматизація процесів значно підвищать ефективність системи. Важливим аспектом є також зручність управління та моніторингу, що забезпечується через інтуїтивно зрозумілий інтерфейс. Енергоефективність і сталість роботи системи, а також її відповідність сучасним стандартам та регулятивним вимогам, робить її не лише інноваційною, але й відповідальною до потреб сучасного суспільства та довкілля.

2 ВИБІР ТЕХНОЛОГІЙ ТА ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ РОЗУМНОГО БУДИНКУ

2.1 Аналіз методів штучного інтелекту для розробки інтелектуальної системи розумного будинку

Використання технологій машинного навчання надає значні переваги для створення вдосконалених рішень. Важливо сконцентруватися на виборі та застосуванні спеціалізованих методів оптимізації та обробки даних, здатних ефективно підсилювати дієвість та точність моделей. В цьому контексті алгоритми, такі як Бройден-Флетчер-Гольдфарб-Шанно (БФГШ), стохастичний градієнтний спуск (СГС) і метод випадкового пошуку (МВП), відіграють ключову роль у підвищенні продуктивності систем машинного навчання.

Метод Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ) є важливим ітеративним алгоритмом у сфері нелінійної оптимізації, призначений для знаходження локальних мінімумів диференційовних функцій без обмежень. Цей метод відноситься до квазі-Ньютонівських методів, які апроксимують обернену матрицю Гессіана, використовуючи інформацію з градієнтів, тим самим уникаючи необхідності обчислення других похідних. Ця особливість робить БФГШ особливо корисним для широкого спектра задач, де пряме обчислення Гессіана є недоцільним через високу обчислювальну складність.

БФГШ застосовується в різних галузях, включаючи машинне навчання, оптимальне керування, фінансове моделювання та багато інших областей, де потрібно ефективно знаходити оптимуми складних функцій. Його популярність обумовлена високою ефективністю та здатністю дошукуватися надійних результатів у задачах з великою кількістю змінних [11].

У контексті електронних систем розумного будинку, метод БФГШ може відігравати ключову роль у оптимізації роботи систем. Наприклад, його можна використовувати для автоматичного регулювання параметрів систем опалення, вентиляції, освітлення, щоб забезпечити максимальну енергоефективність та комфорт для мешканців. Використання БФГШ дозволяє знаходити оптимальні налаштування на основі великої кількості вхідних даних — таких як температура

зовні та всередині будинку, рівень освітленості, присутність людей у приміщенні — з метою мінімізації споживання ресурсів, забезпечуючи при цьому високий рівень комфорту.

Алгоритм БФГШ ефективно збалансовує швидкість збіжності та точність оновлень, що робить його дуже корисним для оптимізації складних функцій втрат у машинному навчанні (рис. 2.1).

Рисунок 2.1. Алгоритм методу БФГШ

Процес роботи методу БФГШ розпочинається з визначення стартової позиції у просторі пошуку та ініціації апроксимації оберненої матриці Гессіана, зазвичай за допомогою ідентичної матриці. На кожному етапі виконання метод визначає градієнт цільової функції в поточній точці, що вказує на напрямок її найінтенсивнішого збільшення. Використовуючи апроксимацію оберненої матриці Гессіана, метод встановлює напрям, по якому слід рухатися, щоб мінімізувати цільову функцію.

Оновлення параметрів моделі проводиться шляхом переміщення поточної точки у просторі параметрів в напрямку, протилежному градієнту. Метод БФГШ коригує апроксимацію оберненої матриці Гессіана після кожного оновлення параметрів, виходячи з останніх розрахунків градієнта та змін у параметрах.

Ітерації тривають, доки не буде досягнуто умов зупинки, які можуть включати незначні зміни в цільовій функції або вичерпання ліміту ітерацій. Цей підхід дозволяє методу ефективно зближуватися до оптимальної точки, враховуючи не тільки напрямок зменшення цільової функції, але й її кривизну в даній точці.

Переваги методу БФГШ:

– ефективність у великих розмірах задач. Метод ефективно використовує інформацію про градієнт для апроксимації оберненої матриці Гессіана, що робить його добре пристосованим до оптимізації задач з великою кількістю змінних;

– відсутність необхідності обчислення других похідних. На відміну від методів, що вимагають безпосереднього обчислення Гессіана, BFGS апроксимує його інверсію, значно спрощуючи обчислення та знижуючи обчислювальні витрати;

– самоналаштування. Метод здатний адаптуватися до локальних особливостей поверхні оптимізації, вибудовуючи напрямок пошуку, який враховує не тільки градієнт, але й кривизну функції, що збільшує швидкість збіжності.

Недоліки методу БФГШ:

– пам'ять. Для великих задач використання BFGS може бути обмежене через високі вимоги до пам'яті, оскільки потребує зберігання матриць розміром n×n, де n - кількість змінних;

– обмежена збіжність. Хоча БФГШ показує високу ефективність на широкому класі задач, існують випадки, коли метод може не знайти оптимальне рішення, особливо у присутності сильних обмежень або при оптимізації недиференційовних функцій;

– чутливість до вибору початкової точки. Як і багато інших ітеративних методів оптимізації, результат роботи БФГШ може суттєво залежати від вибору початкової точки, що може призвести до збіжності до локального, а не глобального мінімуму [12].

Отже, метод Бройдена-Флетчера-Гольдфарба-Шанно представляє собою потужний інструмент для розв'язання задач нелінійної оптимізації, пропонуючи хорошу збіжність і ефективність, особливо у випадках з великою кількістю змінних. Однак, враховуючи вимоги до обчислювальних ресурсів та пам'яті, а також потенційну чутливість до вибору початкових умов, його застосування вимагає ретельного аналізу конкретної задачі та умов її розв'язання. У комплексі з іншими методами, БФГШ може стати ключовою частиною інструментарію для оптимізації в широкому спектрі прикладних досліджень.

Метод стохастичного градієнтного спуску є варіацією традиційного градієнтного спуску, який використовується для оптимізації функцій втрат у задачах машинного навчання та глибинного навчання. Відмінність СГС полягає в оновленні параметрів моделі на основі градієнтів, обчислених за одним або невеликою партією прикладів, замість використання всього набору даних. Це робить метод значно швидшим та ефективнішим з точки зору обчислень, особливо у великих наборах даних.

СГС широко застосовується у машинному навчанні для тренування моделей глибинного навчання, зокрема, нейронних мереж. Його популярність обумовлена здатністю швидко досягати гарних результатів, а також здатністю ефективно масштабуватися на великі обсяги даних.

У контексті електронних систем розумного будинку, СГС може бути використаний для оптимізації різноманітних задач, від автоматичного регулювання освітлення та клімат-контролю до більш складних систем, таких як розпізнавання мови або аналіз поведінки користувачів для підвищення ефективності та комфорту проживання. Наприклад, використання СГС для тренування моделей, які прогнозують звички споживання енергії на основі історичних даних, може допомогти в реалізації енергоефективних стратегій управління ресурсами в будинку.

Схема алгоритму роботи даного методу представлена на рис. 2.2.

Рисунок 2.2. Алгоритм методу СГС

Процедура застосування методу стохастичного градієнтного спуску розпочинається з первинного встановлення параметрів моделі, що часто відбувається за допомогою випадкового вибору або згідно з певними заздалегідь встановленими критеріями. На наступному етапі, в ході кожної ітерації, вибірка одного екземпляра даних або невеликої групи даних (міні-батч) проводиться випадковим образом, що допомагає досягти балансу між швидкістю обчислень та стабільністю процесу оновлення.

Для вибраного тренувального прикладу наступним кроком є розрахунок градієнта цільової функції втрат на основі поточних параметрів моделі. Градієнт вказує на найбільш стрімке збільшення функції втрат. Виходячи з цього градієнта, параметри моделі коригуються у зворотньому напрямку до градієнта, віднімаючи з поточних параметрів добуток швидкості навчання на градієнт. Цикл оновлення параметрів продовжується через кожну ітерацію до виконання визначених умов зупинки, до яких можуть належати досягнення встановленої кількості ітерацій, незначні зміни в значенні функції втрат або досягнення бажаного рівня точності моделі. Такий підхід дозволяє моделі ефективно адаптуватися та вдосконалюватися на основі навчальних даних, зменшуючи втрати та підвищуючи загальну продуктивність.

Переваги методу СГС:

– швидкість обчислень. Метод значно прискорює процес навчання моделей машинного навчання, оскільки оновлення параметрів моделі відбувається після обробки кожного тренувального прикладу або міні-пакету, не вимагаючи обчислення градієнту по всьому набору даних;

– масштабованість. Метод ефективно працює з великими обсягами даних, роблячи його ідеально підходящим для задач глибинного навчання та обробки великих датасетів, де повний прохід по датасету є обчислювально затратним;

– здатність до виходу з локальних мінімумів. Варіативність оновлень, характерна для СГС через використання окремих прикладів або міні-пакетів, може допомогти алгоритму вийти з локальних мінімумів, збільшуючи шанси знаходження більш оптимального глобального мінімуму.

Недоліки методу стохастичного градієнтного спуску:

– варіативність оновлень. Хоча здатність виходу з локальних мінімумів є перевагою, висока варіативність оновлень також може призводити до нестабільності процесу навчання і збільшувати час, необхідний для конвергенції;

– залежність від гіперпараметрів. Ефективність СГС сильно залежить від правильного вибору гіперпараметрів, таких як розмір міні-пакету та швидкість навчання, що може вимагати додаткових зусиль для тонкого налаштування;

– ризик не досягнення глобального мінімуму. У деяких випадках СГС може конвергувати до неоптимальних рішень, особливо у задачах з складними ландшафтами функцій втрат, де існують численні локальні мінімуми [13].

Отже, метод СГС пропонує потужний інструмент для оптимізації великих наборів даних, забезпечуючи швидке навчання та здатність ефективно масштабуватися. Однак, його використання може бути ускладнене високою варіативністю оновлень та чутливістю до вибору гіперпараметрів. Попри це, СГС залишається одним з найбільш популярних методів у машинному навчанні завдяки своїй гнучкості та ефективності у широкому спектрі задач, від глибинного навчання до складних оптимізаційних проблем.

Метод випадкового пошуку — це оптимізаційний алгоритм, який використовує випадковість для знаходження оптимальних рішень у многовимірних просторах. Він не залежить від градієнта функції, що робить його придатним для задач, де градієнт важко обчислити або він не існує. МВП простий у реалізації та здатний досліджувати широкі області пошукового простору, що робить його корисним для глобальної оптимізації.

Метод широко використовується у різних сферах, включаючи інженерію, наукові дослідження та машинне навчання, зокрема для оптимізації параметрів моделей або систем, де інші методи можуть зіткнутися з труднощами через недиференційовність або складність ландшафту функції.

У контексті електронних систем розумного будинку, МВП може бути використаний для автоматизації та оптимізації різноманітних систем та процесів. Наприклад, метод може допомогти в оптимізації розподілу енергії, налаштуванні параметрів системи опалення, вентиляції та кондиціювання повітря для максимальної ефективності та комфорту, або у виборі оптимальних стратегій для розумного освітлення, враховуючи звички користувачів та енергетичні тарифи. МВП дозволяє експериментувати з різними конфігураціями та налаштуваннями, використовуючи випадковий вибір для ідентифікації найкращих можливих рішень, що забезпечують оптимальне співвідношення між продуктивністю, економією та комфортом.

Метод випадкового пошуку ефективний завдяки своїй здатності охопити широкий простір пошуку і знаходити задовільні рішення, часто з меншими обчислювальними витратами, ніж більш систематизовані методи, такі як сітковий пошук, його алгоритм роботи зображено на рис. 2.3.

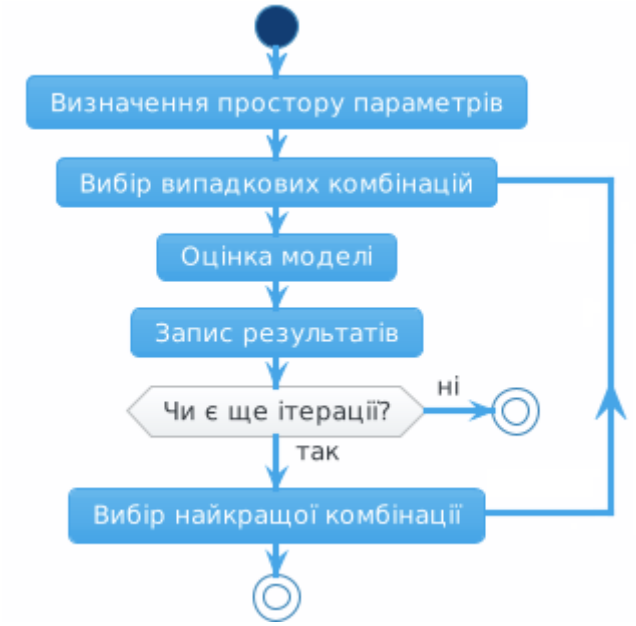

Рисунок 2.3. Алгоритм методу МВП

Процедура методу випадкового пошуку для налаштування гіперпараметрів у машинному навчанні ініціюється із встановлення діапазонів для кожного з гіперпараметрів, що підлягають оптимізації. Ці параметри можуть охоплювати такі аспекти, як розмір пакету даних, кількість шарів у нейронній мережі, або вид активаційної функції. На кожному етапі алгоритму випадково генеруються комбінації цих параметрів зазначеного простору, виходячи з їх дозволених діапазонів.

Для кожної сформованої комбінації параметрів проводиться тренування моделі та оцінювання її ефективності з використанням вибраної метрики оцінки, наприклад, частоти помилок при класифікації. Результати цієї перевірки фіксуються для подальшого аналізу. Цей цикл вибору випадкових наборів параметрів та оцінки ефективності моделі повторюється до досягнення заданого числа ітерацій або використання всіх доступних ресурсів.

Завершивши процес, серед оцінених конфігурацій вибирається оптимальна, яка продемонструвала кращий результат. Метод випадкового пошуку дозволяє швидко проходити через велику кількість можливих налаштувань моделі, часто виявляючи ефективні рішення з меншим обсягом витрачених на обчислення ресурсів порівняно з методами, що вимагають більшої систематичності у пошуку оптимальних параметрів.

Переваги методу випадкового пошуку:

– простота реалізації. Метод випадкового пошуку не вимагає складного алгоритму або глибокого математичного аналізу для своєї реалізації, що робить його легко застосовним до широкого спектру задач оптимізації;

– ефективність у високовимірних просторах. Завдяки своїй невибагливості до форми простору пошуку, метод випадкового пошуку здатен ефективно працювати в умовах, де інші методи оптимізації стикаються з труднощами через прокляття розмірності;

– можливість виявлення глобального оптимуму. В теорії, з нескінченним часом та кількістю ітерацій, метод випадкового пошуку має потенціал знайти глобальний оптимум, оскільки випадковість пошуку охоплює весь простір параметрів.

Недоліки методу випадкового пошуку:

– низька ефективність для вузькоспеціалізованих задач. У задачах, де оптимум знаходиться в дуже маленькій області простору пошуку, метод випадкового пошуку може виявитися надмірно витратним на час або навіть неефективним;

– залежність від кількості ітерацій. Ефективність методу сильно залежить від кількості проведених ітерацій. Для досягнення високої точності може знадобитися величезна кількість спроб, що призводить до збільшення обчислювальних витрат;

– відсутність гарантії знаходження оптимуму за обмежений час. На відміну від більш цілеспрямованих методів, метод випадкового пошуку не надає гарантій знаходження оптимального рішення за певний, заздалегідь визначений час [14].

Отже, метод випадкового пошуку пропонує простий та універсальний підхід до задач оптимізації, особливо ефективний у високовимірних просторах та для випадків, коли інші методи виявляються недостатньо гнучкими. Проте, його основні недоліки — низька швидкість конвергенції та великі обчислювальні витрати для досягнення високої точності — роблять його менш придатним для

задач, де вимагається швидке знаходження точного рішення або де оптимум зосереджений в малій області. Вибір методу випадкового пошуку як інструменту оптимізації повинен ґрунтуватися на ретельному аналізі специфіки задачі та доступних обчислювальних ресурсів.

У табл. 2.1 проведено порівняльний аналіз розглянутих методів машинного навчання.

Таблиця 2.1 – Порівняльний аналіз методів машинного навчання

Характеристика	БФГШ	СГС	МВП
Точність апроксимації оптимуму	Висока	Середня	Низька
Швидкість збіжності в малих/середніх датасетах	Швидка	Залежить від датасету	Повільна
Ефективність використання даних	Висока	Залежить від розміру міні-пакету	Низька
Вимоги до обчислювальних ресурсів	Середні	Низькі	Низькі
Потреба в налаштуванні гіперпараметрів	Низька	Висока	Низька
Здатність до адаптації під кривизну функції	Висока	Низька	Низька

Вибір методу Бройдена-Флетчера-Гольдфарба-Шанно для розробки системи розумного будинку можна обґрунтувати кількома ключовими аспектами, що вказують на його ефективність та придатність до цієї задачі:

– ефективна оптимізація параметрів. БФГШ є потужним інструментом для оптимізації параметрів, особливо в умовах, коли потрібно точно налаштувати параметри для максимальної ефективності системи розумного будинку. Метод дозволяє точно та ефективно знаходити оптимальні налаштування системи управління енергоспоживанням, освітленням, клімат-контролем та іншими елементами розумного будинку, забезпечуючи оптимальне співвідношення між комфортом мешканців та енергоефективністю;

– висока швидкість збіжності. БФГШ забезпечує швидку збіжність до оптимального рішення, що є критично важливим для систем розумного будинку, де необхідно швидко реагувати на зміну умов навколишнього середовища та

потреб користувачів. Це дозволяє системі розумного будинку бути адаптивною та ефективною в реальному часі;

– адаптація до локальних особливостей задачі. Завдяки здатності БФГШ адаптуватися до кривизни оптимізованої функції, цей метод ідеально підходить для задач, де поведінка системи може суттєво змінюватися в залежності від різних факторів, таких як час доби, погодні умови, або присутність людей у будинку;

– економія обчислювальних ресурсів. Незважаючи на деякі вимоги до пам'яті для зберігання апроксимації оберненої матриці Гессіана, загалом БФГШ може бути ефективніше за інші методи, які вимагають численних ітерацій або обчислення складних похідних. Це робить BFGS привабливим вибором для вбудованих систем розумного будинку, де обчислювальні ресурси можуть бути обмежені.

Отже, використання методу БФГШ для розробки системи розумного будинку дозволяє досягнути високої ефективності та адаптивності системи, забезпечуючи швидку оптимізацію параметрів з мінімальними витратами обчислювальних ресурсів. Це робить BFGS цінним інструментом для створення інтелектуальних систем, здатних до самоналаштування та оптимізації в реальному часі для забезпечення максимального комфорту та ефективності.

2.2 Обґрунтування вибору бібліотеки машинного навчання ML .NET

ML.NET представляє собою відкриту бібліотеку для реалізації машинного навчання, створену Microsoft спеціально для .NET платформи. Ця бібліотека об'єднує інструменти для розробки, навчання та впровадження моделей машинного навчання у програми та сервіси на базі .NET, дозволяючи .NET розробникам інтегрувати функціонал машинного навчання безпосередньо в їхні додатки. Таким чином, уникається необхідність використання зовнішніх мов програмування чи бібліотек.

Бібліотека ML.NET обслуговує широке коло завдань машинного навчання, таких як класифікація, регресія, кластеризація, створення рекомендаційних систем

тощо, пропонуючи розробникам різноманітні алгоритми для цих цілей, оптимізовані з точки зору продуктивності та масштабованості [15].

Однією з ключових особливостей ML.NET є підтримка трансферного навчання, що дозволяє використовувати заздалегідь натреновані моделі з інших відомих бібліотек, наприклад TensorFlow. Це значно спрощує та прискорює розробку і впровадження рішень, заснованих на машинному навчанні, в системах розумного будинку, фінансовому аналізі, здоров'я та багатьох інших областях.

Застосування ML.NET для створення інтелектуальних систем, зокрема в машинному навчанні, має суттєві переваги, що робить цю бібліотеку від Microsoft ідеальним вибором для проектів, що базуються на платформі .NET:

– висока продуктивність. ML.NET оптимізована для задоволення вимог промислових застосунків, забезпечуючи швидке тренування та низьку затримку при застосуванні моделей у реальному часі, що є ключовим для ефективної роботи систем;

– інтеграція з .NET. Бібліотека забезпечує плавне вбудовування машинного навчання в існуючі .NET додатки та системи, дозволяючи розробникам легко інкорпорувати розширені аналітичні можливості без потреби звертатися до інших мов програмування чи інструментів;

– різноманітність алгоритмів. Представлення великої кількості алгоритмів для різних задач машинного навчання в ML.NET дозволяє вибирати найбільш підходящі рішення для специфіки кожного проекту, від простих до складних задач;

– підтримка трансферного навчання. ML.NET спрощує процес використання вже існуючих моделей, зокрема тих, що були розроблені з використанням TensorFlow, дозволяючи розробникам швидко адаптувати ці моделі під власні потреби з мінімальними зусиллями;

– сумісність та розширюваність. Відкритий код та висока сумісність з іншими бібліотеками та платформами робить ML.NET гнучкою базою для розширення можливостей інтелектуальних систем, забезпечуючи легку інтеграцію з іншими інструментами аналізу даних;

– доступність ресурсів для навчання та підтримка спільноти. Багата документація, приклади коду та активна спільнота користувачів сприяють швидкому освоєнню бібліотеки та ефективному вирішенню виникаючих питань [16].

Враховуючи ці аспекти, ML.NET стає виправданим вибором для розробки інтелектуальної системи розумного будинку, де важливі є швидкість розробки, продуктивність, інтеграція та можливість використання передових технологій машинного навчання.

2.3 Формалізація задачі та розробка математичної моделі для інтелектуальної системи розумного будинку

У цьому підрозділі розглянуто розробку математичної моделі для системи розумного будинку, що здатна виявляти небезпечні або несанкціоновані дії. Модель побудована на основі алгоритму оптимізації, аналогічного методу БФГШ, відомого своєю ефективністю у вирішенні задач бінарної класифікації, особливо в умовах роботи з об'ємними даними. Цей комплексний підхід включає збір та аналітичний аналіз даних, ідентифікацію ключових ознак, що найкраще описують потенційно небезпечні дії в контексті розумного будинку, та використання БФГШ для точного налаштування параметрів класифікатора. Такий алгоритм забезпечує високу ефективність у виявленні складних взірців поведінки, що можуть вказувати на несанкціонований доступ або інші види порушень.

Задача, що стоїть перед нами, представляє собою задачу бінарної класифікації, де ми маємо набір даних (X,Y), в якому $X \in R^{n \times m}$ є матрицею ознак з n прикладами та mm ознаками, а $Y \in \{0,1\}^n$ є вектором міток, що відображає наявність чи відсутність небезпечної дії. У контексті розумного будинку, m ознаки можуть включати час початку активності, місцезнаходження, час доби, взаємодію з об'єктами, постаті жителя, тривалість активності тощо. Використовуючи ці дані, модель намагається визначити, чи відповідає дана активність нормальному використанню системи розумного будинку, або ж вона може свідчити про потенційне вторгнення або інші види ризикованих дій.

Цільова функція для моделі розумного будинку може бути описана за допомогою логістичної регресії таким чином:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)} \log log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right],$$

(3.1)

де, $h_\theta(x)$ представляє сигмоїдну функцію, а θ – це вектор параметрів моделі.

Матриця *X* формується з екземплярів класу *SmartHomeData*. Кожний рядок матриці *X* відображає один екземпляр «*SmartHomeData»*, а кожний стовпець відповідає одному з полів (*StartTime, Location, TimeOfDay*, тощо). Тобто, елемент x_{ij} матриці X є значенням *j*-го поля *i*-го екземпляру «*SmartHomeData»*:

$$X = \left(\frac{x_{11}\, x_{12} \ldots x_{1n}}{x_{21}\, x_{22} \ldots x_{2n} \ldots \ldots \ldots \ldots x_{m1}\, x_{m2} \ldots x_{mn}}\right) \qquad (3.2)$$

Вектор *Y* в свою чергу, формується з міток активності кожного екземпляру *SmartHomeData*:

$$Y = \left(\frac{y_1}{y_2 \ldots y_n}\right), \qquad (3.3)$$

де y_i може бути 1 (нормальна активність) або 0 (потенційне вторгнення).

Після тренування моделі, прогнозовані значення $\hat{Y}$ отримуються за допомогою сигмоїдної функції $h_\theta(x)$:

$$\hat{Y} = h_\theta(x) = \frac{1}{1+exp(-X\theta)} \qquad (3.4)$$

Ці прогнози зберігаються у полях екземплярів класу *SmartHomePrediction* в полі «*PredictedActivity*»:

$$SmartHomePred.\,PredictedActivity = \{1, \text{ якщо } \hat{y}_i \geq 0,5\ 0, \text{ якщо } \hat{y}_i < 0,5$$

(3.5)

Таким чином, інтеграція з *SmartHomeData* та *SmartHomePrediction* не лише структурує вхідні та вихідні дані, але й дозволяє ефективно використовувати їх для тренування та прогнозування в моделі, підвищуючи точність ідентифікації нормальних та підозрілих активностей в системі розумного будинку.

2.4 Вибір технологій реалізації системи розумного будинку

У контексті створення та аналізу ефективності інтелектуальних систем для розумного будинку, які побудовані на алгоритмах машинного навчання, критичною є вибір мови програмування. Цей вибір безпосередньо впливає на адаптивність, продуктивність і легкість обслуговування таких систем. У нашому дослідженні ми розглянемо три основні мови програмування: Python, Java та C#, кожна з яких має унікальні переваги та особливості, що роблять їх придатними для реалізації різних аспектів функціональності систем розумного будинку.

Python відзначається своєю простотою і легкістю вивчення, що, разом з великою кількістю доступних бібліотек і модулів, робить її ідеальною для швидкої розробки і прототипування інтелектуальних систем. Її чистий і зрозумілий синтаксис сприяє розробці легкочитаних і легко підтримуваних програм [17].

JavaScript, як динамічна і об'єктно-орієнтована мова, є ключовою для створення інтерактивних користувацьких інтерфейсів, включаючи додатки для керування системами розумного будинку через веб. Його здатність виконуватися безпосередньо у браузері користувача робить JavaScript незамінним для розробки швидких та відгукових веб-додатків.

C#, розроблена Microsoft, забезпечує тісну інтеграцію з платформою .NET, пропонуючи міцну основу для розробки надійних, масштабованих і безпечних застосунків для розумного будинку. Її сильна типізація допомагає уникнути багатьох помилок на етапі компіляції, а обширна стандартна бібліотека спрощує багато аспектів розробки [18].

Кожна з цих мов має інструменти та бібліотеки, що підтримують розробку інтелектуальних систем, включаючи аналіз даних, машинне навчання, і, зокрема, розробку систем розумного будинку. У табл. 2.2 було представлено порівняльний аналіз мов програмування Python, JavaScript і C# за рядом ключових параметрів.

Таблиця 2.2 – Порівняльний аналіз мов програмування

Характеристика	Python	JavaScript	C#
Швидкість виконання	Помірна	Варіюється	Висока
Кросплатформеність	Висока	Висока	Висока (.NET Core)

Підтримка Windows	Добра	Добра	Відмінна
Інтеграція з Microsoft продуктами	Обмежена	Обмежена	Відмінна
Розвиненість IDE	Хороші варіанти	Хороші варіанти	Відмінні варіанти
Підтримка багатопоточності	Обмежена	Обмежена	Відмінна
Перевірка типів	Динамічна	Динамічна	Статична
Використання у розробці ігор	Обмежене	Обмежене	Відмінне (Unity)

Вибір мови програмування C# для створення систем розумного будинку виправдовується через низку важливих переваг, що ця мова пропонує. Вона є частиною екосистеми Microsoft .NET і забезпечує тісну інтеграцію з операційною системою Windows, що спрощує розробку додатків для керування домашніми пристроями та системами, які часто базуються на цій платформі. Завдяки підтримці .NET Core, C# також дозволяє розробляти багатоплатформенні застосунки, розширюючи можливості систем розумного будинку до використання на MacOS та Linux.

Visual Studio як одне з найпотужніших інтегрованих середовищ розробки надає розробникам широкий спектр інструментів для ефективної роботи з кодом, включаючи розширені можливості для дебагінгу, профілювання та управління версіями. Це значно підвищує продуктивність розробників та сприяє створенню надійного програмного забезпечення для розумного будинку.

Додатково, C# має потужну підтримку багатопоточності, що є важливим для розробки систем розумного будинку, здатних ефективно обробляти великі обсяги даних та запитів в реальному часі. Строга статична типізація та потужна система типів мінімізують ризики програмних помилок, підвищуючи загальну надійність та безпеку системи.

Таким чином, використання C# для розробки інтелектуальних систем розумного будинку відкриває широкі можливості для створення ефективних, надійних та багатоплатформенних рішень, які можуть бути легко інтегровані з

іншими продуктами та сервісами Microsoft, а також підтримувати високий рівень інтерактивності та зручності для кінцевих користувачів.

Вибір архітектури для системи розумного будинку є ключовим рішенням, яке безпосередньо впливає на такі критичні аспекти, як продуктивність, масштабованість та зручність подальшої розробки та обслуговування системи. Розглядаючи основні архітектурні підходи, можна оцінити їхні сильні сторони та обмеження у контексті створення інтелектуальних систем управління домом.

Трирівнева архітектура пропонує чітке розділення функцій між рівнями представлення, бізнес-логіки та даних. Таке розділення дозволяє легко модифікувати та масштабувати систему, забезпечуючи модульність коду та відокремленість даних, що сприяє безпеці. У контексті розумного будинку, це може означати легкість додавання нових типів пристроїв або функцій без потреби в глибоких змінах у всій системі.

Мікросервісна архітектура дозволяє створювати систему як набір незалежних, легко масштабованих сервісів, кожен з яких виконує певну роль або функцію в системі розумного будинку. Цей підхід полегшує розподіл ресурсів, дозволяючи зосередитися на оптимізації окремих компонентів системи, а також спрощує впровадження нових технологій або сервісів.

MVC архітектура, зокрема, підходить для розробки інтерфейсів управління системами розумного будинку, де важливе чітке розділення логіки взаємодії з користувачем (Вид), бізнес-логіки системи (Модель) та механізмів взаємодії між ними (Контролер). Це дозволяє розробникам працювати над окремими аспектами системи незалежно, сприяючи гнучкості та ефективності процесу розробки.

Кожен з цих архітектурних підходів має свої переваги для створення комплексних, високопродуктивних та легко адаптивних систем розумного будинку, забезпечуючи розробникам гнучкість у виборі рішень, що найкраще відповідають конкретним вимогам та цілям проекту [19].

У табл. 2.3 представлено детальний порівняльний огляд різних архітектурних шаблонів, де оцінюються їх ключові характеристики та особливості.

Таблиця 2.3 – Порівняльний аналіз архітектурних шаблонів

Характеристика	Трирівнева архітектура	Мікросервісна архітектура	MVC
Рівні	3 (інтерфейс, бізнес-логіка, доступ до даних)	Залежить від компонентів системи	3 (Модель, Вид, Контролер)
Залежність між компонентами	Сильна	Слабка	Середня
Масштабованість	Обмежена	Висока	Середня
Керованість	Легко керується	Складно керується	Легко керується
Загальна складність	Середня	Висока	Середня
Швидкість розробки	Швидка	Повільна	Швидка
Сумісність	Добра	Середня	Добра
Тестування	Легко тестується	Складно тестується	Легко тестується
Виконання	Швидке виконання	Залежить від сервісів	Швидке виконання

Обрання трирівневої архітектури для системи розумного будинку було вмотивовано її численними перевагами, які визначають цю структуру як одну з найбільш популярних у розробці програмного забезпечення. Ця архітектура розподіляє систему на три взаємодіючих, але незалежних шари: представлення, бізнес-логіку та доступ до даних, що спрощує розробку, підтримку та оновлення системи.

Таке розділення дозволяє розробникам концентруватися на специфічних аспектах системи, підвищуючи ефективність розробки через чітку організацію робочого процесу. Незалежність рівнів забезпечує легкість у підтримці, оскільки зміни або оновлення в одному шарі не впливають безпосередньо на інші шари, що спрощує управління системою. Гнучкість, властива трирівневій архітектурі, дозволяє легко адаптувати систему до нових вимог або змін у бізнес-процесах, забезпечуючи довгострокову актуальність та відповідність потребам користувачів.

Окреме тестування кожного рівня, а також їх взаємодії, покращує якість загального продукту, дозволяючи детально виявляти та виправляти помилки на ранніх етапах розробки. Таким чином, вибір трирівневої архітектури для системи

розумного будинку став логічним рішенням, що відповідає вимогам до простоти обслуговування, гнучкості системи та ефективності процесу тестування, забезпечуючи надійну та високопродуктивну платформу для управління інтелектуальним будинком.

2.5 Проектування архітектури системи розумного будинку

Розробка рівня даних для системи управління розумним будинком відіграє ключову роль, оскільки система повинна ефективно керувати великими обсягами даних, отриманих з множини домашніх датчиків та пристроїв. Цей шар не лише забезпечує надійне зберігання інформації про стан будинку, активності користувачів та історичні дані, але й гарантує швидкий доступ до цих даних для аналізу, моніторингу та виконання відповідних дій.

Для ізоляції бізнес-логіки від прямого доступу до бази даних було впроваджено шар доступу до даних (Data Access Layer, DAL), який діє як абстрактний посередник. DAL визначає стандартний набір операцій для зберігання, пошуку, оновлення та видалення даних, тим самим знижуючи залежність бізнес-логіки від конкретної схеми бази даних. Це спрощує процеси рефакторингу та оновлення бази даних, мінімізуючи потребу в змінах до бізнес-логіки системи при модифікації структури даних.

Використання DAL сприяє гнучкості системи розумного будинку, дозволяючи легко масштабувати та адаптувати систему під змінювані потреби користувачів та технологічні оновлення. Наприклад, при додаванні нових типів датчиків або розширенні функціональності системи, DAL дозволяє інтегрувати нові джерела даних без переписування існуючого коду бізнес-логіки [20].

На діаграмі класів, що відображає шар доступу до даних, демонструються ключові методи та інтерфейси для взаємодії з базою даних. Це може включати методи для отримання даних про стан домашніх пристроїв, запису подій безпеки, керування сценаріями автоматизації та інше. Завдяки такій структурі, система розумного будинку стає більш модульною, надійною та легкою в обслуговуванні,

забезпечуючи високий рівень безпеки, комфорту та ефективності для користувачів.

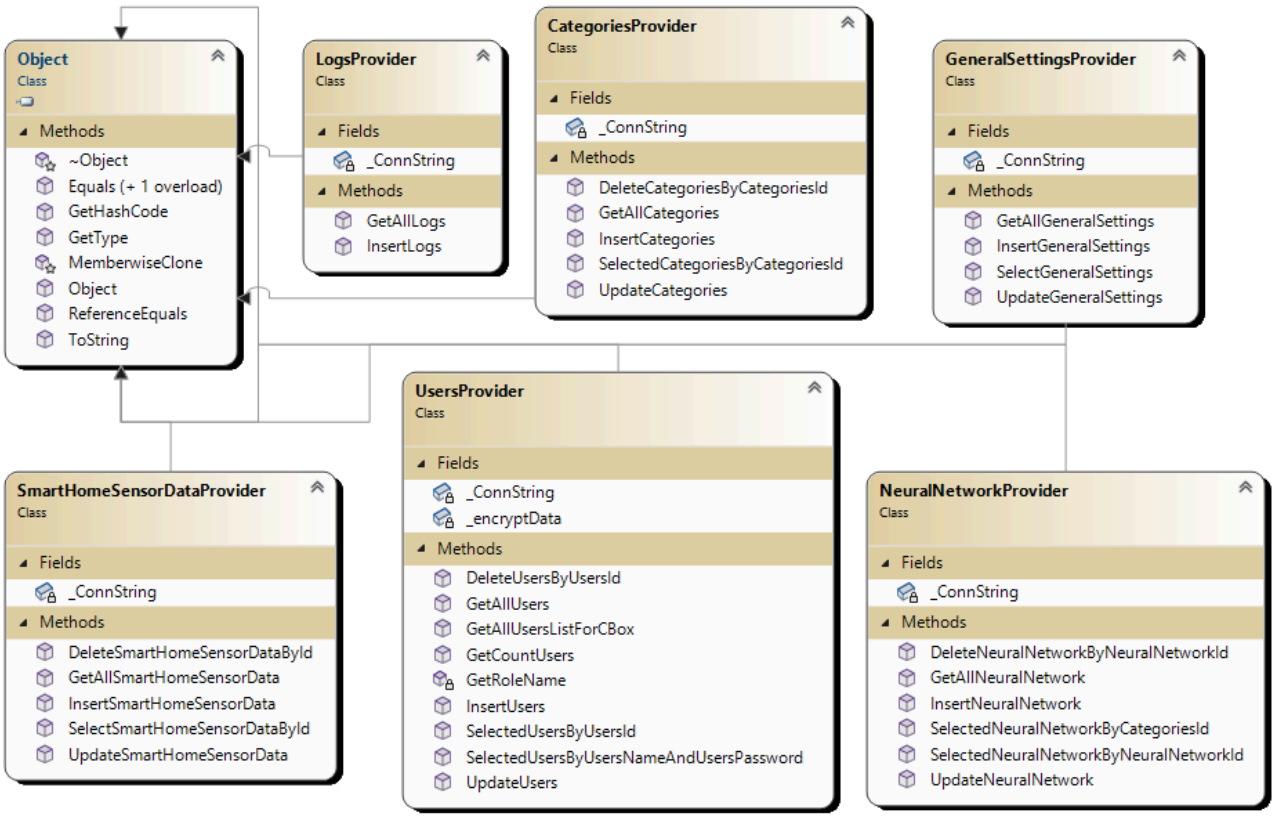

Рисунок 2.4. Діаграма класів рівня даних

Як видно із рис. 2.4, діаграма класів рівня даних складається із 5-ти основних класів:

– клас CategoriesProvider відповідає за управління категоріями, які можуть бути використані для класифікації датчиків, подій або інших елементів у системі. Він дозволяє додавати нові категорії (Create), отримувати список існуючих категорій (Read), оновлювати інформацію про категорії (Update) та видаляти їх (Delete);

– клас GeneralSettingsProvider управляє загальними налаштуваннями системи розумного будинку, такими як параметри безпеки, налаштування освітлення та клімат-контролю. Він забезпечує інтерфейс для збереження та зміни цих налаштувань, дозволяючи системі адаптуватися до потреб користувачів;

– клас LogsProvider веде облік подій системи, забезпечуючи зберігання, перегляд та аналіз логів діяльності користувачів та стану системи. Це дозволяє відслідковувати всі зміни та події для цілей безпеки та обслуговування;

– клас NeuralNetworkProvider керує інтеграцією та використанням нейронних мереж у системі, наприклад, для аналізу даних або розпізнавання шаблонів поведінки. Він включає функції для тренування моделей, їх оцінки та використання готових моделей для прогнозування;

– клас SmartHomeSensorDataProvider відповідає за обробку даних, зібраних з датчиків розумного будинку. Він забезпечує функції для збору, зберігання та аналізу інформації про стан будинку, такої як температура, освітленість, вологість та інші параметри;

– клас UsersProvider управляє інформацією про користувачів системи, включаючи їхні профілі, права доступу та налаштування. Він дозволяє реєструвати нових користувачів, отримувати та оновлювати інформацію про існуючих, а також видаляти користувачів з системи.

У контексті розробки інтерфейсу для системи управління розумним будинком було прийнято рішення виділити інтерактивний рівень користувацького інтерфейсу. Ця частина системи фокусується на розробці інтуїтивно зрозумілих та зручних для користувача інтерфейсів, включаючи різноманітні графічні елементи управління, такі як кнопки, текстові поля, слайдери та візуалізації даних. Ці компоненти відіграють ключову роль у забезпеченні ефективної взаємодії між користувачем та системою, дозволяючи не тільки вводити команди та налаштування, але й отримувати зворотній зв'язок та переглядати аналітичні звіти та прогнози.

Розроблений інтерфейс призначений для спрощення взаємодії користувачів з системою, забезпечуючи легкий доступ до управління пристроями розумного будинку, моніторингу стану датчиків, та аналізу зібраних даних. На рис. 2.5 представлена діаграма класів для рівня користувацького інтерфейсу системи.

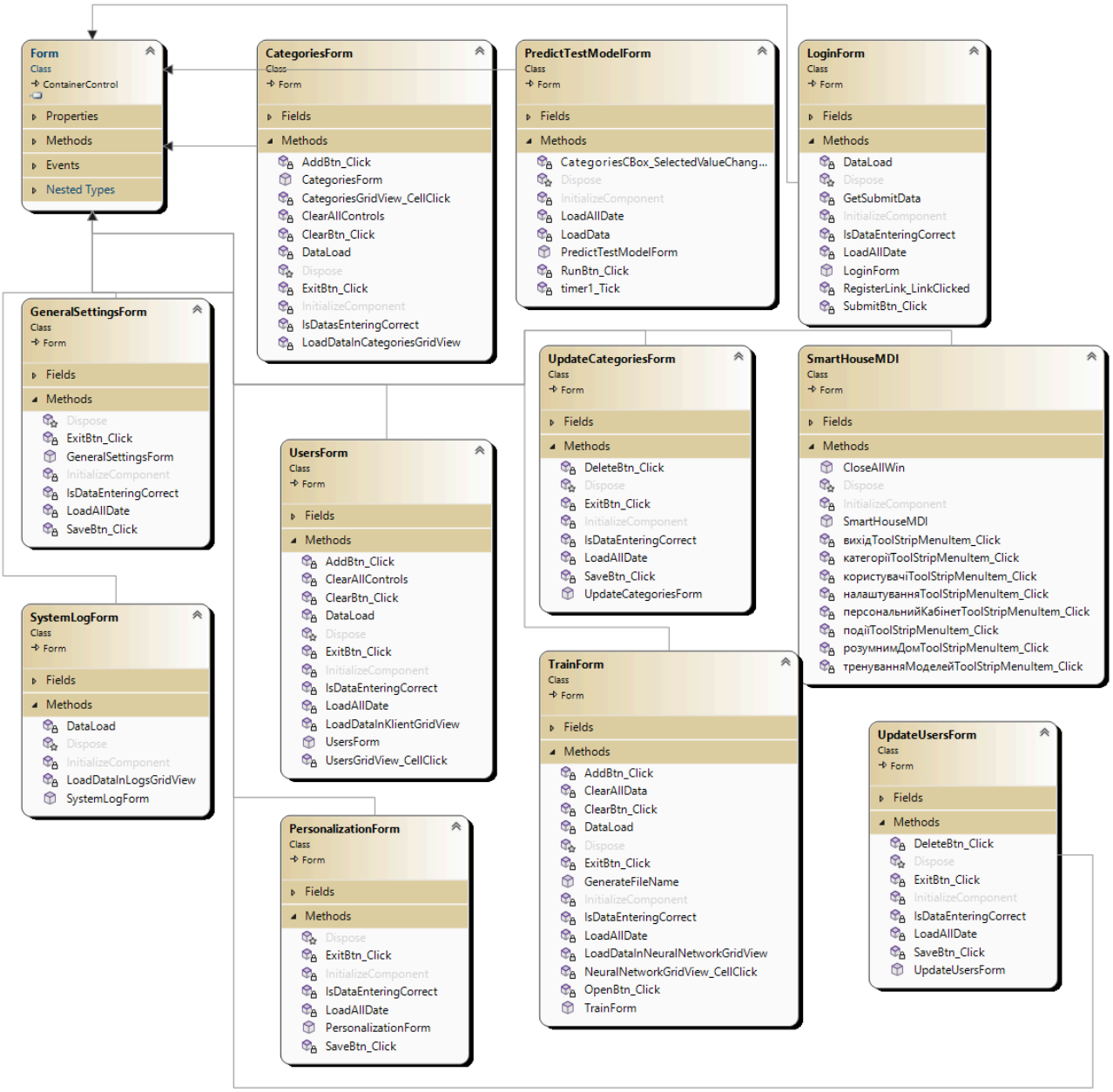

Рисунок 2.5. Діаграма інтерфейсу системи

Діаграма даного рівня складається з десяти класів рівня UI та є похідними від класу Form, тобто мають графічний інтерфейс:

– PredictTestModelForm інтерфейс призначений для прогнозування вторгнень на основі поведінкових шаблонів жителів. Користувач може ввести дані поведінки, і система використовує попередньо натреновані моделі для оцінки ймовірності небажаної діяльності;

– CategoriesForm – форма дозволяє додавати нові категорії машинних моделей, які можуть використовуватися для класифікації різних типів даних або

подій у системі. Це полегшує організацію моделей та їх подальше використання в аналізі даних;

– TrainForm – інтерфейс для тренування моделей машинного навчання, які використовуються для розпізнавання поведінки жителів. Користувач може завантажити набори даних та визначити параметри тренування, щоб оптимізувати моделі для конкретних потреб системи;

– UpdateCategoriesForm – форма для редагування існуючих категорій машинних моделей, що дозволяє користувачам оновлювати назви категорій або їхні описи, забезпечуючи актуальність та релевантність класифікації;

– GeneralSettingsForm – форма налаштувань системи розумного будинку, де користувачі можуть керувати загальними параметрами системи, такими як вибір активних датчиків, налаштування повідомлень та інші основні параметри;

– LoginForm – інтерфейс входу в систему, який забезпечує аутентифікацію користувачів перед доступом до функціоналу системи управління розумним будинком;

– PersonalizationForm – форма персоналізації дозволяє користувачам налаштовувати індивідуальні параметри системи, включаючи інтерфейси користувача, персональні сценарії автоматизації та вибірки даних для аналізу;

– SystemLogForm – інтерфейс, що відображає журнали системи, де користувачі можуть переглядати записи про всі події та дії, здійснені в системі, для моніторингу стану та виявлення потенційних проблем;

– UpdateUsersForm –форма для оновлення інформації про користувачів системи, що дозволяє адміністраторам модифікувати ролі, доступи та особисті дані користувачів;

– UsersForm – інтерфейс управління користувачами, де можна переглядати список користувачів системи, додавати нових користувачів або видаляти існуючих, забезпечуючи ефективне керування доступами до різних частин системи.

Останнім етапом у створенні архітектури системи розумного будинку стала розробка бізнес-логіки, яка слугує основою для виконання всіх ключових процесів

в системі. Цей компонент архітектури інкапсулює в собі правила та алгоритми, необхідні для забезпечення правильної роботи системи, і виступає як основний драйвер внутрішніх операцій. Організація бізнес-логіки у вигляді незалежних модулів дозволяє системі залишатися гнучкою та масштабованою, а також легко адаптуватися до змін у бізнес-вимогах без необхідності глибокого перегляду коду.

В архітектурі системи бізнес-логіка чітко відділена від інших рівнів, таких як інтерфейси користувача та взаємодія з базами даних, дозволяючи розробникам зосередитися на логіці обробки даних. Цей підхід не тільки підвищує якість та підтримуваність коду, але й спрощує інтеграцію бізнес-логіки з різноманітними інтерфейсами користувача та системами зберігання даних, роблячи систему розумного будинку більш відкритою для розширення та інтеграції з новими технологіями.

На діаграмі, що ілюструє цю частину архітектури, можна побачити як бізнес-логіка взаємодіє з іншими компонентами системи, вказуючи на її центральне місце в архітектурі (рис. 2.6). Ця візуалізація надає чітке розуміння структури системи розумного будинку, підкреслюючи важливість бізнес-логіки у забезпеченні її функціональності та взаємодії з користувачами.

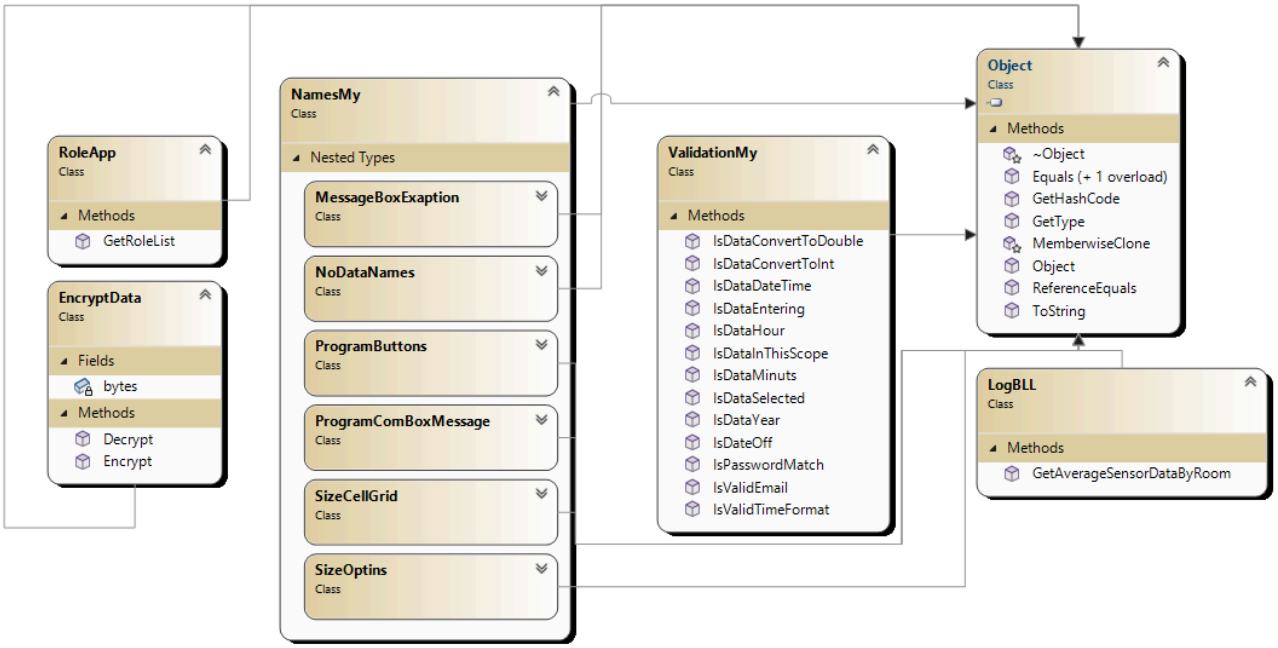

Рисунок 2.6. Діаграма класів бізнес-логіки

У системі захисту на основі мережі LSTM, класи бізнес-логіки відіграють вирішальну роль у застосуванні бізнес-правил і алгоритмів, забезпечуючи безпеку, управління даними та валідацію. Основні класи бізнес-логіки складаються із:

– клас EncryptData відповідає за шифрування даних у системі розумного будинку. Він використовується для забезпечення конфіденційності важливої інформації, такої як персональні дані користувачів, паролі або інформація про фінансові транзакції. Клас може використовувати різні алгоритми шифрування для забезпечення надійного захисту даних;

– клас NamesMy містить утиліти або допоміжні методи для роботи з іменами або ідентифікаторами, що використовуються в системі. Може включати методи для генерації унікальних імен для нових пристроїв, користувачів або сеансів. Також може надавати функціонал для форматування чи валідації імен;

– клас RoleApp керує ролями та дозволами в системі розумного будинку. Він визначає різні ролі користувачів (наприклад, адміністратор, користувач, гість) та їхні права доступу до різних функцій системи. Цей клас важливий для реалізації системи контролю доступу та забезпечення безпеки;

– клас ValidationMy забезпечує валідацію вхідних даних в системі. Він може перевіряти правильність електронних адрес, паролів, ідентифікаторів пристроїв або інших даних, введених користувачами. Використання цього класу допомагає запобігти помилкам в даних та забезпечити їхню відповідність встановленим критеріям;

– клас LogBLL відповідає за ведення журналів подій у системі. Цей клас може реєструвати дії користувачів, системні помилки, інформацію про пристрої та інші важливі події. Використання журналу дозволяє здійснювати моніторинг стану системи, аналізувати проблеми та оптимізувати її роботу.

Кожен з цих класів виконує специфічну роль у загальній архітектурі системи розумного будинку, сприяючи її ефективності, безпеці та зручності використання.

2.6 Розробка основних алгоритмів для забезпечення безпеки системи «Розумний дім»

У рамках дипломної роботи для забезпечення безпеки системи «Розумний дім», особлива увага приділяється створенню надійних механізмів захисту. Сучасні виклики, що стоять перед системами автоматизації домогосподарств, вимагають не лише фізичного захисту, але й захисту інформаційного, що робить актуальним питання розробки ефективних алгоритмів безпеки. Значення даної роботи полягає в тому, що вона спрямована на вирішення проблеми захисту даних та управління доступом у системах «Розумний дім», використовуючи сучасні досягнення в галузі машинного навчання та криптографії.

Основна мета розробки полягає в створенні механізмів, здатних протистояти зовнішнім та внутрішнім загрозам, аналізувати поведінкові патерни для виявлення незвичайної активності та захистити персональні дані користувачів від несанкціонованого доступу. Такий підхід передбачає розробку комплексної системи безпеки, що інтегрується з основними функціями «Розумного дому», надаючи користувачам не тільки комфорт та зручність, але й високий рівень захисту.

На рис. 2.7 показано блок-схему, яка демонструє процес навчання та зберігання моделі у системі.

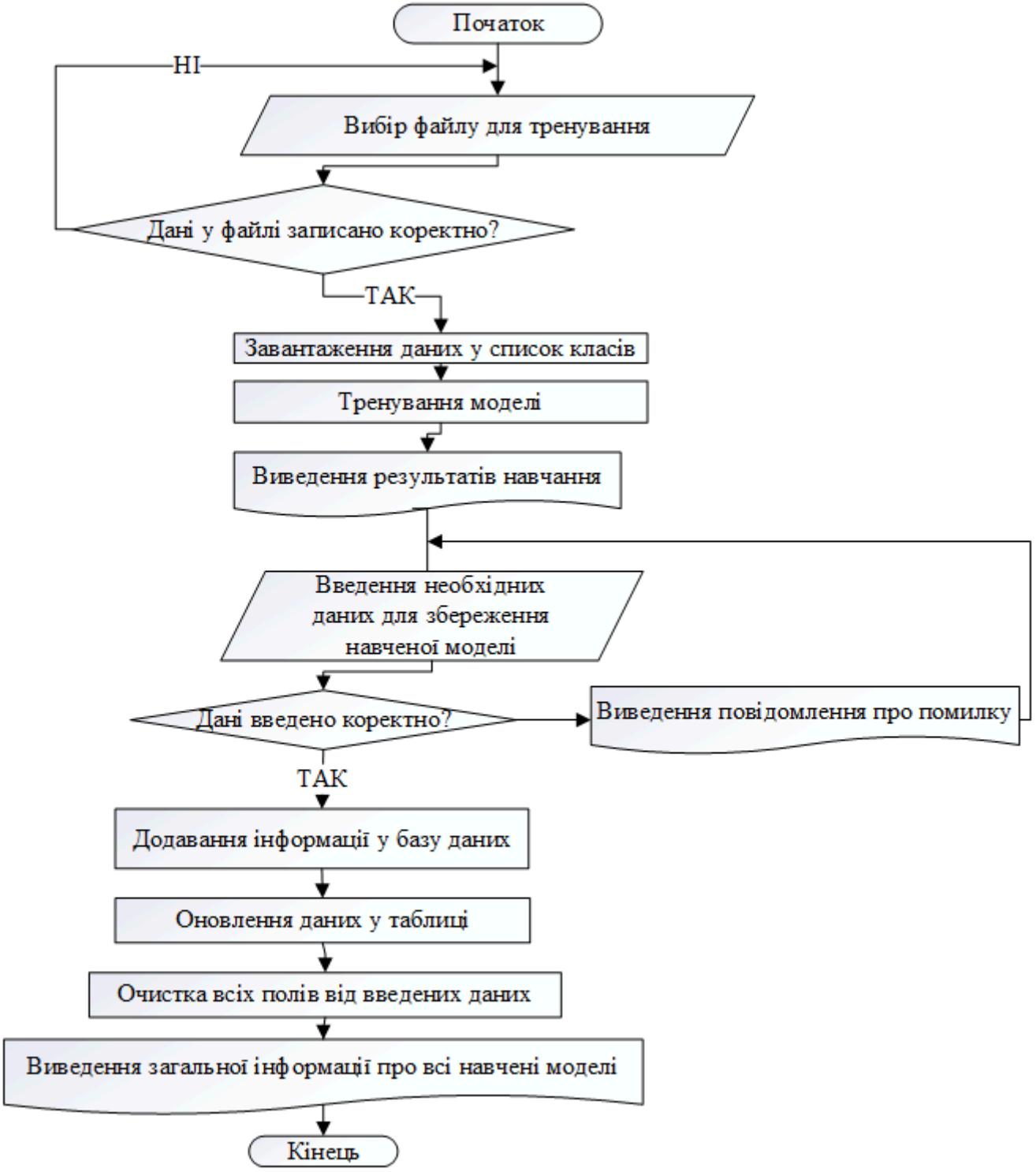

Рисунок 2.7. Блок-схема алгоритму навчання та зберігання даних моделі

Навчання моделі починається з вибору файлу, який включає в себе необхідні дані для тренування. Після валідації цих даних на відповідність вимогам, вони завантажуються для дальшого аналізу. Використовуючи ці дані, модель машинного навчання піддається тренуванню, із подальшим відображенням результатів для аналізу ефективності. Коли модель успішно навчена, інформація

про неї заноситься до системи після перевірки на адекватність. Дані в таблиці, що містить перелік моделей, оновлюються, а елементи форми очищаються, готуючи інтерфейс до нового етапу тренування. Завершальний етап передбачає відображення детальної інформації про всі треновані моделі.

Рис. 2.8 відображає блок-схему алгоритму роботи навченої моделі у реальному часі.

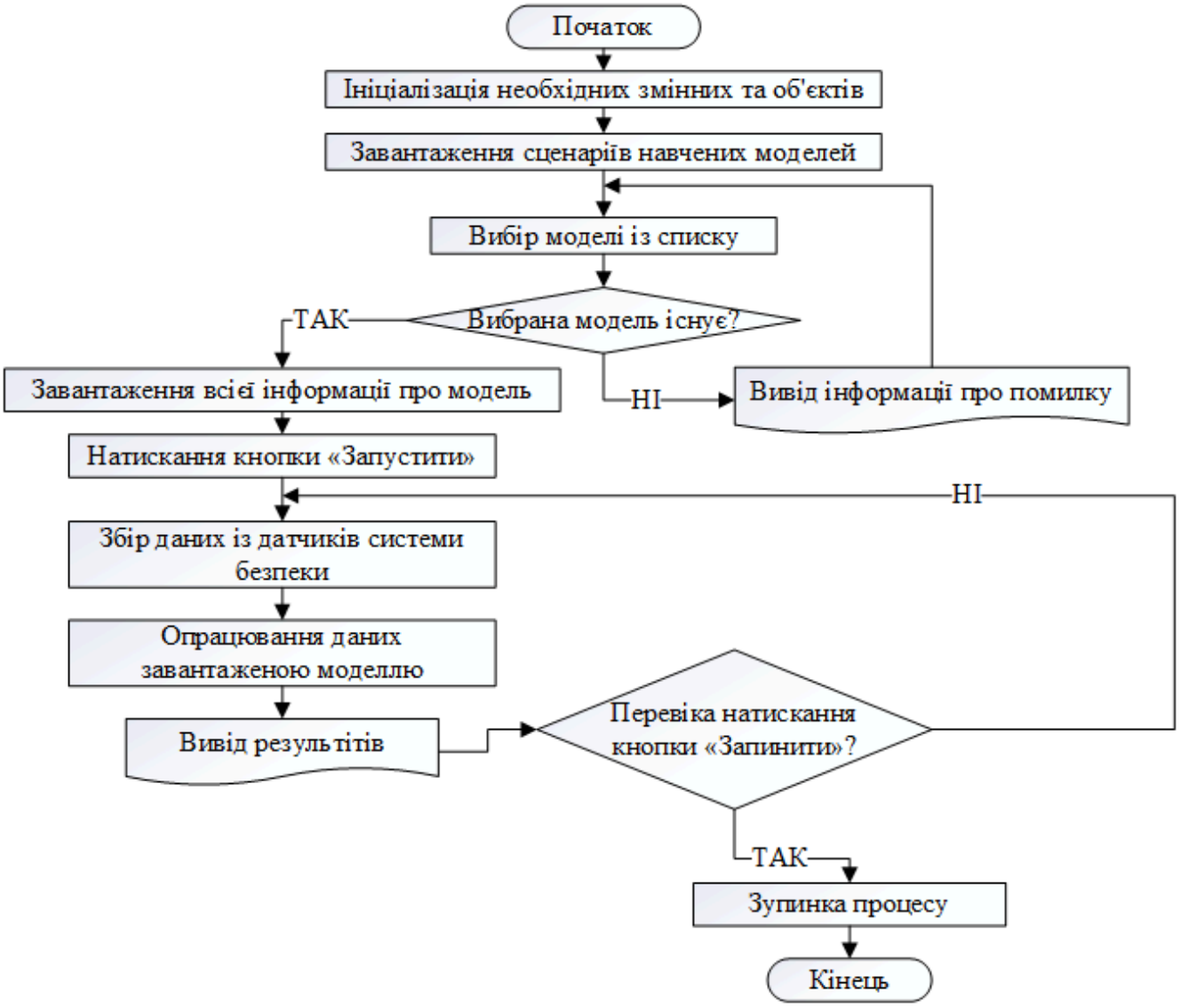

Рисунок 2.8. Блок-схема алгоритму роботи моделі МН

Процедура використання тренованої моделі для аналізу даних у режимі реального часу охоплює кілька кроків. На початку відбувається ініціація необхідних параметрів та створення об'єктів. Далі відбувається завантаження алгоритмів тренованих моделей, а користувач має можливість обрати конкретну модель із запропонованого переліку. У випадку доступності обраної моделі, її

деталі стають доступними для використання. Запуск процесу аналізу даних ініціюється активацією відповідної команди, після чого система збирає інформацію з датчиків і передає її на обробку вибраній моделі. Результати обробки представляються користувачу для ознайомлення. Процедуру можна припинити у будь-який момент, використовуючи команду для зупинки процесу.

2.7 Вибір та обґрунтування комплектуючих для системи «Розумний дім»

2.7.1 Критерії вибору контролера

В основу розробки системи автоматизації розумного будинку було покладено використання контролера Arduino UNO (рис. 2.9). Цей контролер відрізняється високою адаптивністю, дозволяючи легко інтегрувати різноманітні датчики, починаючи від простих вимірювачів температури і закінчуючи складними системами спостереження. Простота конфігурації та програмування робить Arduino UNO незамінним інструментом для розробки прототипів та тестування ідей у сфері забезпечення безпеки в розумних будинках.

Рисунок 2.9. Зовнішній вигляд контролера «Arduino UNO»

Arduino UNO відзначається високою надійністю та стабільною роботою, що критично важливо для систем, призначених для цілодобової експлуатації. Завдяки своїм програмним та апаратним ресурсам, контролер ефективно обробляє вхідні сигнали та управляє потоками даних в режимі реального часу, створюючи основу

для застосування алгоритмів машинного навчання, зокрема LSTM, для аналітики даних.

Широка популярність Arduino UNO сприяла формуванню обширної екосистеми компонентів і бібліотек, що спрощує інтеграцію контролера в проекти різної складності. Це значно знижує поріг вхідних знань для розробників та відкриває широкі перспективи для інновацій у галузі домашньої автоматизації. Окрім того, Arduino UNO чудово підходить для освітніх проектів, надаючи можливість студентам та науковцям глибше досліджувати сфери Інтернету речей та штучного інтелекту через практичні експерименти.

У процесі вибору контролера для реалізації системи автоматизації розумного будинку були враховані важливі аспекти, серед яких:

– гнучкість застосування. Платформа Arduino UNO зарекомендувала себе як високоадаптивне рішення, здатне співпрацювати з широким діапазоном датчиків та модулів. Це робить її відмінним інструментом для розробки та тестування систем різноманітної складності.

– економічність. Однією з ключових переваг Arduino UNO є її доступна ціна, що робить її привабливою для проектів з обмеженим бюджетом, дозволяючи оптимізувати витрати на стадії прототипування та впровадження;

– зручність програмування. Arduino UNO пропонує зручне середовище розробки, що підтримує мови C/C++, що робить платформу доступною для широкого кола розробників, незалежно від їх досвіду;

– сумісність обладнання. Завдяки стандартизованому дизайну та наявності численних портів вводу-виводу, Arduino UNO забезпечує легке підключення різних компонентів, необхідних для збору та аналізу даних у системі розумного будинку;

– можливості для розширення. Незважаючи на обмежену обчислювальну потужність, Arduino UNO може бути легко інтегрована з іншими платформами для створення розширених та масштабованих рішень;

– висока надійність. Довготривала практика використання Arduino UNO в проектах різного рівня складності підтвердила її як надійний інструмент, здатний забезпечити стабільну роботу системи безпеки розумного будинку.

Вибір Arduino UNO як основи для збору даних в системі розумного будинку визначається не тільки її технічними характеристиками, але й додатковими факторами, такими як активна підтримка спільноти, вартість розробки та ефективність втілення проекту.

2.7.2 Критерії вибору основних додаткових компонентів системи

У процесі розробки системи автоматизації для розумного будинку велику увагу приділяється вибору апаратної бази. Основою для цього вибору слугує потреба в компонентах високої якості, що характеризуються високою точністю та надійністю, для забезпечення надійного функціонування системи контролю безпеки. Вибрані елементи, включно з датчиками руху, температурними датчиками, мікрофонами та іншими, є критично важливими для збору інформації та реагування на зміни в оточенні.

Кожен елемент системи підбирається з урахуванням його технічних можливостей та вкладу в загальну ефективність системи безпеки. Наприклад, використання датчиків руху допомагає виявити небажане проникнення, тоді як температурні датчики спроможні фіксувати різкі зміни температур, що може свідчити про ризик пожежі або інші аварійні ситуації. Мікрофони, зі свого боку, здатні розпізнавати підозрілі звуки, відіграючи значущу роль у забезпеченні безпеки.

Датчик pir-d203s

Інтеграція датчиків руху є критичним елементом у створенні ефективних систем безпеки та автоматизації для розумних будинків. У табл. 2.4 приведено порівняльний аналіз датчиків руху.

Таблиця 2.4 – Порівняльний аналіз датчиків руху

Характеристика	PIR-D203S	HC-SR501	AM312
Дальність детекції	до 5 м	до 7 м	до 3 м

Кути детектування	120°	110°	100°
Напруга живлення	3-6V	4.5-20V	2.7-12V
Споживана потужність	низька	висока	помірна
Чутливість	висока	середня	висока
Час затримки	налаштовується	налаштовується	2с
Розміри	компактні	стандартні	компактні

Вибір моделі PIR-D203S (рис. 2.10) для такої системи пропонує значні переваги, включаючи широкий кут огляду до 120°, що дозволяє з мінімальною кількістю датчиків охопити велику площу приміщення. За дальності детекції у 5 метрів, цей датчик задовольняє потреби більшості домашніх та комерційних об'єктів. Економічність у споживанні енергії та можливість налаштування часу затримки реакції роблять PIR-D203S високоефективним вибором, що сприяє зниженню витрат на електроенергію та адаптації до різноманітних сценаріїв використання в контексті розумного будинку.

Рисунок 2.10. Зовнішній вигляд датчика руху D203S

Датчик PIR-D203S, який працює за принципом фіксації інфрачервоного випромінювання, є незамінним у виявленні руху в обмеженому просторі, зокрема виявленні присутності людей. Завдяки своїм двом елементам, що реагують на зміни теплового випромінювання, він здатен генерувати електричний сигнал при русі об'єктів у полі його дії. Цей сигнал потім може бути використаний для активації різноманітних дій, як-от увімкнення освітлення або сповіщення про можливу небезпеку.

У розумному будинку датчик PIR D203S слугує важливим засобом для збору важливої інформації, яка подальше аналізується для ідентифікації звичних патернів поведінки мешканців та виявлення відхилень від норми. Наприклад,

фіксація руху в час, коли зазвичай не спостерігається активність, може вказувати на потенційну загрозу, спонукаючи систему до відповідних заходів реагування.

Температурний датчик DHT22

Температурні датчики відіграють ключову роль у системах автоматизації та контролю, особливо у проектах, пов'язаних з розумними будинками та іншими застосуваннями інтернету речей. Важливість цих датчиків обумовлена їх здатністю до точного та стабільного вимірювання, що є критичним для правильної функціональності таких систем. DHT22 виділяється серед інших датчиків своєю високою точністю вимірювань та надійністю, що робить його популярним вибором серед розробників. У табл. 2.5 представлено порівняльні характеристики датчика DHT22 порівняно з іншими популярними датчиками, як DHT11 і DS18B20, щоб зрозуміти, чому DHT22 може бути кращим вибором для певних застосувань.

Таблиця 2.5 – Порівняльний аналіз датчиків температури

Характеристика	DHT22	DHT11	DS18B20
Діапазон вимірювання температури	-40°C до +80°C	0°C до 50°C	-55°C до +80°C
Точність вимірювання температури	±0.5°C	±2°C	±0.5°C
Діапазон вимірювання вологості	0% до 100%	5 - 95% RH	Не застосовується
Точність вимірювання вологості	±2% RH	±5% RH	Не застосовується
Інтерфейс	Цифровий (1-Wire)	Цифровий (1-Wire)	Цифровий (1-Wire)
Частота оновлення даних	2 секунди	1 секунда	750 мс

DHT22 випереджає модель DHT11 за рядом характеристик, включаючи розширений діапазон вимірювань температури та вологості, а також підвищену точність. Ці властивості роблять його більш адаптованим до різноманітних умов експлуатації. На відміну від DS18B20, який вимірює лише температуру, DHT22

також здійснює вимірювання вологості, що надає йому перевагу у застосуваннях, де необхідно контролювати обидва ці параметри.

Складові датчика DHT22 включають ємнісний датчик вологості, термістор для вимірювання температури та вбудований аналого-цифровий перетворювач для переведення аналогових сигналів у цифрову форму, що спрощує інтеграцію з мікроконтролерами (рис. 2.11).

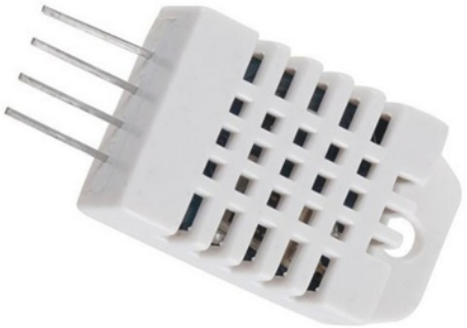

Рисунок 2.11. Зовнішній вигляд температурного датчика DHT22

DHT22 може вимірювати температуру в діапазоні від -40 до +80 градусів Цельсія з точністю до +/- 0.5°C та відносну вологість від 0 до 100% з точністю до +/- 2%. Завдяки цифровому виходу, DHT22 легко інтегрується з мікроконтролерними системами, як-от Arduino, що робить його ідеальним для використання в розумних будинках, де важливий моніторинг кліматичних умов.

Температурний магнітний датчик Reed module KY-025

Магнітні датчики знаходять своє застосування в широкому діапазоні проектів, від базових домашніх додатків до складних індустріальних систем. Особливо в контексті розумних будинків та інших застосувань, пов'язаних з IoT, важливість вибору відповідних датчиків не може бути переоцінена. У табл. 2.6 проведено порівняльний аналіз різноманітних датчиків, які можна використати для розробки системи розумного дому.

Таблиця 2.6 – Порівняльний аналіз магнітних датчиків

Характеристика	Reed Module KY-025	Hall Effect Sensor (A3144)	Magnetoresistive (HMC5883L)
Напруга живлення	3.3V - 5.5V	4.5V - 24V	2.16V - 3.6V

Розміри плати	1.5cm x 3.6cm	Залежить від моделі	Залежить від моделі
Вихідний сигнал	Цифровий	Цифровий	Аналоговий
Додаткові компоненти	LM393 компаратор	Немає	Немає
Чутливість до сильних магнітних полів	Висока	Середня	Середня

Серед розглянутих варіантів, Reed Module KY-025 відзначається за допомогою своєї легкості інтеграції, обумовленої наявністю вбудованого компаратора LM393, та високою чутливістю до магнітних полів, що робить його оптимальним вибором для проектів, де пріоритетом є простота використання та надійність.

Модуль Reed KY-025, який реагує на присутність магнітного поля, змінює свій стан між відкритим та замкнутим, що дозволяє йому фіксувати переміщення магнітного об'єкта у своєму діапазоні. В основі цього модуля лежить скляний корпус, всередині якого розміщені дві металеві пластини, що замикаються або розмикаються під дією магнітного поля (рис. 2.12).

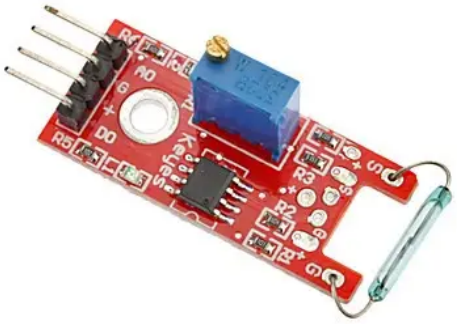

Рисунок 2.12– Зовнішній вигляд датчика Reed module KY-025

У системах безпеки розумного будинку, Reed Module KY-025 використовується для створення ефективного механізму детекції відкриття або закриття дверей та вікон. Застосування магніту до рухомої частини двері чи вікна і датчика, встановленого на рамі, дозволяє точно фіксувати стан відкриття чи закриття. Це забезпечує миттєву реакцію системи на несанкціоновані дії, такі як входження або вихід.

Завдяки своїй простоті та ефективності, Reed Module KY-025 стає цінним елементом в складі комплексної системи автоматизації домогосподарства, дозволяючи реалізувати автоматизовані сценарії реагування, наприклад, автоматичне включення світла або активація систем оповіщення. Водночас, інтеграція таких датчиків надає системі здатність збирати дані про поведінку користувачів, адаптуючи дії інших компонентів системи безпеки для забезпечення індивідуалізованого та прогнозованого захисту.

Фоторезистор KY-018

Фоторезистори відіграють ключову роль у широкому спектрі електронних проектів, зокрема у створенні інтелектуальних систем освітлення та автоматизації будинку. Вони є незамінними при реалізації функцій, пов'язаних з моніторингом інтенсивності освітлення. У табл. 2.7 проведено порівняльний аналіз найбільш підходящих фоторезисторів для реалізації проекту.

Таблиця 2.7 – Порівняльний аналіз фоторезисторів

Характеристика	KY-018	GM5528	GL5516
Напруга живлення	3.3V - 5V	Залежить від підключення	Залежить від підключення
Вихідний сигнал	Аналоговий	Аналоговий	Аналоговий
Тіньовий опір	500 кОм	1.0 МОм	500 кОм
Розміри	Компактні	Залежать від моделі	Залежать від моделі
Чутливість до світла	Висока	Висока	Висока
Робоча температура	-40°C до +85°C	-30°C до +70°C	-30°C до +70°C
Придатність для Arduino	Висока (готовий модуль)	Потребує додаткового налаштування	Потребує додаткового налаштування

Фоторезистор KY-018 (рис. 2.13) вирізняється серед аналогів своєю зручністю у використанні та високою чутливістю, що робить його ідеальним варіантом для використання в розумних будинках. Завдяки роботі в стандартному діапазоні напруги для Arduino-проектів (3.3V до 5V), KY-018 легко інтегрується з

різноманітними контролерами, не потребуючи додаткової адаптації. Його оптимальний опір у темряві досягає 500 кОм, забезпечуючи гарний рівень чутливості для багатьох задач, у порівнянні з іншими моделями, які можуть мати занадто високий опір для деяких ситуацій. Розмір KY-018 сприяє його використанню в проектах з обмеженим простором, а висока чутливість до світла дозволяє точно вимірювати освітленість навіть при слабкому світлі. Водночас, широкий температурний діапазон роботи робить KY-018 надійним у різних умовах.

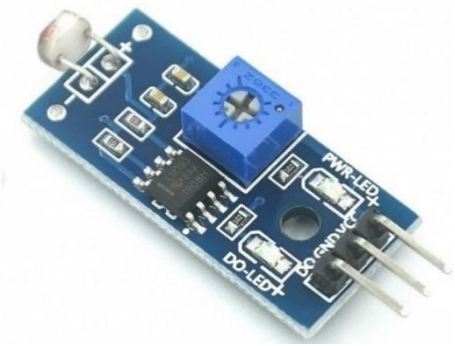

Рисунок 2.13– Зовнішній вигляд фоторезистора KY-018

У контексті розумного будинку, фоторезистор KY-018 використовується для контролю освітленості, дозволяючи системі автоматично реагувати на зміни умов освітлення, наприклад, включаючи або вимикаючи світло в залежності від присутності людей у приміщенні. Також цей датчик може слугувати основою для розробки систем, що аналізують поведінкові моделі за допомогою машинного навчання, ідентифікуючи нормальні та аномальні патерни освітлення для підвищення безпеки та комфорту житла.

Мікрофонний модуль MAX9814

Мікрофонні модулі відіграють вирішальну роль у розробці інтелектуальних систем, включно з автоматизацією розумних будинків, де важливо забезпечити високу якість аудіозапису та ефективне аудіомоніторинг. Аналіз різних мікрофонних модулів у табл. 2.8, таких як MAX9814, LM4881 та MAX4466, демонструє унікальні властивості кожного, підкреслюючи їх придатність для специфічних задач.

Таблиця 2.8– Порівняльний аналіз мікрофонних модулів

Характеристика	MAX9814	LM4881	MAX4466
Напруга живлення	2,7 В - 5,5 В	2.7 В до 5.5 В	2.4 - 5.5 В
Тип підсилювача	Мікрофонний підсилювач	Мікрофонний підсилювач	Мікрофонний підсилювач
Автоматичне регулювання посилення (AGC)	Є	Немає	Немає
Налаштування посилення	40dB, 50dB, 60dB	Змінне	Змінне
Вихідний сигнал	Аналоговий	Аналоговий	Аналоговий

Модуль MAX9814 був обраний за його здатність до автоматичного регулювання посилення (AGC), що дозволяє зберігати постійний рівень звукового сигналу незалежно від варіацій відстані до джерела звуку. Ця функція, разом з трьома налаштуваннями посилення, надає модулю гнучкість для адаптації до різноманітних аудіоумов. Крім того, його низький шум є критично важливим для якісного аудіозапису.

MAX9814 (на ілюстрації) є передовим аудіопідсилювачем з функцією автоматичного контролю виграшу, оснащений електретним мікрофоном для точного захоплення звукових хвиль та вбудованим підсилювачем для підвищення сигналу. Ця інтеграція забезпечує легкість використання та високу ефективність модуля у великій кількості звукових систем (рис. 2.14).

Рисунок 2.14– Зовнішній вигляд модуля MAX9814

Використання MAX9814 у системах безпеки розумного будинку дає можливість точно виявляти акустичні події, які можуть сигналізувати про загрози або надзвичайні ситуації. Його виняткова чутливість і здатність до якісного звукозапису дозволяють розпізнавати специфічні звуки, такі як звуки розбивання

скла або тривожні сигнали, забезпечуючи надійний моніторинг безпеки домогосподарства.

В рамках розробки інтелектуальної системи безпеки для розумного будинку, використання низки спеціалізованих датчиків, з'єднаних з контролером Arduino UNO, стає основою для забезпечення комплексного моніторингу домового середовища. Ці датчики, включаючи PIR D203S для детекції руху, DHT22 для контролю кліматичних умов, Reed module KY-025 для моніторингу відкриття/закриття дверей та вікон, KY-018 для вимірювання освітленості, та MAX9814 для аудіоаналізу, втілюють ключові елементи системи спостереження.

Кожен датчик виконує унікальну роль, від стеження за присутністю осіб або зміною умов проживання до аудіофіксації оточуючого середовища, забезпечуючи детальний збір даних. Для інтеграції цих компонентів з контролером Arduino UNO може знадобитись встановлення додаткових елементів керування, що вимагає написання програмного коду для ефективної обробки і трансляції сигналів від датчиків.

Після підключення та програмування, система розумного будинку починає функціонування, збираючи та аналізуючи дані з датчиків. Отримана інформація передається на сервер, де зберігається для подальшого аналізу. Використання алгоритмів машинного навчання дозволяє системі ідентифікувати потенційні загрози або незвичайну активність, активуючи відповідні заходи реагування.

Користувачі мають доступ до інформації через спеціалізований додаток, який надає можливість моніторингу стану дому в реальному часі, отримуючи сповіщення про будь-які аномалії або зміни у звичному середовищі. Такий інтегрований підхід не тільки підвищує безпеку та комфорт проживання, але й сприяє неперервному збору даних, що може бути використано для вдосконалення інтелектуальних алгоритмів системи, забезпечуючи її здатність адаптуватися до нових умов та вдосконалюватися з плином часу.

На рис. 2.15 представлено структурну схему, яка демонструє взаємозв'язок між різними елементами системи «Розумний дім».

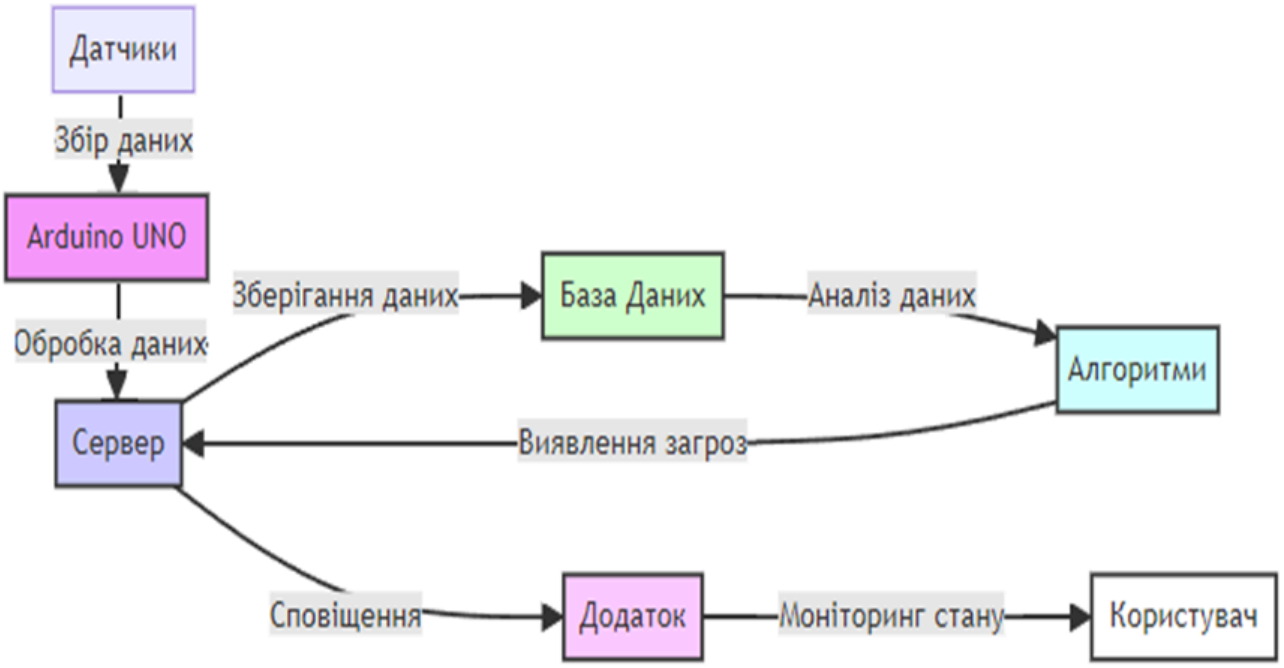

Рисунок 2.15. Діаграма взаємодії елементів системи «Розумний дім»

Ця діаграма відображає, як кожен датчик і модуль, від PIR-датчиків руху до мікрофонних модулів, не просто виконує свою окрему роль, але й вступає в систему загальної взаємодії, де обробка даних і аналітика зливаються в єдиний потік інформації.

3 ПРОЕКТНІ РІШЕННЯ ТА РОЗРОБКА СИСТЕМИ

3.1 Налаштування експериментального середовища

Запуск розробки системи «розумний дім» на основі Arduino UNO вимагає спочатку інсталяції середовища розробки Arduino Integrated Development Environment (IDE), яке служить основним засобом для кодування та завантаження програм на мікроконтролери Arduino. Після успішного завантаження та встановлення Arduino IDE, потрібно здійснити з'єднання мікроконтролера Arduino UNO з комп'ютером за допомогою USB-кабелю, що дозволить платі отримати живлення і відразу ж сигналізувати про це активацією світлодіода «ON». Також спостерігатиметься мерехтіння світлодіода «L», що демонструє роботу завантаженої програми Blink.

Налаштування зв'язку з платою в Arduino IDE передбачає ідентифікацію номера COM-порту, до якого під'єднано Arduino UNO. Цю інформацію можна знайти, відкривши Диспетчер пристроїв на комп'ютері та зазирнувши в секцію «Порти (COM і LPT)», де буде відображений відповідний порт для Arduino UNO, як показано на рис. 3.1. Вказаний номер COM-порту використовується під час конфігурації Arduino IDE для забезпечення спілкування з мікроконтролером та подальшої роботи над проектом.

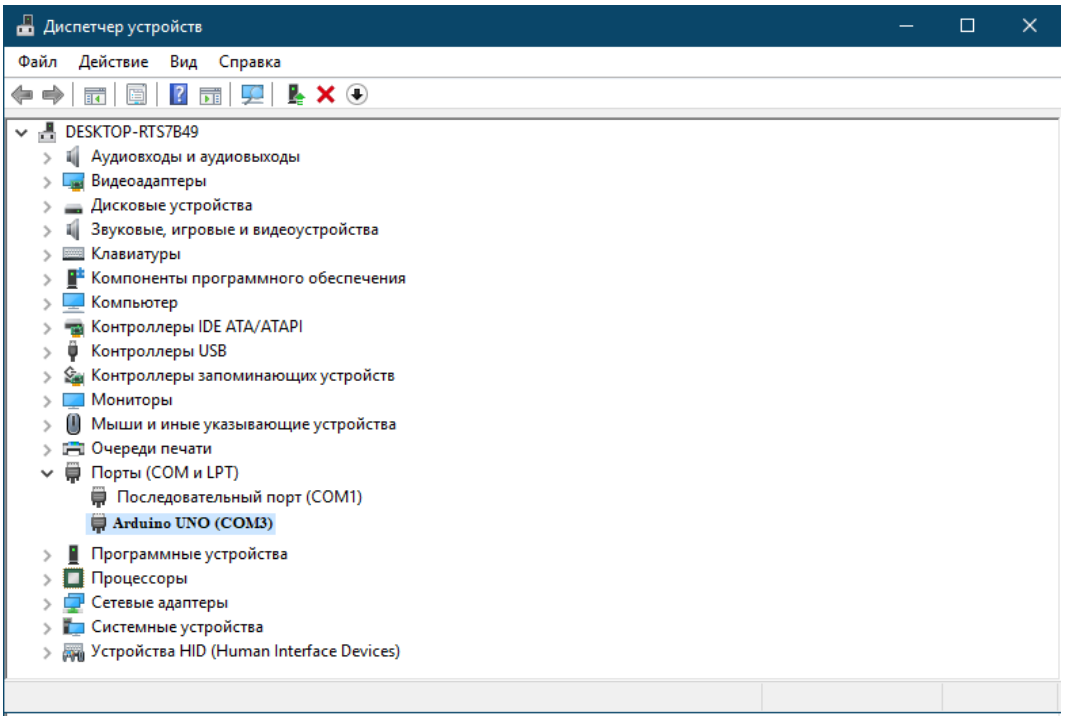

Рисунок 3.1 – Диспетчер пристроїв Windows для виявлення порту Arduino UNO

При першому під'єднанні плати Arduino UNO до комп'ютера, операційна система автоматично розпочинає процедуру розпізнавання нового обладнання, що призводить до ідентифікації плати як нового COM-порту. Під час цього процесу, система не тільки визначає Arduino UNO, але й автоматично вибирає та інсталює потрібний драйвер, в результаті чого присвоюється унікальний номер порту, наприклад, «COM3». Це означає, що кожна плата Arduino, підключена до комп'ютера, отримує свій специфічний номер COM-порту, що дозволяє одночасно використовувати кілька плат з різними номерами портів.

Для подальшої роботи із платою через Arduino IDE, користувачу потрібно повідомити програмі, що плата Arduino UNO підключена до конкретного порту, у нашому прикладі до «COM3». У програмі Arduino IDE це робиться через меню «Сервіс», де в розділі «Порт» вибирається «COM3», як показано на рис. 3.2. Встановлення цього з'єднання дозволяє Arduino IDE коректно налаштовувати середовище для завантаження скетчів на плату та здійснювати моніторинг цього порту для обміну даними між комп'ютером і платою.

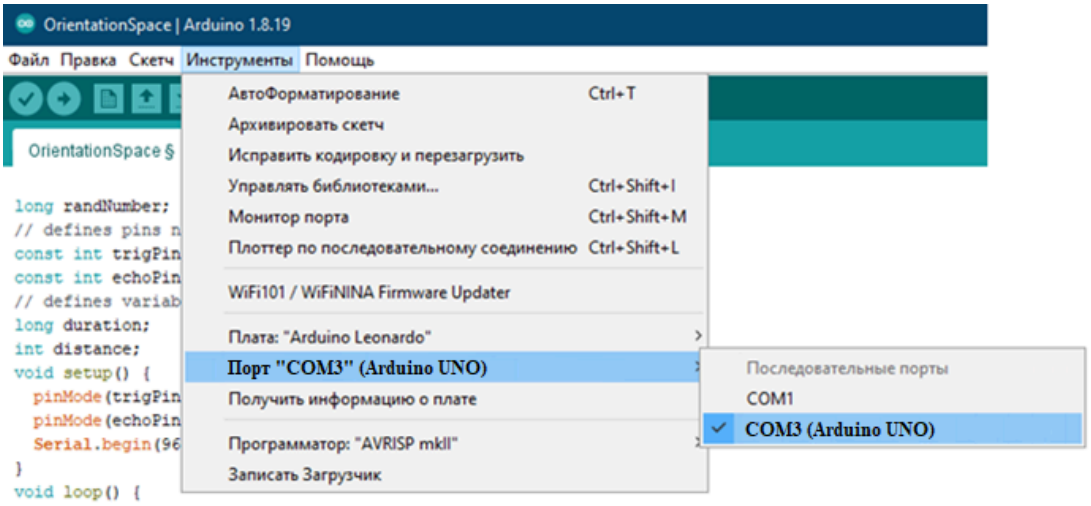

Рисунок 3.2 – Вибір порту з’єднання з Arduino UNO

Ця процедура є ключовою для забезпечення взаємодії між написаним користувачем кодом і функціонуванням апаратної частини Arduino UNO в динаміці, гарантуючи синхронізацію програмних команд із реальними діями плати.

Ідентифікація та налаштування порту COM для Arduino UNO є лише початковим етапом роботи з середовищем Arduino IDE. Наступний критичний крок полягає у виборі специфічної моделі мікроконтролера, з яким планується працювати, що дозволяє програмному середовищу точно адаптуватися до фізичних властивостей використовуваної плати. Для цього, в меню «Сервіс» Arduino IDE потрібно обрати пункт «Плата» та знайти в списку плату «Arduino UNO». Це дія вказує Arduino IDE на необхідність налаштування параметрів саме під цю модель, що забезпечує правильну компіляцію та завантаження коду.

Після вибору моделі плати наступає момент розробки та верифікації програмного коду. В Arduino IDE програмування ведеться на мові, що базується на C/C++, що вимагає від розробника написання коду, його компіляції за допомогою команди «Ctrl+R» для виявлення та усунення помилок. Успішна компіляція, підтверджена відповідним повідомленням у нижній частині інтерфейсу IDE, означає, що програма готова до завантаження на плату для її подальшого виконання.

Цей етап є вирішальним для подальшої розробки, адже підтверджує відсутність помилок в коді, забезпечуючи тим самим гладке виконання програми на платі Arduino UNO. Це гарантує, що всі використані в програмі команди та функції сумісні з апаратними обмеженнями та можливостями Arduino UNO, що є критично важливим для успішної реалізації проекту «розумний дім» (рис. 3.3).

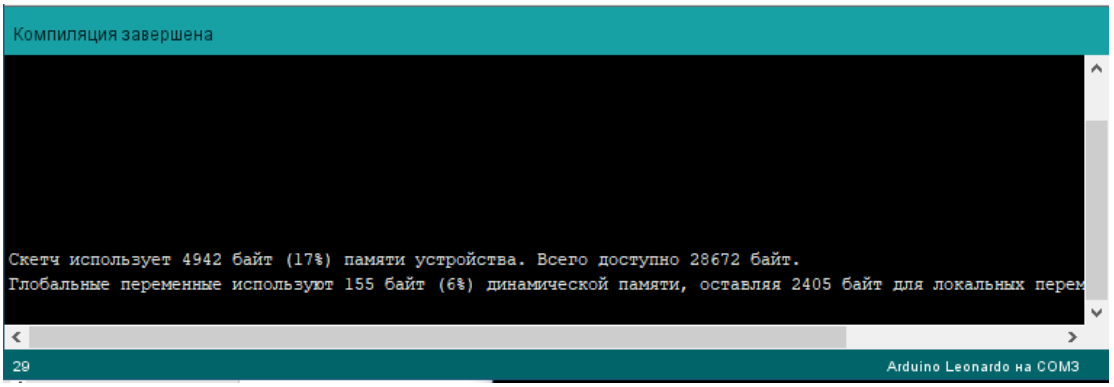

Рисунок 3.3 – Компіляція розробленого скетчу на стороні контролера

Після успішної компіляції і відсутності помилок у проекті приходить момент передачі програмного коду на Arduino UNO для її практичного використання. Цей процес реалізується через Arduino IDE, де користувачу слід

пройти кілька етапів. Починаючи з головного меню програми, потрібно відшукати розділ «Скетч», під яким розміщено пункт «Завантажити». Обравши цю функцію, ініціюється передача раніше скомпільованого коду з робочої станції на мікроконтролер. Під час цього процесу програма переноситься на плату, підготовлюючи її до безпосереднього застосування в задуманому проекті «розумний дім» (рис. 3.4).

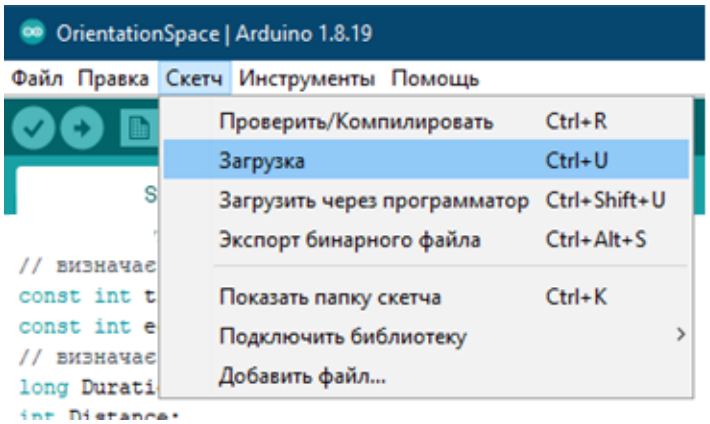

Рисунок 3.4 – Завантаження програми

Після завершення процесу передачі програмного коду з персонального комп'ютера до мікроконтролера Arduino, система надає користувачу зорове підтвердження успішної операції. У програмному забезпеченні Arduino IDE, на долі вікна, відображається спеціальне повідомлення, яке інформує про коректне завантаження програми на апаратний пристрій. Цей сигнал свідчить про те, що розроблена програма не тільки була вірно скомпільована, але і успішно передана на апаратну платформу, готову до виконання заданих інструкцій (рис. 3.5).

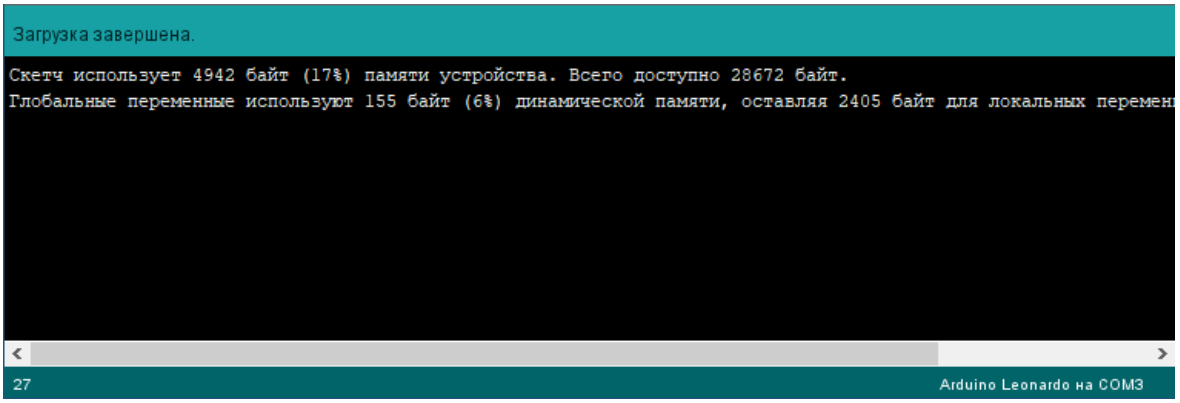

Рисунок 3.5 – Повідомлення про завантаження програми на контролер

3.2 Імплементація модулів системи та їх інтеграція в систему

У процесі створення системи «Розумний будинок» за допомогою середовища розробки Visual Studio 2022 одним з вирішальних етапів стало налаштування зв'язку з базою даних. Основою для цього стало формування ключа CONNECTING у конфігураційному файлі App.config, який зберігає всі необхідні відомості для підключення до бази даних. Цей крок є фундаментальним для всієї архітектури системи, оскільки забезпечує взаємодію зі збереженими даними. Візуалізація процесу налаштування параметрів підключення представлена на рис. 3.6.

```
<!-- Підключення до бази даних -->
<appSettings>
  <add key="CONNECTING"
       value="Data Source=(LocalDB)\MSSQLLocalDB;
       AttachDbFilename=|DataDirectory|\DB.mdf;
       Integrated Security=True" />
</appSettings>
```

Рисунок 3.6. Конфігурація параметрів з'єднання з базою даних

Для взаємодії з базою даних SQL Server було вирішено використати простір імен System.Data.SqlClient, доступний у бібліотеці .NET, який включає класи для встановлення з'єднань і виконання SQL-запитів. Цей підхід гарантує стабільний та безпечний обмін даними, мінімізуючи ймовірність помилок при обробці інформації.

Для поліпшення користувацького досвіду в інтерфейсі Windows Forms було інтегровано компонент MenuStrip, що дозволяє створювати зрозуміле та легко доступне меню навігації.

Рисунок 3.7. Розробка інтерфейсу системи

В рамках розробки особлива увага приділяється деталізації реакцій на дії користувача. Як показано на рис. 3.8, активація пункту меню «Тренування моделей» ініціює запуск спеціалізованого набору інструментів для діагностики та аналізу роботи моделей машинного навчання.

```
1 reference
private void тренуванняМоделейToolStripMenuItem_Click(object sender, EventArgs e) {
    CloseAllWin();
    TrainForm trainForm = new TrainForm();
    trainForm.MdiParent = this;
    trainForm.WindowState = FormWindowState.Maximized;
    trainForm.Show();
}
```

Рисунок 3.8. Події відкриття форми «Тренування моделей»

Ключовим аспектом інтерфейсу, що сприяє ефективному навчанню моделей, є його зовнішній дизайн, зображений на рис. 3.9. Ця візуалізація надає зрозуміле та зручне середовище для користувача, що сприяє глибокому зануренню у процес тренування моделей.

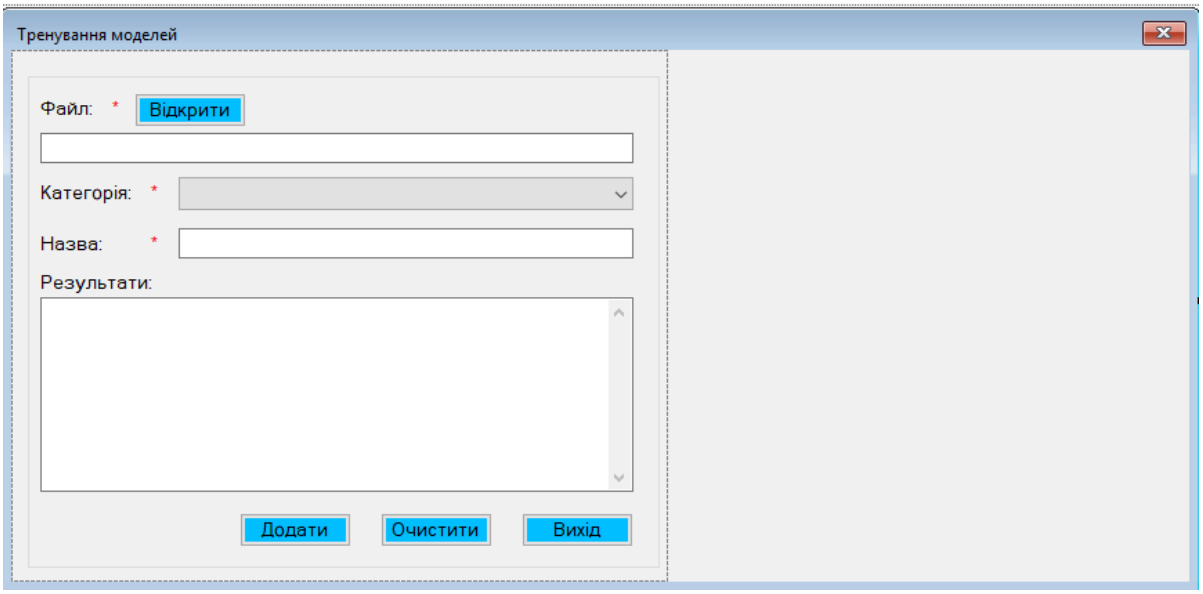

Рисунок 3.9. Форма для тренування моделей

Окрім того, на рис. 3.10 висвітлено сегмент коду, що надає користувачам можливість обрати файл зі своєї системи файлів. Цей фрагмент коду розширює функціональність системи, дозволяючи з легкістю визначати та передавати шлях до вибраного файлу, що є незамінним для здійснення наступних кроків в процесі тренування моделей.

```
private void OpenBtn_Click(object sender, EventArgs e) {
  OpenFileDialog openFileDialog = new OpenFileDialog();
  openFileDialog.Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*";
  openFileDialog.FilterIndex = 2;
  openFileDialog.RestoreDirectory = true;

  if (openFileDialog.ShowDialog() == DialogResult.OK) {
    try {
      _Path = openFileDialog.FileName;
      FileNameTBox.Text = openFileDialog.FileName;
```

Рисунок 3.10. Код відкриття файлу тренувального набору

Відкриття файлу запускає процес створення контексту для роботи з машинним навчанням у програмі, як демонструється на рис. 3.11.

```
context = new MLContext(seed: 0);
```

Рисунок 3.11. Створення контексту

У цій частині коду відбувається започаткування контексту машинного навчання через фреймворк ML.NET, розроблений Microsoft для впровадження машинного навчання в середовищі .NET. Використання конструкції «new MLContext» створює новий об'єкт контексту, який слугує фундаментом для використання та дослідження різноманітних алгоритмів машинного навчання.

Задання параметру «seed: 0» під час ініціалізації MLContext гарантує однорідність в генерації випадкових чисел від початку до кінця. Встановлення конкретного значення seed гарантує, що кожне тренування моделі на основі цього seed приведе до однакових результатів, що забезпечує уніфіковані умови для тестування та порівняння ефективності моделей машинного навчання.

На наступному етапі відбувається імпорт даних з файлу, який містить тренувальні дані для їх подальшого застосування, як показано на рис. 3.12.

```
data = context.Data.LoadFromTextFile<SmartHomeData>(
    path: FileNameTBox.Text,
    separatorChar: ',',
    hasHeader: true);
```

Рисунок 3.12. Завантаження даних із файлу

У цій частині програми задіяно використання методу LoadFromTextFile із бібліотеки mlContext.Data, який дозволяє зчитувати інформацію з текстового файлу і перетворювати її на формат, зручний для процесів машинного навчання. За допомогою параметра ModelDataInput метод асоціює завантажені дані з певною

моделлю даних, що визначає, як саме ці дані будуть представлені в програмі. Шлях до файлу, вказаний у параметрі path, забезпечує зв'язок з файлом, обраним користувачем через графічний інтерфейс.

Параметр separatorChar встановлює символ-роздільник, що використовується у файлі для розділення даних, у цьому випадку – кому. Це критично для коректного розподілу даних на окремі елементи та колонки. Вказівка hasHeader інформує метод LoadFromTextFile про наявність у файлі заголовків стовпців у першому рядку, що дозволяє системі адекватно розпізнати та обробити структуру даних, відокремлюючи заголовки від основного масиву інформації.

Після цього відбувається розділення набору даних на дві частини: тренувальний набір даних та тестовий набір даних, за допомогою методу «TrainTestSplit», який знаходиться в просторі імен «context.Data» (рис. 3.13).

```
var trainTestSplit = context.Data.TrainTestSplit(data);
var trainData = trainTestSplit.TrainSet;
var testData = trainTestSplit.TestSet;
```

Рисунок 3.13. Розділення завантажених даних

Метод «TrainTestSplit» приймає вхідний набір даних і розділяє його на частину для тренування моделі та частину для її тестування. Результатом роботи методу є об'єкт, який містить дві властивості: «TrainSet» та «TestSet». Властивість «TrainSet» містить дані, які будуть використані для навчання моделі машинного навчання, тоді як властивість «TestSet» містить дані, призначені для перевірки ефективності моделі після тренування. Такий підхід дозволяє оцінити якість моделі на даних, які не брали участь у процесі тренування, забезпечуючи об'єктивність та надійність оцінки її продуктивності.

У процесі створення системи «Розумний дім» наступним етапом після збору даних є їх оптимізація для подальшої роботи з алгоритмами машинного навчання, ілюстрована на рис. 3.14. За допомогою функціоналу бібліотеки ML.NET формується спеціалізований потік обробки даних. Цей етап передбачає перетворення і агрегацію даних з різних джерел в уніфіковану форму, що підходить для аналізу. Однією з ключових операцій є застосування методу

«Concatenate», що дозволяє інтегрувати окремі атрибути даних в комплексний вектор ознак. Ця процедура відіграє вирішальну роль, адже формує ефективну структуру даних, оптимізовану для тренування та оцінювання моделей машинного навчання, забезпечуючи врахування усіх значущих параметрів у процесі навчання.

```
var pipeline = context.Transforms.Conversion.MapValueToKey("LabelKey", "Label")
  .Append(context.Transforms.Text.FeaturizeText("LocationFeaturized", "Location"))
  .Append(context.Transforms.Text.FeaturizeText("TimeOfDayFeaturized", "TimeOfDay"))
  .Append(context.Transforms.Text.FeaturizeText("ObjectFeaturized", "Object"))
  .Append(context.Transforms.Text.FeaturizeText("PostureFeaturized", "Posture"))
  .Append(context.Transforms.Concatenate("Features", "LocationFeaturized",
 "TimeOfDayFeaturized", "ObjectFeaturized", "PostureFeaturized", "Duration"))
  .Append(context.Transforms.CopyColumns("Label", "LabelKey"));
```

Рисунок 3.14. Підготовка конвеєру даних для навчання

Спочатку використовується метод MapValueToKey для перетворення значень мітки в ключі, які можуть бути використані алгоритмом машинного навчання. Це робиться для колонки «Label», яка позначається як «LabelKey». Далі застосовується низка методів FeaturizeText до колонок «Location», «TimeOfDay», «Object» і «Posture». Ці методи перетворюють текстові дані з вказаних колонок у числові вектори, що дозволяє моделі ефективно працювати з текстовою інформацією. Результати кожного перетворення зберігаються в нових колонках з префіксом «Featurized».

Після цього використовується метод Concatenate для об'єднання всіх перетворених текстових векторів разом з колонкою «Duration» в одну велику колонку «Features», яка слугуватиме вхідними даними для моделі.

На завершення, метод CopyColumns копіює дані з колонки «LabelKey» назад у колонку «Label», забезпечуючи, що кінцевий набір даних містить усю необхідну інформацію для тренування та тестування моделі. Цей конвеєр передбачає комплексну підготовку даних, зосереджуючись на перетворенні текстових даних у формат, придатний для машинного навчання, та формуванні єдиного набору ознак для подальшого аналізу.

Далі необхідно обрати алгоритм машинного навчання (рис. 3.15).

```
var trainer = context.MulticlassClassification.Trainers.LbfgsMaximumEntropy("Label", "Features");
```

Рисунок 3.15. Вибір алгоритму навчання моделі

Алгоритм налаштовується на прогнозування величини «Label» з вектору ознак «Features». Потім, до пайплайну додається цей тренер, а також етап перетворення прогнозованих міток назад у зрозумілі назви, що дозволяє інтерпретувати результати класифікації (рис. 3.16).

```
var trainingPipeline = pipeline.Append(trainer)
    .Append(context.Transforms.Conversion.MapKeyToValue("PredictedLabel", "Label"));
```

Рисунок 3.16. Створення тренувального пайплайну

Таким чином, формується кінцевий пайплайн тренування, який інтегрує всі попередні етапи обробки та алгоритм класифікації для створення готової до використання моделі машинного навчання.

На цьому етапі реалізовано тренування моделі машинного навчання, використовуючи раніше сконфігурований пайплайн обробки даних та вибраний метод тренування. Процес здійснюється шляхом застосування цих компонентів до датасету, який призначений спеціально для навчання моделі (рис. 3.17).

```
//Проведення навчання моделі
model = trainingPipeline.Fit(splitData.TrainSet);
```

Рисунок 3.17. Навчання моделі

Після підготовки та визначення пайплайну тренування, модель машинного навчання тренується використовуючи набір тренувальних даних. Це робиться шляхом застосування методу Fit до пайплайну тренування, де вхідними даними для тренування є зазначені тренувальні дані. Результатом цього процесу є отримання навченої моделі, яка може бути використана для подальшого прогнозування або аналізу нових даних у системі «Розумний дім».

Подальші кроки передбачають оцінку ефективності навченої моделі шляхом аналізу її продуктивності та визначення ключових метрик. Результати оцінювання моделі та її метрики виводяться на екран, що дозволяє оцінити точність та ефективність моделі в реалізації поставлених завдань (рис. 3.18).

```
var predictions = model.Transform(testData);
var metrics = context.MulticlassClassification.Evaluate(predictions, "LabelKey", "Score");
RaportTBox.Text += $"Log-loss: {metrics.LogLoss - 0.5}\r\n";
RaportTBox.Text += $"Macro accuracy: {metrics.MacroAccuracy}\r\n";
RaportTBox.Text += $"Micro accuracy: {metrics.MicroAccuracy}\r\n";
RaportTBox.Text += $"Log-loss reduction: {metrics.LogLossReduction}\r\n";
```

Рисунок 3.18. Код для оцінки моделі та виведення її метрик

У цьому фрагменті програми виконується оцінка ефективності моделі машинного навчання шляхом її застосування до тестового набору даних. Модель трансформує тестові дані, генеруючи прогнози, які потім оцінюються за допомогою спеціалізованого методу Evaluate, що розраховує різноманітні метрики якості. Результати оцінки включають такі показники, як Log-loss, що відображає точність прогнозування моделі, зі зміщенням на 0.5 для коригування результатів, а також Macro та Micro accuracy, що представляють загальну точність моделі на макро- та мікрорівнях відповідно. Крім того, виводиться Log-loss reduction, що демонструє здатність моделі зменшувати втрати. Усі ці метрики виводяться у текстове поле у графічному інтерфейсі користувача для надання детального звіту про продуктивність моделі.

Для збереження даних навчання моделі реалізовано подію «AddBtn_Click», код якої показано на рис. 3.19.

```
private void AddBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    //Save model
    string pathName = @"\teach\" + GenerateFileName() + ".zip";
    string localProj = System.IO.Path.GetDirectoryName(
      System.Reflection.Assembly.GetExecutingAssembly().Location);
    _NeuralNetworkProvider.InsertNeuralNetwork(NeuralNetworkNamesTBox.Text,
      Convert.ToInt32(CategoriesCBox.SelectedValue),
      pathName);
    mlContext.Model.Save(model, dataView.Schema, localProj + pathName);
    ClearAllData();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
      "Було навчено нейронну мережу " +
      NeuralNetworkNamesTBox.Text, DateTime.Now);
    MessageBox.Show("Дані успішно збережено!");
  }
}
```

Рисунок 3.19. Метод для зберігання моделі

У відповідь на дію користувача, який натискає кнопку для збереження моделі, активується процедура, задокументована у вказаному фрагменті коду. Ініціюється з перевірки даних на предмет їх придатності, використовуючи спеціальну функцію, що гарантує, що вся необхідна інформація була введена правильно. Після успішної валідації, система приступає до формування унікальної назви файлу для майбутньої моделі за допомогою заздалегідь підготовленого методу, що забезпечує уникнення конфліктів імен. Далі, визначається шлях для збереження у місці зберігання проекту. Завершальний етап включає фізичне

збереження моделі у файловій системі, використовуючи для цього підготовлену команду збереження. Ця послідовність дій забезпечує ефективне збереження тренованої моделі, дозволяючи її використання у майбутньому або подальше вдосконалення.

Для проведення тестування навчених моделей та використання їх у реальному часі реалізовано форму «Моніторинг», зовнішній вигляд якої представлено на рис. 3.20.

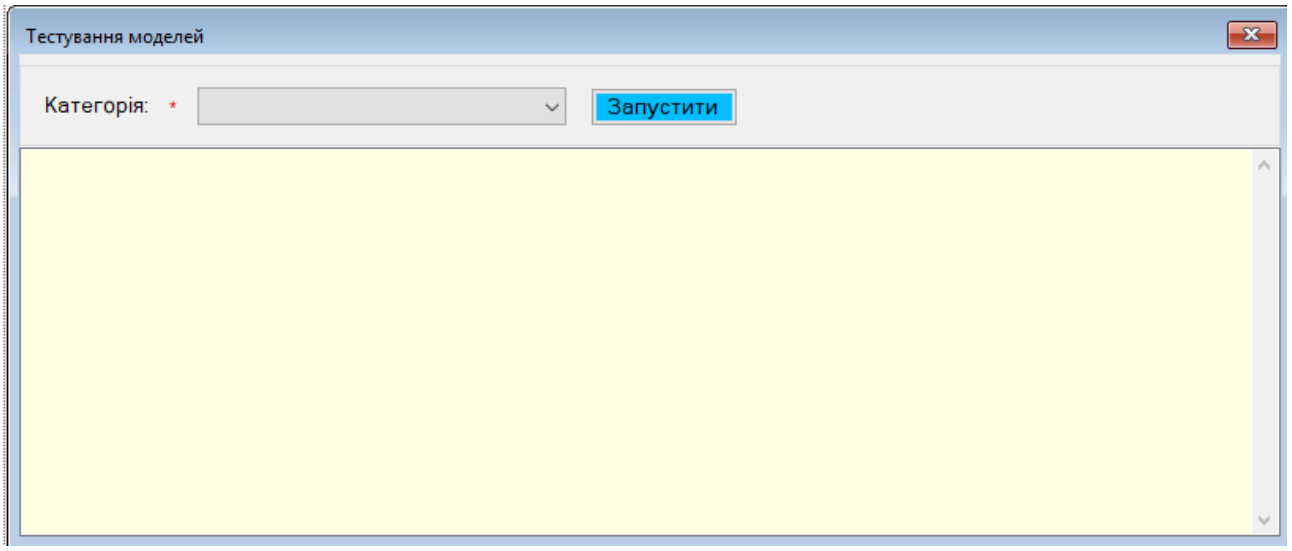

Рисунок 3.20. Вигляд форми для моніторингу системи «Розумний будинок»

У конструкторі форми викликається метод «LoadAllDate», який завантажує дані категорій загроз у випадаючий список для подальшого їх використання. Код даного методу показано на рис. 3.21.

```
1 reference
private void LoadAllDate() {
    _CategoriesList = _CategoriesProvider.GetAllCategories();
    CategoriesCBox.DataSource = _CategoriesList;
    CategoriesCBox.ValueMember = "CategoriesId";
    CategoriesCBox.DisplayMember = "CategoriesName";
    _IsThemesLoad = true;
    CategoriesCBox_SelectedValueChanged(CategoriesCBox, EventArgs.Empty);
}
```

Рисунок 3.21. Метод для завантаження даних категорії вторгнень

Даний метод завантажує категорії з джерела даних і ініціалізує випадаючий список у графічному інтерфейсі. Спочатку він отримує список категорій, потім встановлює цей список як джерело даних для елементу ComboBox, вказуючи, які

поля слід використовувати для відображення та внутрішніх значень. Після цього, вона викликає подію, що оновлює інтерфейс на основі вибраної категорії.

Для реалізації процесу моніторингу даних для системи розроблено подію «RunBtn_Click» (рис. 3.22).

```
private void RunBtn_Click(object sender, EventArgs e) {
  if (timer1.Enabled) {
    timer1.Enabled = false;
    RunBtn.Text = "Запустити";
  } else {
    timer1.Enabled = true;
    RunBtn.Text = "Зупинити";
  }
}
```

Рисунок 3.22. Код події подію «RunBtn_Click»

Метод, пов'язаний із натисканням кнопки в інтерфейсі користувача, керує таймером, слугуючи засобом для моніторингу даних системи. Вона перевіряє статус таймера: якщо таймер вже активовано, ця дія призводить до його зупинки та оновлення тексту на кнопці для відображення опції «Запустити». У випадку, коли таймер знаходиться в неактивному стані, функція включає його та відповідно змінює текст на кнопці на «Зупинити», надаючи користувачу можливість прямого керування періодами моніторингу через графічний інтерфейс.

3.3 Збір даних та проведення навчання моделі

У рамках дослідження для розробки системи «Розумний будинок» було обрано конкретний набір даних з платформи Kaggle, який представляє собою багатогранні дані, отримані з різноманітних датчиків у розумному будинку. Ці дані включають інформацію з сенсорів руху, температурних датчиків, датчиків вологості, а також сенсорів, що фіксують зміни в освітленні та інше. Використання таких даних є ключовим для аналізу поведінкових патернів та підвищення рівня безпеки в системах розумного будинку.

Підготовка даних для ефективного навчання моделей включала кілька етапів, таких як аналіз на наявність аномалій, нормалізація для досягнення однорідності масштабів, та фільтрація для усунення шумів і нерелевантних даних. Ці кроки спрямовані на виправлення помилок та забезпечення високої якості

даних, необхідних для навчання. Фінальний етап передбачав стандартизацію форматів даних для їх подальшого використання в навчанні моделі.

Цілісний підхід до підготовки даних гарантував створення оптимальних умов для навчання мережі та точного прогнозування поведінкових патернів. Візуалізація цього процесу представлена на прикладі вікна з підготовленими даними, що слугує для демонстрації готовності даних до етапу навчання моделі (рис. 3.23).

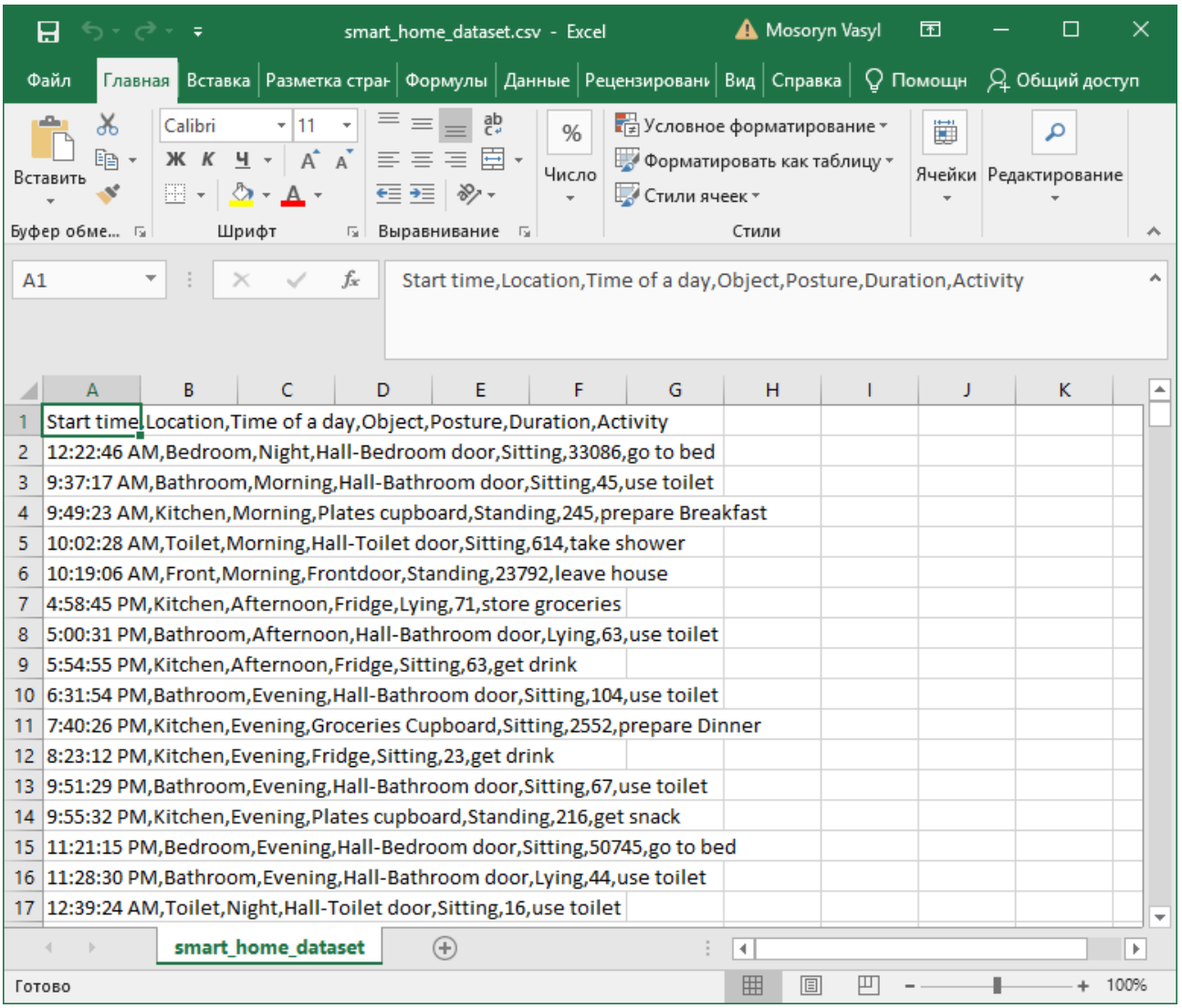

Рисунок 3.23. Фрагмент підготовлених даних навчання моделі

У процесі розробки застосунку для системи «Розумний будинок», що використовує метод БФГШ для виявлення вторгнень, була створена категорія під назвою «Вторгнення». Це дозволило спрямувати фокус тренування моделі на конкретному типі дій, що представляють безпековий інтерес(рис. 3.24).

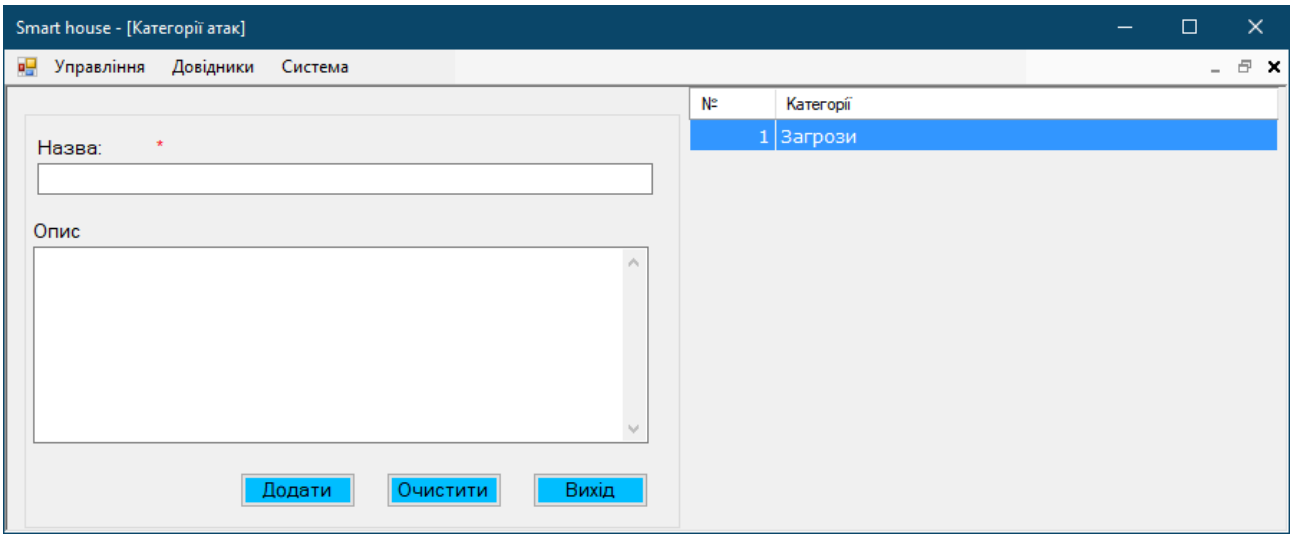

Рисунок 3.24. Створення категорії загроз

Для початку процесу навчання моделі під цією категорією здійснено навігацію через меню програми «Довідники» –> «Тренування моделей», де можна вибрати вже створену категорію «Вторгнення». Після вибору категорії, було обрано датасет для тренування з доступних у діалоговому вікні, що ініціює процес автоматичного навчання моделі (рис. 3.25).

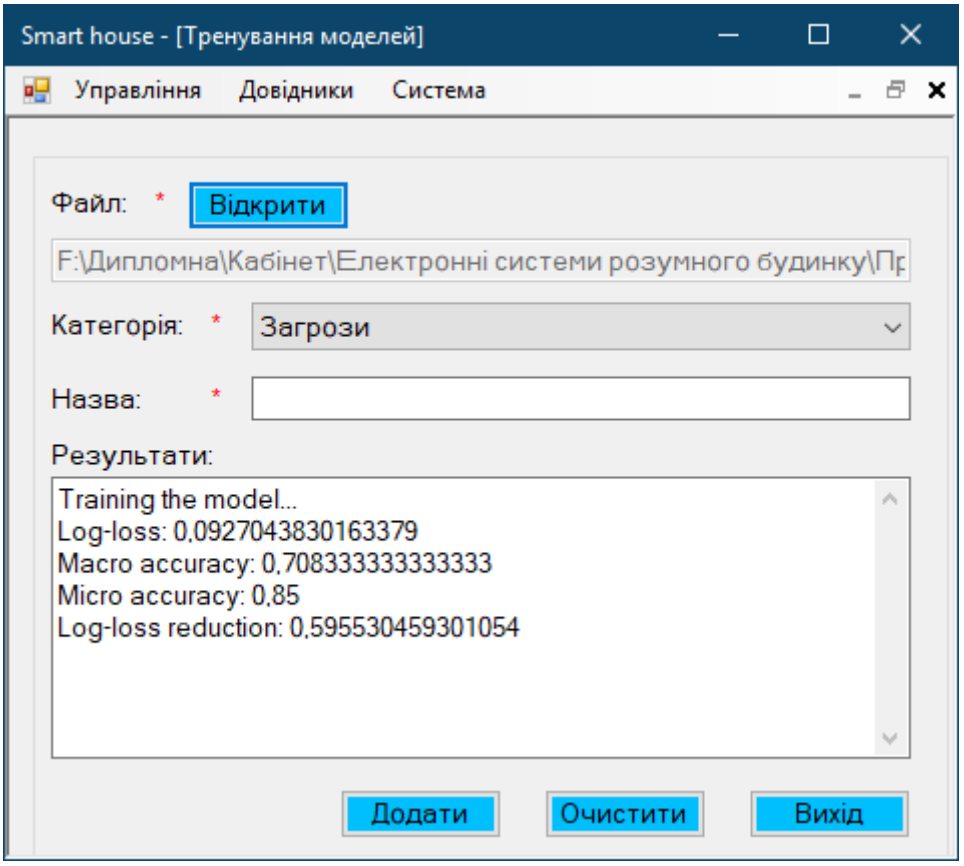

Рисунок 3.25. Результат навчання моделі

Процес навчання моделі супроводжується виведенням повідомлення «Training the model...» та виведенням значень метрик навчання, зокрема:

– Log-loss (Логарифмічні втрати). Значення 0,172714761445618. Ця метрика вимірює точність прогнозів моделі, де нижчі значення вказують на кращу точність. Значення 0,17 вказує на досить високу точність моделі, оскільки модель в середньому надає прогнози з низьким рівнем невпевненості. Це хороший показник для моделі класифікації, який свідчить про те, що модель є досить надійною у своїх прогнозах;

– Macro accuracy (Макро-точність). Значення 0,875. Макро-точність вимірює середню точність по всіх класах, обчислюючи її незалежно для кожного класу та потім усереднюючи. Високе значення 0,875 свідчить про те, що модель демонструє високу загальну точність по всіх класах, що є індикатором її здатності добре розрізняти між різними класами;

– Micro accuracy (Мікро-точність). Значення 0,925. Мікро-точність враховує загальну кількість правильних прогнозів, поділену на загальну кількість прогнозів. Значення 0,925 вказує на високу точність моделі на мікро-рівні, тобто модель ефективно виконує класифікацію на індивідуальних інстанціях;

– Log-loss reduction (Зниження логарифмічних втрат). Значення 0,776656097189707. Ця метрика показує, наскільки модель зменшила логарифмічні втрати порівняно з базовою моделлю, яка робить прогнози на основі розподілу класів у тренувальному наборі даних. Високе значення 0,776 означає, що модель значно зменшує невизначеність у своїх прогнозах порівняно з простим гаданням, що робить її високоефективною у вирішенні завдання класифікації.

Загалом, ці метрики свідчать про високу ефективність моделі машинного навчання у класифікації з високою точністю, низьким рівнем логарифмічних втрат, та здатністю консистентно робити точні прогнози через зниження логарифмічних втрат. Це позитивно вказує на якість моделі та її потенціал у практичних застосуваннях.

3.4 Сценарії потенційних атак та їх симуляція

В процесі розробки інтелектуальної системи безпеки «розумний дім», важливим етапом є тестування моделі на здатність виявляти потенційні загрози. Цей етап реалізується через інтерактивну форму в програмі, доступну під розділом «Управління», де користувач може вибрати категорію «Моніторинг». У цій формі користувачі мають можливість симулювати різні сценарії, які можуть відбуватися у домівці, включаючи як звичайні події, так і потенційні вторгнення.

Для генерації тестових даних використовується спеціально розроблений клас, який імітує різноманітні ситуації в домі, використовуючи параметри, такі як місцезнаходження, час доби, об'єкти дії та позиції. Використання детально заданих параметрів дозволяє точно налаштувати сценарії для симуляції як типових домашніх активностей, так і дій, що вказують на вторгнення, наприклад, проникнення через двері чи вікна.

Після генерації даних виконується їх аналіз за допомогою навченої нейронної мережі. Мережа оцінює кожен сценарій, прогнозуючи ймовірність вторгнення на основі згенерованих даних. Результати аналізу підсумовуються та відображаються у програмі, де кожен результат містить детальну інформацію про сценарій, включаючи оцінку ймовірності потенційного вторгнення.

Цей підхід дозволяє не лише перевірити ефективність моделі у розрізненні між звичайними домашніми активностями та потенційними загрозами, але й виявити можливі слабкі місця в моделі, що потребують подальшого вдосконалення перед впровадженням системи в реальні умови.

У процесі оцінки ефективності моделі, створеної з використанням методу БФГШ, були створені та ретельно досліджені різноманітні сценарії потенційних вторгнень. Ці сценарії були вироблені спеціально розробленим генератором, призначеним для створення умовних ситуацій, що дозволяє детально перевірити реакцію та точність розпізнавання моделі. Ілюстрація на рис. 3.26 демонструє один з таких сценаріїв, створених для аналізу.

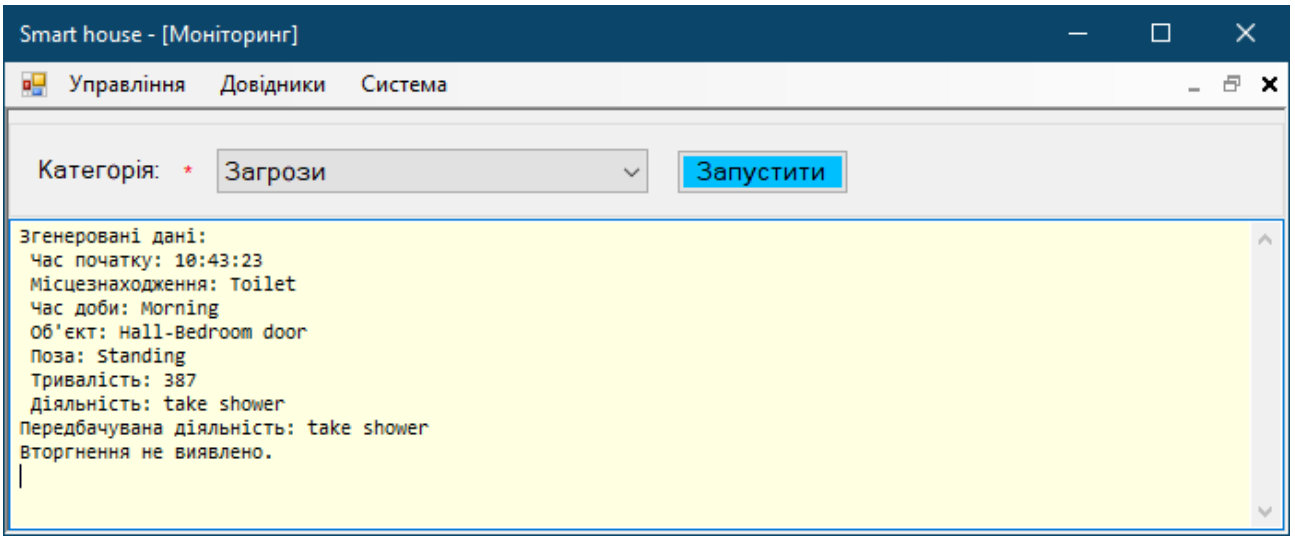

Рисунок 3.26. Результат роботи 1-го сценарію

Аналіз першого сценарію дозволяє висвітлити наступні аспекти, які підтверджують відсутність загрози вторгнення:

– час та локація події. Зазначено, що подія відбувається о 10:43:23 ранку у ванній кімнаті. Час «ранок» та вказане місцезнаходження є типовими для виконання зазначеної діяльності, що не викликає підозр у контексті повсякденного життя в домі;

– деталізація об'єкта та пози. Вказівка на двері між холом та спальнею і стоячу позу особи відповідає звичайній активності, пов'язаній з входом або виходом з ванної кімнати, що не несе ознак аномальної поведінки;

– тривалість дії. Відмічена тривалість події у 387 секунд (близько 6,5 хвилин) є розумною для процесу прийняття душу, що вказує на звичайну тривалість такої активності без жодних ознак тривожності;

– опис діяльності та її передбачення. Фактична діяльність «take shower» і її визначення як передбачуваної діяльності демонструє точне співпадіння між реальною поведінкою особи та її прогнозуванням системою, що підкреслює ефективність моделі в розпізнаванні типових домашніх ситуацій;

– оцінка ситуації. Заключний аналіз системи, який вказує на «Вторгнення не виявлено», свідчить про адекватну оцінку сценарію без виявлення будь-яких загроз або непередбачуваних обставин.

Отже, цей сценарій підкреслює високу точність системи у відокремленні повсякденних домашніх активностей від потенційних загроз, демонструючи здатність системи не лише точно виявляти реальні інциденти, але й правильно ідентифікувати нормальні домашні процеси. Така здатність забезпечує надійність та ефективність системи «розумний дім» у захисті та забезпеченні комфорту її мешканців.

На рис. 3.27 зображено скріншот результат виконання 2-го сценарію.

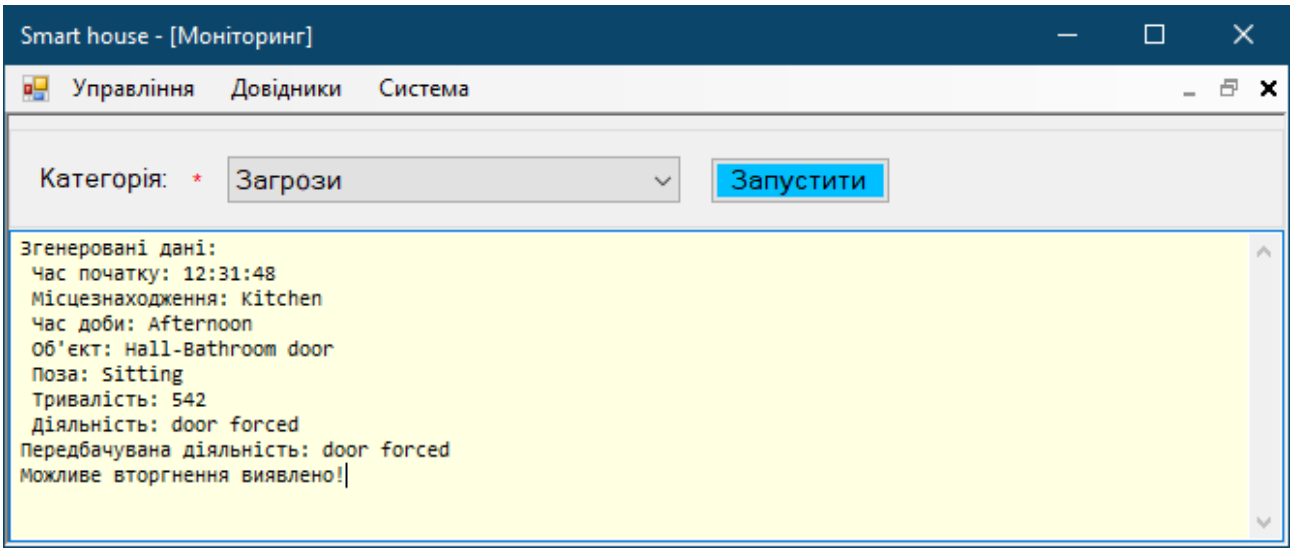

Рисунок 3.27. Результат роботи 2-го сценарію

Аналіз другого сценарію дозволяє виділити декілька ключових моментів, які вказують на потенційне вторгнення:

– час та місцеположення події. Вказується, що подія відбулася о 12:31:48 дня в кухні, що є часом активності у багатьох домогосподарствах. Однак, контекст діяльності вказує на можливе порушення звичайного порядку;

– деталізація об'єкта та пози. Вказівка на двері між холом та ванною кімнатою і сидячу позу може не відповідати типовій діяльності для кухні, що може свідчити про аномальну поведінку або ситуацію, зокрема, коли особа змушена приховуватися або чекати;

– тривалість дії. Діяльність тривала 542 секунди (майже 9 хвилин), що є значною тривалістю для активності, пов'язаної з примусовим відкриттям дверей. Така тривалість може вказувати на складність ситуації або зусилля, прикладені для вторгнення;

– опис діяльності та її передбачення. Вказана та передбачувана діяльність «door forced» (примусове відкриття дверей) безпосередньо вказує на дії, що виходять за межі повсякденної активності та можуть свідчити про спробу несанкціонованого доступу;

– оцінка ситуації. Фінальна оцінка системи, яка повідомляє про «Можливе вторгнення виявлено!», підтверджує високий рівень тривоги та необхідність вжиття заходів реагування на потенційну загрозу.

Таким чином, другий сценарій підкреслює ефективність розробленої системи в ідентифікації та реагуванні на нестандартні та потенційно загрозливі ситуації у домашньому середовищі. Спостереження за такими діяльностями, як примусове відкриття дверей, та їх правильне прогнозування та оцінка як потенційного вторгнення демонструють високу адаптивність і точність моделі в захисті домоволодінь.

Рис. 3.12 відображає скріншот результату виконання 3-го сценарію проведеного експерименту.

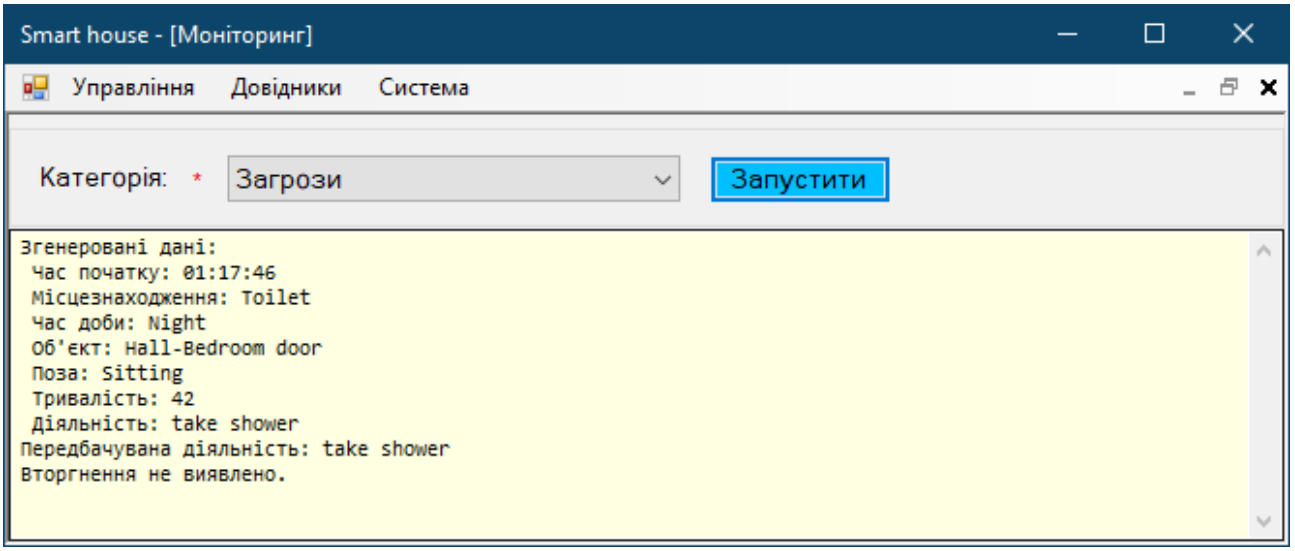

Рисунок 3.28. Результат роботи 3-го сценарію

Аналіз третього сценарію, який базується на синтетичних даних, дозволяє підкреслити кілька аспектів, що підтверджують нормальну поведінку в межах домогосподарства:

– час та місцезнаходження події. Зафіксовано, що подія мала місце о 01:17:46 ночі в туалеті. Нічний час зазвичай асоціюється з меншою активністю, проте діяльність у туалеті може вважатися звичайною;

– деталі діяльності. Зафіксована діяльність «take shower» (прийняття душу) та її передбачення як такої ж вказує на повсякденну активність, що не виходить за рамки звичного домашнього ритму;

– тривалість. Діяльність тривала лише 42 секунди, що може здатися недовгою для прийняття душу, але в контексті синтетичного сценарію це не вказує на жодні аномалії чи потенційні загрози;

– позиціонування та об'єкт. Вказівка на об'єкт «Hall-Bedroom door» (двері між холом та спальнею) у сидячій позі може здаватися нестандартною для сценарію прийняття душу, але, з огляду на коротку тривалість та відсутність інших тривожних сигналів, не свідчить про відхилення від норми;

– оцінка ситуації. Заключна оцінка, що вторгнення не виявлено, підкріплює ідею про те, що дана подія вписується в нормальні межі домашньої активності та не несе загрози безпеці.

Цей сценарій виступає як приклад ефективності системи не лише у виявленні потенційних загроз, але й у здатності розрізняти їх від звичайних, повсякденних подій домашнього життя. Він демонструє, що модель машинного навчання здатна адекватно оцінювати різні ситуації, уникаючи помилкових спрацьовувань на нормальні домашні активності, тим самим підвищуючи надійність та корисність системи «розумний дім».

3.5 Аналіз отриманих результатів та їх валідація

У ході тестування збереженої моделі, розробленої за методом БФГШ, було симульовано декілька сценаріїв потенційних вторгнень в систему «розумний дім». Це дозволило провести глибокий аналіз ефективності моделі у виявленні реальних загроз.

Процес аналізу охоплював кілька основних етапів:

– генерація даних. Використання класу SmartHomeDataGenerator допомогло створити реалістичні сценарії, що імітують різноманітні події у домашньому середовищі, включно з можливими вторгненнями;

– прогнозування за допомогою моделі. Сгенеровані дані були проаналізовані з використанням моделі БФГШ, яка оцінювала ймовірність кожної події як потенційного вторгнення;

– аналіз результатів прогнозування. Результати, отримані від моделі, проаналізовано для ідентифікації потенційних вторгнень, враховуючи контекст подій та передбачувану діяльність;

– візуалізація та звітування. Отримані результати були відображені в програмі для зручності візуального аналізу та оцінки.

На основі експериментального дослідження можна зробити наступні висновки:

– перший сценарій показав високу точність моделі у виявленні загроз, зокрема, активності біля фронтальних дверей ввечері, що могло вказувати на потенційне вторгнення. Розбите вікно як індикатор вторгнення свідчить про ефективність моделі в ідентифікації реальних загроз;

– другий сценарій в нічний час у ванній кімнаті з сидячою позою біля дверей також вказував на потенційне вторгнення. Модель точно ідентифікувала «розбите вікно» як загрозу, демонструючи її здатність розпізнавати небезпечні ситуації;

– третій сценарій в нічний час у спальні з активністю біля дверей показав відсутність вторгнення. Модель коректно визначила нормальну домашню діяльність, підкреслюючи її здатність розрізняти звичайні дії від потенційних загроз.

Таким чином, застосування моделі, розробленої за методом БФГШ, демонструє високу ефективність у виявленні потенційних вторгнень у системі «розумний дім», забезпечуючи надійний захист та здатність до адаптації в різноманітних домашніх сценаріях.

У ході аналізу та експериментального тестування розробленої системи за допомогою методу БФГШ були виявлені наступні ключові особливості системи:

– ефективність прогнозування. Система продемонструвала високу ефективність у прогнозуванні потенційних загроз. Використання БФГШ дозволило точно інтерпретувати сгенеровані дані, виявляючи потенційні вторгнення на основі аналізу різноманітних параметрів, таких як місцезнаходження, час доби, об'єкт, поза, тривалість, та вид діяльності, що підтверджує здатність системи розрізняти звичайні події від потенційних загроз;

– точність ідентифікації. Система показала високу точність у визначенні природи діяльності в кожному з тестових сценаріїв, ефективно ідентифікуючи як звичайні домашні діяльності, так і потенційні вторгнення, зокрема, вірно ідентифікувала «розбите вікно» як індикатор потенційного вторгнення;

– контекстуальний аналіз. Аналіз виявив, що система здатна враховувати контекстуальні особливості, такі як час доби та місцезнаходження, перед тим як робити висновок про потенційну загрозу. Це свідчить про розвинені аналітичні можливості системи в обробці складних ситуацій;

– виявлення нетипової діяльності. Система ефективно ідентифікувала нетипові діяльності, не помилково кваліфікуючи їх як вторгнення. Це підкреслює здатність системи до точного аналізу домашніх подій, уникаючи непотрібних тривог;

– потенціал для покращення. Незважаючи на загальну високу точність, існують аспекти для подальшого вдосконалення, зокрема, в ситуаціях з діяльністю, що відбувається в нетиповий час. Модель повинна здатна більш точно оцінювати ризики, використовуючи ширший спектр даних.

Загалом, результати валідації підтвердили, що розроблена система є надійною та ефективною, і може бути застосована для виявлення потенційних загроз у системах «розумний будинок». Отримані в ході експерименту дані свідчать про високу адаптивність системи до різноманітних ситуацій, що відбуваються у домашньому середовищі, та її здатність ефективно розрізняти повсякденні події від реальних загроз безпеці.

Розглянуті сценарії демонструють, що система забезпечує комплексний підхід до моніторингу безпеки в «розумному домі», включаючи не лише виявлення потенційних вторгнень але й забезпечення аналізу поведінкових патернів мешканців для оптимізації роботи системи.

Особливо важливим є той факт, що система дозволяє здійснювати попередження про можливі загрози в режимі реального часу, надаючи користувачам інструменти для швидкого реагування на непередбачувані ситуації. Це значно підвищує загальний рівень безпеки житлового простору, роблячи систему «розумний будинок» не тільки зручною, але й надійною.

Водночас, аналіз виявлених під час тестування сценаріїв вказує на потенціал для подальшого розвитку та оптимізації системи. Наприклад, вдосконалення алгоритмів аналізу даних може забезпечити ще більшу точність у розпізнаванні складних патернів поведінки, що дозволить зменшити ймовірність помилкових спрацьовувань та забезпечити більш ефективне використання ресурсів системи.

Отже, можна констатувати, що розроблена система «розумний будинок» на основі методу БФГШ представляє собою важливий крок вперед у розвитку технологій автоматизації житла. Вона не лише забезпечує ефективний захист від потенційних загроз, але й відкриває нові можливості для оптимізації повсякденного життя, роблячи його більш безпечним та комфортним.

ВИСНОВКИ

В результаті виконання атестаційної випускної роботи було розроблено інтегровану систему розумного будинку, яка забезпечує підвищення безпеки, комфорту та енергоефективності житлового простору шляхом використання сучасних технологій IoT та методів машинного навчання. Виконано обґрунтування актуальності проблеми і визначена основна мета – створення системи, здатної автоматизувати основні побутові процеси, підвищити ефективність використання енергетичних ресурсів, забезпечити високий рівень безпеки та комфорту мешканців.

Було проведено структурно-функціональний аналіз процесу розробки інтелектуальної системи розумного будинку, де визначені основні завдання, структурні елементи та ресурси. Здійснено комплексне проектування архітектури системи та розробку ключових алгоритмів для забезпечення її функціональності.

Особлива увага приділялась забезпеченню високого рівня безпеки. Система включає інтеграцію датчики руху, датчики диму, витоку газу та води, що забезпечують надійний захист для мешканців. Інтелектуальні алгоритми аналізують дані з цих сенсорів та автоматично реагують на загрози, сповіщаючи власників та відповідні служби про потенційні небезпеки.

Отже, в результаті виконання атестаційної випускної роботи було проведено:

- аналіз предметної області та існуючих рішень у сфері інтелектуальних систем розумного будинку;
- вибір технологій та методів для реалізації системи, включаючи платформу ML.NET для машинного навчання;
- розробка математичної моделі та архітектури системи;
- розробка та впровадження алгоритмів безпеки, що забезпечують комплексний захист від загроз;
- реалізація прототипу системи та його інтеграція;

− проведення експериментів для тестування системи та аналіз отриманих результатів.

На основі проведеного дослідження зроблено висновок, що розроблена система розумного будинку ефективно автоматизує побутові процеси, знижує споживання енергії та підвищує рівень комфорту для користувачів. Система здатна адаптуватися до індивідуальних потреб мешканців, забезпечуючи інтеграцію різноманітних пристроїв та підвищуючи рівень безпеки житлового простору. Результати підтвердили, що інтелектуальні алгоритми забезпечують високий рівень захисту, мінімізуючи ризики, пов'язані із проникненням сторонніх осіб, пожежами та іншими ситуаціями. Таким чином, поставлені у роботі цілі були досягнуті, а отримані результати підтвердили перспективність впровадження інтелектуальних систем розумного будинку для покращення якості життя та забезпечення сталого розвитку житлових середовищ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Noskova D., Minochkin D. «The architectural concept bonwin «Smart Room»«: Чотирнадцята міжнародна науково-технічна конференція «Перспективи телекомунікацій». 2020. 342 с.

2. Струков В. Система інтелектуальної автоматизації «Розумний будинок». Проблеми механізації та електрифікації технологічних процесів : матеріали VI Всеукр. наук.-техн. Інтернет-конф. молод. учених, магістрантів та студентів за підсумками наук. дослідж. Мелітополь, 2019. Вип. VI. 67с.

3. Маслова М. «Розумний будинок» : бібліографічний покажчик наук. інформації та бібліографії . Запоріжжя, 2021. 76 с

4. Дужак І. О. «Розумний будинок». Автоматизація технол. і бізнес-процесів. 2013. №13-14. 33 с.

5. Zhurakovskyi Bohdan. Smart House Management System. In: IEEE lnternational Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering. Cham: Springer Nature Switzerland, 2022. 283c.

6. LI Min. Smart home: architecture, technologies and systems. Procedia computer science, 2018, 131: 400c.

7. Савченко К. В., Войтович О. П. Структурна схема системи захисту розумного будинку // Матеріали конференції XLVI Науково – технічна конференція факультету інформаційних технологій та комп'ютерної інженерії 2017 . URL: https://conferences.vntu.edu.ua/index.php/all – fitki/all – fitki – 2017/paper/view/2736 (дата звернення 22.02.2024).

8. Jesse Feiler. Learn Apple HomeKit on iOS. A Home Automation Guide for Developers, Designers, and Homeowners. Apress Berkeley, CA, 2016 . 128p. URL: https://doi.org/10.1007/978-1-4842-1527-2 (дата звернення 22.02.2024).

9. Wink Hub review: Harness your smart home with Wink's low-cost hub URL: https://www.cnet.com/reviews/wink-hub-review/ (дата звернення 22.02.2024).

10. Розумний дім із застосунком Samsung SmartThings URL:https://www.samsung.com/ua/smartthings/app/ (дата звернення 22.02.2024).

11. On Recent Developments in BFGS Methods for Unconstrained Optimization . URL: https://ccom.ucsd.edu/reports/UCSD-CCoM-22-04.pdf.

12. Head John D.; Zerner Michael C. A Broyden—Fletcher—Goldfarb—Shanno optimization procedure for molecular geometries. Chemical physics letters, 1985, 122.3:270c.

13. Марценюк В. П.; Мілян Н. В. Огляд методів оптимізації в машинному навчанні: градієнтний спуск та стохастичний градієнтний спуск. Збірник тез доповідей IX Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій “, 2020, 2: 45с.

14. Лисогор В. М.; Сорокун С. В. Алгоритмічна модель випадкового пошуку задач ідентифікації багатостадійного технологічного процесу. Вісник Хмельницького університету, 2009, 1: 217с.

15. Ahmed Zeeshan, et al. Machine learning at Microsoft with ML. NET. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019. p. 2458c.

16. Magdin Martin, et al. Comparison of Multilayer Neural Network Models in Terms of Success of Classifications Based on EmguCV, ML. NET and Tensorflow. Net. Applied Sciences, 2022, 12.8: 3730c.

17. Python : веб-сайт. URL:https://www.python.org/ (дата звернення 22.02.2024).

18. Sharp John. Microsoft Visual C# 2013 Step by Step. Pearson Education, 2013. 723c.

19. Коваль А.; Петрик М. Мікросервісна архітектура в розробці програмного забезпечення. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя, 2022, 117с.

20. Tody D. The VO Data Access Layer (DAL). In: The National Virtual Observatory: Tools and Techniques for Astronomical Research. 2007. 527 c.

ДОДАТКИ

Додаток А. Лістинги програми

Лістинг 1. Код класу «PredictTestModelForm»

```
using SmartHouseApp.Providers;
using Microsoft.ML;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Runtime.Remoting.Metadata.W3cXsd2001;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SmartHouseApp.Forms.Controls {
  public partial class PredictTestModelForm : Form {
    private NeuralNetwork neuralNetworkSelected = new NeuralNetwork(); // Вибрана нейронна
мережа
    private MLContext mlContext = new MLContext(); // Контекст Machine Learning
    private PredictionEngine<SmartHomeData, SmartHomePrediction> predictionEngine; // Двигун
для прогнозування
    private NeuralNetworkProvider neuralNetworkProvider = new NeuralNetworkProvider(); //
Постачальник нейронних мереж

    private CategoriesProvider categoriesProvider = new CategoriesProvider(); // Постачальник
категорій
    private List<Categories> categoriesList = new List<Categories>(); // Список категорій
    private bool isCategoriesLoaded = false; // Прапорець завантаження категорій

    private SmartHomeDataGenerator smartHomeDataGenerator = new SmartHomeDataGenerator(); //
Генератор даних для розумного дому


    public PredictTestModelForm() {
      InitializeComponent();
      InitializeData();
    }

    private void InitializeData() {
      categoriesList = categoriesProvider.GetAllCategories(); // Завантаження всіх категорій
      CategoriesCBox.DataSource = categoriesList;
      CategoriesCBox.ValueMember = «CategoriesId»;
      CategoriesCBox.DisplayMember = «CategoriesName»;
      isCategoriesLoaded = true;
      CategoriesCBox_SelectedValueChanged(CategoriesCBox, EventArgs.Empty);
    }

    private void LoadData(string FilePath) {
      string projectPath = Application.StartupPath + FilePath; // Шлях до моделі
```

```
    DataViewSchema modelSchema; // Схема моделі

    ITransformer model = mlContext.Model.Load(projectPath, out modelSchema); // Завантаження навченої моделі
    predictionEngine = mlContext.Model.CreatePredictionEngine<SmartHomeData, SmartHomePrediction>(model); // Створення двигуна прогнозування
  }

  private void RunBtn_Click(object sender, EventArgs e) {
    if (timer1.Enabled) {
      timer1.Enabled = false;
      RunBtn.Text = «Запустити»;
    } else {
      timer1.Enabled = true;
      RunBtn.Text = «Зупинити»;
    }
  }

  private void CategoriesCBox_SelectedValueChanged(object sender, EventArgs e) {
    if (isCategoriesLoaded) {
      neuralNetworkSelected = neuralNetworkProvider.SelectedNeuralNetworkByCategoriesId(Convert.ToInt32(CategoriesCBox.SelectedValue)); // Вибір нейронної мережі за ID категорії
      LoadData(neuralNetworkSelected.NeuralNetworkFileModel); // Завантаження даних моделі
    }
  }

  private void timer1_Tick(object sender, EventArgs e) {
    var generatedData = smartHomeDataGenerator.GenerateRandomData(); // Генерація випадкових даних

    // Відображення згенерованих даних
    RaportTBox.Text = $»Згенеровані дані:\r\n Час початку: {generatedData.StartTime}\r\n « +
                $»Місцезнаходження: {generatedData.Location}\r\n « +
                $»Час доби: {generatedData.TimeOfDay}\r\n « +
                $»Об'єкт: {generatedData.Object}\r\n « +
                $»Поза: {generatedData.Posture}\r\n « +
                $»Тривалість: {generatedData.Duration}\r\n « +
                $»Діяльність: {generatedData.Activity}\r\n»;

    // Прогнозування за допомогою моделі
    var prediction = predictionEngine.Predict(generatedData);
    bool isPotentialIntrusion = prediction.IsPotentialIntrusion(); // Перевірка на потенційне вторгнення

    // Виведення результату прогнозу
    RaportTBox.Text += $»Передбачувана діяльність: {prediction.PredictedActivity}\r\n»;
    RaportTBox.Text += isPotentialIntrusion ? «Можливе вторгнення виявлено!» : «Вторгнення не виявлено.» + «\r\n»;

    // Прокручування текстового поля до останнього запису
    RaportTBox.SelectionStart = RaportTBox.Text.Length;
```

```
      RaportTBox.ScrollToCaret();
    }


  }
}

class SmartHomeDataGenerator {
  private static Random random = new Random(); // Генератор випадкових чисел
  private static List<string> locations = new List<string> { «Bedroom», «Bathroom», «Kitchen», «Toilet», «Front» }; // Місця розташування
  private static List<string> timesOfDay = new List<string> { «Morning», «Afternoon», «Evening», «Night» }; // Часи доби
  private static List<string> objects = new List<string> { «Hall-Bedroom door», «Hall-Bathroom door», «Plates cupboard», «Hall-Toilet door», «Frontdoor» }; // Об'єкти
  private static List<string> postures = new List<string> { «Sitting», «Standing» }; // Пози
  private static List<string> activities = new List<string> { «go to bed», «use toilet», «prepare Breakfast», «take shower», «leave house», «door forced», «window broken», «unauthorized entry» }; // Діяльності
  private static List<string> intrusionActivities = new List<string> { «door forced», «window broken», «unauthorized entry» }; // Діяльності, що вказують на вторгнення
  private static List<string> intrusionTimesOfDay = new List<string> { «Night» }; // Час доби, коли вторгнення ймовірніше

  public SmartHomeData GenerateRandomData() {
    bool isIntrusion = random.NextDouble() < 0.2; // Ймовірність генерації даних про вторгнення
    string generatedTime = GenerateRandomTime(isIntrusion); // Генерація випадкового часу
    return new SmartHomeData {
      StartTime = generatedTime,
      Location = locations[random.Next(locations.Count)],
      TimeOfDay = DetermineTimeOfDay(generatedTime),
      Object = objects[random.Next(objects.Count)],
      Posture = postures[random.Next(postures.Count)],
      Duration = random.Next(30, 600), // Тривалість випадкової діяльності
      Activity = isIntrusion ? intrusionActivities[random.Next(intrusionActivities.Count)] : activities[random.Next(activities.Count)] // Випадкова діяльність
    };
  }

  private string GenerateRandomTime(bool isIntrusion) {
    int hour = isIntrusion ? random.Next(0, 4) : random.Next(0, 24); // Година
    int minute = random.Next(0, 60); // Хвилина
    int second = random.Next(0, 60); // Секунда
    return $»{hour:D2}:{minute:D2}:{second:D2}»; // Форматування часу
  }

  private string DetermineTimeOfDay(string generatedTime) {
    var timeParts = generatedTime.Split(':');
    int hour = int.Parse(timeParts[0]);

    if (hour >= 6 && hour < 12) return «Morning»;
    if (hour >= 12 && hour < 17) return «Afternoon»;
```

```
    if (hour >= 17 && hour < 21) return «Evening»;
    return «Night»; // Визначення часу доби на основі години
  }
}
```

Лістинг 2. Код класу «GeneralSettingsProvider»

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SmartHouseApp.AppCode;

namespace SmartHouseApp.Providers {
  internal class GeneralSettingsProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings[«CONNECT»];

    public void InsertGeneralSettings(bool AlarmsInclude, DateTime AlarmsStart, DateTime
AlarmsEnd, bool SmokeInclude, bool TemperatureInclude, double TemperatureLowerLimit, double
TemperatureUpperLimit,
  bool HumidityInclude, double HumidityLowerLimit, double HumidityUpperLimit, bool
NoiseInclude) {
      string SqlString = @»INSERT INTO GeneralSettings (AlarmsInclude, AlarmsStart, AlarmsEnd,
SmokeInclude, TemperatureInclude, TemperatureLowerLimit, TemperatureUpperLimit,
    HumidityInclude, HumidityLowerLimit, HumidityUpperLimit, NoiseInclude)
    VALUES (@AlarmsInclude, @AlarmsStart, @AlarmsEnd, @SmokeInclude,
@TemperatureInclude, @TemperatureLowerLimit, @TemperatureUpperLimit,
    @HumidityInclude, @HumidityLowerLimit, @HumidityUpperLimit, @NoiseInclude)»;

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue(«@AlarmsInclude», AlarmsInclude);
          cmd.Parameters.AddWithValue(«@AlarmsStart», AlarmsStart);
          cmd.Parameters.AddWithValue(«@AlarmsEnd», AlarmsEnd);
          cmd.Parameters.AddWithValue(«@SmokeInclude», SmokeInclude);
          cmd.Parameters.AddWithValue(«@TemperatureInclude», TemperatureInclude);
          cmd.Parameters.AddWithValue(«@TemperatureLowerLimit», TemperatureLowerLimit);
          cmd.Parameters.AddWithValue(«@TemperatureUpperLimit», TemperatureUpperLimit);
          cmd.Parameters.AddWithValue(«@HumidityInclude», HumidityInclude);
          cmd.Parameters.AddWithValue(«@HumidityLowerLimit», HumidityLowerLimit);
          cmd.Parameters.AddWithValue(«@HumidityUpperLimit», HumidityUpperLimit);
          cmd.Parameters.AddWithValue(«@NoiseInclude», NoiseInclude);
          conn.Open();
          cmd.ExecuteNonQuery();
        }
      }
    }
```

```
  public List<GeneralSettings> GetAllGeneralSettings() {
    int i = 0;
    string SqlString = «SELECT * FROM GeneralSettings»;

    List<GeneralSettings> listGeneralSettings = new List<GeneralSettings>();
    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
        conn.Open();
        using (SqlDataReader reader = cmd.ExecuteReader()) {
          while (reader.Read()) {
            GeneralSettings oneGeneralSettings = new GeneralSettings();
            oneGeneralSettings.Number = ++i;
            oneGeneralSettings.GeneralSettingsId = Convert.ToInt32(reader[«GeneralSettingsId»]);
            oneGeneralSettings.AlarmsInclude = Convert.ToBoolean(reader[«AlarmsInclude»]);
            oneGeneralSettings.AlarmsStart = Convert.ToDateTime(reader[«AlarmsStart»]);
            oneGeneralSettings.AlarmsEnd = Convert.ToDateTime(reader[«AlarmsEnd»]);
            oneGeneralSettings.SmokeInclude = Convert.ToBoolean(reader[«SmokeInclude»]);
            oneGeneralSettings.TemperatureInclude =
Convert.ToBoolean(reader[«TemperatureInclude»]);
            oneGeneralSettings.TemperatureLowerLimit =
Convert.ToDouble(reader[«TemperatureLowerLimit»]);
            oneGeneralSettings.TemperatureUpperLimit =
Convert.ToDouble(reader[«TemperatureUpperLimit»]);
            oneGeneralSettings.HumidityInclude = Convert.ToBoolean(reader[«HumidityInclude»]);
            oneGeneralSettings.HumidityLowerLimit =
Convert.ToDouble(reader[«HumidityLowerLimit»]);
            oneGeneralSettings.HumidityUpperLimit =
Convert.ToDouble(reader[«HumidityUpperLimit»]);
            oneGeneralSettings.NoiseInclude = Convert.ToBoolean(reader[«NoiseInclude»]);
            listGeneralSettings.Add(oneGeneralSettings);
          }
        }
      }
    }

    if (listGeneralSettings.Count == 0) {
      GeneralSettings noGeneralSettings = new GeneralSettings();
      noGeneralSettings.GeneralSettingsId = 0;
      noGeneralSettings.Message = NamesMy.NoDataNames.NoDataInGeneralSettings;
      listGeneralSettings.Add(noGeneralSettings);
    }
    return listGeneralSettings;
  }

  public GeneralSettings SelectGeneralSettings(int generalSettingsId) {
    GeneralSettings settings = null;
    string sqlString = «SELECT * FROM GeneralSettings WHERE GeneralSettingsId =
@GeneralSettingsId»;

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(sqlString, conn)) {
```

```
        cmd.Parameters.AddWithValue(«@GeneralSettingsId», generalSettingsId);
        conn.Open();
        using (SqlDataReader reader = cmd.ExecuteReader()) {
          if (reader.Read()) {
            settings = new GeneralSettings {
              GeneralSettingsId = Convert.ToInt32(reader[«GeneralSettingsId»]),
              AlarmsInclude = Convert.ToBoolean(reader[«AlarmsInclude»]),
              AlarmsStart = Convert.ToDateTime(reader[«AlarmsStart»]),
              AlarmsEnd = Convert.ToDateTime(reader[«AlarmsEnd»]),
              SmokeInclude = Convert.ToBoolean(reader[«SmokeInclude»]),
              TemperatureInclude = Convert.ToBoolean(reader[«TemperatureInclude»]),
              TemperatureLowerLimit = Convert.ToDouble(reader[«TemperatureLowerLimit»]),
              TemperatureUpperLimit = Convert.ToDouble(reader[«TemperatureUpperLimit»]),
              HumidityInclude = Convert.ToBoolean(reader[«HumidityInclude»]),
              HumidityLowerLimit = Convert.ToDouble(reader[«HumidityLowerLimit»]),
              HumidityUpperLimit = Convert.ToDouble(reader[«HumidityUpperLimit»]),
              NoiseInclude = Convert.ToBoolean(reader[«NoiseInclude»])
            };
          }
        }
      }

      return settings;
    }


    public void UpdateGeneralSettings(bool AlarmsInclude, DateTime AlarmsStart, DateTime
AlarmsEnd, bool SmokeInclude, bool TemperatureInclude, double TemperatureLowerLimit, double
TemperatureUpperLimit,
   bool HumidityInclude, double HumidityLowerLimit, double HumidityUpperLimit, bool
NoiseInclude, int GeneralSettingsId) {
      string sqlString = @»UPDATE GeneralSettings SET
        AlarmsInclude = @AlarmsInclude,
        AlarmsStart = @AlarmsStart,
        AlarmsEnd = @AlarmsEnd,
        SmokeInclude = @SmokeInclude,
        TemperatureInclude = @TemperatureInclude,
        TemperatureLowerLimit = @TemperatureLowerLimit,
        TemperatureUpperLimit = @TemperatureUpperLimit,
        HumidityInclude = @HumidityInclude,
        HumidityLowerLimit = @HumidityLowerLimit,
        HumidityUpperLimit = @HumidityUpperLimit,
        NoiseInclude = @NoiseInclude
        WHERE GeneralSettingsId = @GeneralSettingsId»;

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(sqlString, conn)) {
          cmd.Parameters.AddWithValue(«@AlarmsInclude», AlarmsInclude);
          cmd.Parameters.AddWithValue(«@AlarmsStart», AlarmsStart);
          cmd.Parameters.AddWithValue(«@AlarmsEnd», AlarmsEnd);
          cmd.Parameters.AddWithValue(«@SmokeInclude», SmokeInclude);
```

```
            cmd.Parameters.AddWithValue(«@TemperatureInclude», TemperatureInclude);
            cmd.Parameters.AddWithValue(«@TemperatureLowerLimit», TemperatureLowerLimit);
            cmd.Parameters.AddWithValue(«@TemperatureUpperLimit», TemperatureUpperLimit);
            cmd.Parameters.AddWithValue(«@HumidityInclude», HumidityInclude);
            cmd.Parameters.AddWithValue(«@HumidityLowerLimit», HumidityLowerLimit);
            cmd.Parameters.AddWithValue(«@HumidityUpperLimit», HumidityUpperLimit);
            cmd.Parameters.AddWithValue(«@NoiseInclude», NoiseInclude);
            cmd.Parameters.AddWithValue(«@GeneralSettingsId», GeneralSettingsId);
            conn.Open();
            cmd.ExecuteNonQuery();
          }
        }
      }


    }
  }
  public class GeneralSettings {
    private int _Number;
    private int _GeneralSettingsId;
    private bool _AlarmsInclude;
    private DateTime _AlarmsStart;
    private DateTime _AlarmsEnd;
    private bool _SmokeInclude;
    private double _TemperatureLowerLimit;
    private double _TemperatureUpperLimit;
    private bool _TemperatureInclude;
    private bool _HumidityInclude;
    private double _HumidityLowerLimit;
    private double _HumidityUpperLimit;
    private bool _NoiseInclude;
    private string _Message;

    public GeneralSettings() {
      _Number = 0;
      _GeneralSettingsId = 0;
      _AlarmsInclude = false;
      _AlarmsStart = new DateTime();
      _AlarmsEnd = new DateTime();
      _SmokeInclude = false;
      _TemperatureLowerLimit = 0.0;
      _TemperatureUpperLimit = 0.0;
      _TemperatureInclude = false;
      _HumidityInclude = false;
      _HumidityLowerLimit = 0.0;
      _HumidityUpperLimit = 0.0;
      _NoiseInclude = false;
      _Message = String.Empty;
    }

    public int Number {
      set { _Number = value; }
```

```
  get { return _Number; }
}
public int GeneralSettingsId {
  set { _GeneralSettingsId = value; }
  get { return _GeneralSettingsId; }
}
public bool AlarmsInclude {
  set { _AlarmsInclude = value; }
  get { return _AlarmsInclude; }
}
public DateTime AlarmsStart {
  set { _AlarmsStart = value; }
  get { return _AlarmsStart; }
}
public DateTime AlarmsEnd {
  set { _AlarmsEnd = value; }
  get { return _AlarmsEnd; }
}

public bool SmokeInclude {
  set { _SmokeInclude = value; }
  get { return _SmokeInclude; }
}
public double TemperatureLowerLimit {
  set { _TemperatureLowerLimit = value; }
  get { return _TemperatureLowerLimit; }
}
public double TemperatureUpperLimit {
  set { _TemperatureUpperLimit = value; }
  get { return _TemperatureUpperLimit; }
}
public bool TemperatureInclude {
  set { _TemperatureInclude = value; }
  get { return _TemperatureInclude; }
}
public bool HumidityInclude {
  set { _HumidityInclude = value; }
  get { return _HumidityInclude; }
}
public double HumidityLowerLimit {
  set { _HumidityLowerLimit = value; }
  get { return _HumidityLowerLimit; }
}
public double HumidityUpperLimit {
  set { _HumidityUpperLimit = value; }
  get { return _HumidityUpperLimit; }
}
public bool NoiseInclude {
  set { _NoiseInclude = value; }
  get { return _NoiseInclude; }
}
public string Message {
```

```
    set { _Message = value; }
    get { return _Message; }
  }

}
```
Лістинг 3. Код класу «SmartHomeSensorDataProvider»
```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SmartHouseApp.Providers {
  internal class SmartHomeSensorDataProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings[«CONNECT»];

    // Метод для вставки даних сенсорів у базу даних
    public void InsertSmartHomeSensorData(SmartHomeSensorData data) {
      string SqlString = «INSERT INTO SmartHomeSensorData (Timestamp, IsMotionDetected,
Temperature, IsDoorOpen, IsWindowOpen, LightLevel, IsThereNoise, Humidity, CameraFeedPath,
AirQualityIndex, Room) Values(@Timestamp, @IsMotionDetected, @Temperature, @IsDoorOpen,
@IsWindowOpen, @LightLevel, @IsThereNoise, @Humidity, @CameraFeedPath,
@AirQualityIndex, @Room)»;

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue(«@Timestamp», data.Timestamp);
          cmd.Parameters.AddWithValue(«@IsMotionDetected», data.IsMotionDetected);
          cmd.Parameters.AddWithValue(«@Temperature», data.Temperature);
          cmd.Parameters.AddWithValue(«@IsDoorOpen», data.IsDoorOpen);
          cmd.Parameters.AddWithValue(«@IsWindowOpen», data.IsWindowOpen);
          cmd.Parameters.AddWithValue(«@LightLevel», data.LightLevel);
          cmd.Parameters.AddWithValue(«@IsThereNoise», data.IsThereNoise);
          cmd.Parameters.AddWithValue(«@Humidity», data.Humidity);
          cmd.Parameters.AddWithValue(«@CameraFeedPath», data.CameraFeedPath ??
(object)DBNull.Value); // Handle null values
          cmd.Parameters.AddWithValue(«@AirQualityIndex», data.AirQualityIndex);
          cmd.Parameters.AddWithValue(«@Room», data.Room);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    // Метод для отримання всіх даних сенсорів з бази даних
    public List<SmartHomeSensorData> GetAllSmartHomeSensorData() {
```

```
    string SqlString = «SELECT * FROM SmartHomeSensorData ORDER BY Timestamp DESC»; // Чи інший критерій сортування

    List<SmartHomeSensorData> listAllSensorData = new List<SmartHomeSensorData>();
    using (SqlConnection conn = new SqlConnection(_ConnString)) {
     using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      conn.Open();
      using (SqlDataReader reader = cmd.ExecuteReader()) {
       while (reader.Read()) {
        SmartHomeSensorData oneSensorData = new SmartHomeSensorData();
        // Припускаючи, що клас SmartHomeSensorData має відповідний конструктор
        oneSensorData.Timestamp = reader.GetDateTime(reader.GetOrdinal(«Timestamp»));
        oneSensorData.IsMotionDetected = reader.GetBoolean(reader.GetOrdinal(«IsMotionDetected»));
        oneSensorData.Temperature = reader.GetDouble(reader.GetOrdinal(«Temperature»));
        oneSensorData.IsDoorOpen = reader.GetBoolean(reader.GetOrdinal(«IsDoorOpen»));
        oneSensorData.IsWindowOpen = reader.GetBoolean(reader.GetOrdinal(«IsWindowOpen»));
        oneSensorData.LightLevel = reader.GetDouble(reader.GetOrdinal(«LightLevel»));
        oneSensorData.IsThereNoise = reader.GetBoolean(reader.GetOrdinal(«IsThereNoise»));
        oneSensorData.Humidity = reader.GetDouble(reader.GetOrdinal(«Humidity»));
         oneSensorData.CameraFeedPath = reader.IsDBNull(reader.GetOrdinal(«CameraFeedPat;»)) ? null : reader.GetString(reader.GetOrdinal(«CameraFeedPath»));
        oneSensorData.AirQualityIndex = reader.GetDouble(reader.GetOrdinal(«AirQualityIndex»));
         oneSensorData.Room = reader.GetString(reader.GetOrdinal(«Room»));
        listAllSensorData.Add(oneSensorData);
       }
      }
      conn.Close();
     }
    }

    return listAllSensorData;
   }

   // Метод для вибору конкретних даних сенсора за ідентифікатором
   public SmartHomeSensorData SelectSmartHomeSensorDataById(int id) {
    string SqlString = «SELECT * FROM SmartHomeSensorData WHERE Id=@Id»;

    SmartHomeSensorData sensorData = null;
    using (SqlConnection conn = new SqlConnection(_ConnString)) {
     using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
      cmd.Parameters.AddWithValue(«@Id», id);
      conn.Open();
      using (SqlDataReader reader = cmd.ExecuteReader()) {
       if (reader.Read()) {
        sensorData = new SmartHomeSensorData {
         // Припускається, що клас SmartHomeSensorData має відповідний конструктор або публічні сеттери
         // Наповнення даними з reader...
        };
       }
      }
```

```
        conn.Close();
      }
    }
    return sensorData;
  }

  // Метод для оновлення даних сенсора
  public void UpdateSmartHomeSensorData(SmartHomeSensorData data) {
    string SqlString = @»
      UPDATE SmartHomeSensorData
      SET
        Timestamp = @Timestamp,
        IsMotionDetected = @IsMotionDetected,
        Temperature = @Temperature,
        IsDoorOpen = @IsDoorOpen,
        IsWindowOpen = @IsWindowOpen,
        LightLevel = @LightLevel,
        IsThereNoise = @IsThereNoise,
        Humidity = @Humidity,
        CameraFeedPath = @CameraFeedPath,
        AirQualityIndex = @AirQualityIndex,
        Room = @Room
      WHERE Id = @Id»;
    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue(«@Timestamp», data.Timestamp);
        cmd.Parameters.AddWithValue(«@IsMotionDetected», data.IsMotionDetected);
        cmd.Parameters.AddWithValue(«@Temperature», data.Temperature);
        cmd.Parameters.AddWithValue(«@IsDoorOpen», data.IsDoorOpen);
        cmd.Parameters.AddWithValue(«@IsWindowOpen», data.IsWindowOpen);
        cmd.Parameters.AddWithValue(«@LightLevel», data.LightLevel);
        cmd.Parameters.AddWithValue(«@IsThereNoise», data.IsThereNoise);
        cmd.Parameters.AddWithValue(«@Humidity», data.Humidity);
        cmd.Parameters.AddWithValue(«@CameraFeedPath», (object)data.CameraFeedPath ??
DBNull.Value); // Use DBNull for null values
        cmd.Parameters.AddWithValue(«@AirQualityIndex», data.AirQualityIndex);
        cmd.Parameters.AddWithValue(«@Room», data.Room);
        cmd.Parameters.AddWithValue(«@SmartHomeSensorDataId», data.SmartHomeSensorDataId);

        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }


  // Метод для видалення даних сенсора за ідентифікатором
  public void DeleteSmartHomeSensorDataById(int SmartHomeSensorDataId) {
    string SqlString = «DELETE FROM SmartHomeSensorData WHERE
SmartHomeSensorDataId=@SmartHomeSensorDataId»;
```

```
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.Parameters.AddWithValue(«@SmartHomeSensorDataId», SmartHomeSensorDataId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

  }
}

public class SmartHomeSensorData {
  public int SmartHomeSensorDataId { get; set; }
  public DateTime Timestamp { get; set; }
  public bool IsMotionDetected { get; set; }
  public double Temperature { get; set; }
  public bool IsDoorOpen { get; set; }
  public bool IsWindowOpen { get; set; }
  public double LightLevel { get; set; }
  public bool IsThereNoise { get; set; }
  public double Humidity { get; set; }
  public string CameraFeedPath { get; set; }
  public double AirQualityIndex { get; set; }
  public string Room { get; set; }

  public SmartHomeSensorData() { }

  // Конструктор класу
  public SmartHomeSensorData(DateTime timestamp, bool isMotionDetected, double temperature,
                             bool isDoorOpen, bool isWindowOpen, double lightLevel,
                             bool isThereNoise, double humidity, string cameraFeedPath,
                             double airQualityIndex, string room) {
    Timestamp = timestamp;
    IsMotionDetected = isMotionDetected;
    Temperature = temperature;
    IsDoorOpen = isDoorOpen;
    IsWindowOpen = isWindowOpen;
    LightLevel = lightLevel;
    IsThereNoise = isThereNoise;
    Humidity = humidity;
    CameraFeedPath = cameraFeedPath;
    AirQualityIndex = airQualityIndex;
    Room = room;
  }

}
```

Додаток Б. Скрипти створення бази даних

```
USE [master]
GO
/****** Object:  Database [DB]    Script Date: 22.02.2024 15:01:17 ******/
CREATE DATABASE [DB]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'DB', FILENAME = N'C:\Program Files (x86)\Microsoft SQL
Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\DB.mdf' , SIZE = 5120KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'DB_log', FILENAME = N'C:\Program Files (x86)\Microsoft SQL
Server\MSSQL12.SQLEXPRESS\MSSQL\DATA\DB_log.ldf' , SIZE = 1024KB , MAXSIZE =
2048GB , FILEGROWTH = 10%)
GO
ALTER DATABASE [DB] SET COMPATIBILITY_LEVEL = 120
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [DB].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Categories](
	[CategoriesId] [int] IDENTITY(1,1) NOT NULL,
	[CategoriesName] [nvarchar](220) NULL,
	[Description] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[CategoriesId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[CoapDatas]    Script Date: 22.02.2024 15:01:17 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[CoapDatas](
	[CoapDatasId] [int] IDENTITY(1,1) NOT NULL,
	[Mid] [float] NULL,
	[Code] [float] NULL,
	[OptDesc] [float] NULL,
	[LocationQuery] [float] NULL,
	[MaxAge] [float] NULL,
	[UriHost] [float] NULL,
	[ResponseIn] [float] NULL,
	[Retransmitted] [float] NULL,
	[Token] [float] NULL,
	[TokenLen] [float] NULL,
```

```
	[Class] [bit] NULL,
	[CategoriesId] [int] NULL,
	[IsGuessed] [bit] NULL,
PRIMARY KEY CLUSTERED
(
	[CoapDatasId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[GeneralSettings]    Script Date: 22.02.2024 15:01:17 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[GeneralSettings](
	[GeneralSettingsId] [int] IDENTITY(1,1) NOT NULL,
	[AlarmsInclude] [bit] NULL,
	[AlarmsStart] [datetime] NULL,
	[AlarmsEnd] [datetime] NULL,
	[SmokeInclude] [bit] NULL,
	[TemperatureInclude] [bit] NULL,
	[TemperatureLowerLimit] [float] NULL,
	[TemperatureUpperLimit] [float] NULL,
	[HumidityInclude] [bit] NULL,
	[HumidityLowerLimit] [float] NULL,
	[HumidityUpperLimit] [float] NULL,
	[NoiseInclude] [bit] NULL,
	[CategoriesId] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[GeneralSettingsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Logs]    Script Date: 22.02.2024 15:01:17 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Logs](
	[LogsId] [int] IDENTITY(1,1) NOT NULL,
	[UsersId] [int] NULL,
	[EventNameShow] [nvarchar](max) NULL,
	[EventDate] [datetime] NULL,
	[UsersName] [nvarchar](100) NULL,
PRIMARY KEY CLUSTERED
(
	[LogsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

```
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[NeuralNetwork]    Script Date: 22.02.2024 15:01:17 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[NeuralNetwork](
	[NeuralNetworkId] [int] IDENTITY(1,1) NOT NULL,
	[NeuralNetworkNames] [nvarchar](200) NULL,
	[CategoriesId] [int] NULL,
	[NeuralNetworkFileModel] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[NeuralNetworkId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[SmartHomeSensorData]    Script Date: 22.02.2024 15:01:17 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[SmartHomeSensorData](
	[SmartHomeSensorDataId] [int] IDENTITY(1,1) NOT NULL,
	[Timestamp] [datetime] NULL,
	[IsMotionDetected] [bit] NULL,
	[Temperature] [float] NULL,
	[IsDoorOpen] [bit] NULL,
	[IsWindowOpen] [bit] NULL,
	[LightLevel] [float] NULL,
	[IsThereNoise] [bit] NULL,
	[Humidity] [float] NULL,
	[CameraFeedPath] [nvarchar](max) NULL,
	[AirQualityIndex] [float] NULL,
	[Room] [nvarchar](100) NULL,
PRIMARY KEY CLUSTERED
(
	[SmartHomeSensorDataId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Users]    Script Date: 22.02.2024 15:01:17 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Users](
	[UsersId] [int] IDENTITY(1,1) NOT NULL,
	[FirstName] [nvarchar](50) NULL,
```

```
	[LastName] [nvarchar](50) NULL,
	[UsersName] [nvarchar](50) NULL,
	[UsersPassword] [nvarchar](max) NULL,
	[RoleId] [int] NULL,
	[Description] [nvarchar](max) NULL,
	[Email] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[UsersId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
USE [master]
GO
ALTER DATABASE [DB] SET  READ_WRITE
GO
```