**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

15.03 — КМР. 1939 –“С” 2022.12.30. 023 ПЗ

КИРИШУНА ДМИТРІЯ АНАТОЛІЙОВИЧА

2023 р.

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

**Факультет інформаційних технологій**

**УДК 004.9:728-049.5**

| **«ПОГОДЖЕНО»** | **«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»** |
|---|---|
| Декан факультету інформаційних технологій | Завідувач кафедри комп'ютерних наук |
| Глазунова О.Г., д.п.н., професор | Голуб Б.Л., к.т.н., доцент |
| _____ | _____ |
| **____ _____ 2023 р.** | **____ _____ 2023 р.** |

**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

**на тему** Дослідження системи моніторингу безпеки будівлі

Спеціальність 121 Інженерія програмного забезпечення
(код і назва)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

**Гарант освітньої програми**

_____ _____ _____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

**Керівник магістерської кваліфікаційної роботи**

канд. техн. наук, доцент _____ Ткаченко О. М.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

**Виконав** _____ Киришун Д. А.
(підпис) (ПІБ студента)

**КИЇВ-2023**

**НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ**

**І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет (ННІ)** _____Інформаційних технологій_____

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри** ___комп'ютерних наук__
_____
_____к.т.н., доцент_____ _____ _Голуб Б. Л._
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"____" _____20_____року

**З А В Д А Н Н Я**

**ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ**

_____Киришуну Дмитрію Анатолійовичу_____
(прізвище, ім'я, по батькові)

Спеціальність _ __121 Інженерія програмного забезпечення_____
(код і назва)

Освітня програма ___Програмне забезпечення інформаційних систем_____
(назва)

Орієнтація освітньої програми__ Освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи___Дослідження системи моніторингу безпеки будівлі_____
_____

затверджена наказом ректора НУБіП України від "_30_"__грудня__2023 р. №_1939 «С»_

Термін подання завершеної роботи на кафедру __ 2023.11.05_____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи __Дані з датчиків безпеки: Це дані, які вимірюються датчиками температури, вологості, датчиками диму, датчиками руху, дверей і вікон. Ці дані включають значення показників, час їх вимірювання та іншу відповідну інформацію._____
_____
_____

Перелік питань, що підлягають дослідженню:

1. Концепції розумних будинків та їхніх систем безпеки_____
2. Огляд сучасних методів забезпечення безпеки в розумних будинках _____
3. Оцінка ключових ризиків для безпеки розумних будинків _____

Перелік графічного матеріалу (за потреби) _____
_____

**Дата видачі завдання** "_____"_____ 20____ р.

**Керівник магістерської кваліфікаційної роботи** _____ __Ткаченко О. М.__
( підпис ) (прізвище та ініціали)

**Завдання прийняв до виконання** _____ _____Киришун Д. А._____
(підпис ) (прізвище та ініціали студента)

**ЗМІСТ**

# ВСТУП

Системи моніторингу безпеки будівель відіграють критичну роль у сучасному суспільстві, де прискорена урбанізація та зростання комплексності інфраструктур вимагають все більш вдосконалених методів забезпечення безпеки. Основні задачі таких систем полягають у виявленні, моніторингу та реагуванні на загрози, такі як пожежі, злом, технічні несправності тощо. Проте, сучасний стан цих систем залишає бажати кращого — вони часто є розрізненими, неефективними та вимагають великих витрат на встановлення та обслуговування.

Забезпечення надійності та ефективності систем моніторингу безпеки будівель є не тільки технологічним викликом, але й соціальною необхідністю. Крім того, дана проблематика має велику економічну значущість, оскільки відсутність ефективних систем може призвести до великих матеріальних втрат та, що найгірше, загрози життю людей.

Враховуючи вищезгадані аспекти, проведення дослідження системи моніторингу безпеки будівлі є не тільки актуальним, але й необхідним для підвищення якості життя, ефективності виробництва та забезпечення національної безпеки.

Сучасний етап розвитку технічних наук, зокрема в галузі інформаційних технологій та кібернетики, відкриває нові можливості для вдосконалення систем моніторингу безпеки. Водночас, зростаюча кількість катастрофічних подій, зумовлених техногенними факторами, робить цю тему надзвичайно актуальною. Через обмеженість ресурсів та відсутність єдиної методології, існуючі системи не завжди забезпечують належний рівень безпеки.

З огляду на геополітичну ситуацію та економічні умови в Україні, проблема набуває ще більшої ваги. Дослідження в даній області може забезпечити значущий внесок у розвиток безпекових технологій, які можуть бути адаптовані для конкретних умов нашої країни.

Актуальність теми полягає не лише в розробці нових методів та технологій, але й в можливості їх інтеграції з існуючими системами. Критичний

аналіз та порівняння з відомими розв'язаннями показують, що незважаючи на наявність численних досліджень в цій області, є значний простір для наукових та технічних вдосконалень.

Метою даного дослідження є розробка програмного забезпечення для моніторингу безпеки будівель, яка б відзначалась високою надійністю, ефективністю та адаптивністю до змінюваних умов експлуатації.

Для досягнення поставленої мети необхідно вирішити наступний комплекс задач:

– аналіз існуючих систем моніторингу безпеки. Вивчення наукової літератури, патентів та інших джерел інформації для формування зрозумілого уявлення про стан справ в даній області;

– визначення ключових параметрів, метрик безпеки та критеріїв, на основі яких буде оцінюватися ефективність системи моніторингу;

– вибір технологій розробки;

– створення прототипу системи. Створення робочої моделі системи на основі вибраного технічного рішення, у тому числі тестування та оцінка ефективності.

– проведення випробувань перевірки надійності та ефективності розробленої системи.

Об'єктом даного дослідження є будівлі різного призначення.

Предметом дослідження є програмне забезпечення для моніторингу системи безпеки в будівлях.

Для досягнення поставленої мети та вирішення конкретних задач дослідження було використано комплекс методів, які дозволяють глибоко аналізувати і розробляти системи моніторингу безпеки будівель:

– аналітичний метод був застосований для вивчення наукової літератури, патентів та інших джерел інформації з метою аналізу існуючих систем і технологій. Цей метод дозволив сформувати загальний огляд проблеми та ідентифікувати ключові напрямки для подальших досліджень;

– метод моделювання був використаний для створення концептуальної та технічної моделі системи моніторингу безпеки. Це включало в себе визначення основних компонентів системи та їх взаємодії;

– статистичний метод використовувався для аналізу даних, отриманих в ході експериментальних досліджень. Це дозволило оцінити ефективність розробленої системи та визначити можливі вектори для подальшого вдосконалення.

З урахуванням високого ступеня готовності та практичної значущості, рекомендується розглянути можливість впровадження даної системи на різних типах об'єктів — від житлових будинків до промислових комплексів та державних установ.

# 1 ОСНОВНІ ТЕОРЕТИЧНІ ПРИНЦИПИ ТА ДІАГНОСТИКА СТАНУ ІСНУЮЧИХ СИСТЕМ БЕЗПЕКИ У РОЗУМНИХ БУДИНКАХ

## 1.1 Концепції розумних будинків та їхніх систем безпеки

Розумний будинок представляє собою інтегровану систему автоматизації, що об'єднує в собі широкий спектр пристроїв та технологічних рішень для оптимізації життєдіяльності мешканців в аспектах комфорту, енергоефективності, та безпеки. Основні характеристики системи "Розумного Будинку" можна розділити на:

– мультидисциплінарна інтеграція. Система здійснює комплексну інтеграцію різноманітних функціональних блоків, зокрема систем освітлення, терморегуляції, вентиляції, кондиціонування повітря, енергозбереження, домашньої електроніки, та систем безпеки;

– процесна автоматизація. Дана характеристика передбачає використання алгоритмічних методів для автоматизації рутинних та сложних процесів. Мешканці мають можливість програмувати індивідуальні сценарії роботи системи, що можна керувати через спеціалізоване програмне забезпечення або мобільні додатки;

– адаптивність та машинне навчання. Система вбудовує алгоритми машинного навчання, які аналізують поведінку мешканців та зміни в зовнішньому середовищі для оптимізації параметрів роботи. Це забезпечує високий рівень персоналізації та адаптивності системи;

– Інтеграція з Інтернетом Речей (IoT). Розумний будинок активно взаємодіє з екосистемою Інтернету Речей, що дозволяє здійснювати ефективний збір, обмін та аналіз даних. Це не тільки поліпшує функціональні можливості системи, але й забезпечує підвищення рівня безпеки та комфорту мешканців [1].

Можна сказати, що розумний будинок є високотехнологічною інтегрованою системою, яка синтезує передові досягнення в областях автоматизації, машинного навчання, та Інтернету Речей для створення

оптимального мікроклімату, енергоефективності та безпеки житлового простору.

Система безпеки розумного будинку представляє собою критично важливий елемент, дизайн якого орієнтований на мінімізацію ризиків та забезпечення фізичної інтегральності мешканців та їхнього матеріального добра.

Основні компоненти системи безпеки складаються із:

– сенсорна інфраструктура. В основі системи лежать сенсори та датчики, що специфічно розроблені для детекції різноманітних параметрів та подій, включаючи, але не обмежуючись, рух, стан дверей та вікон, температурні аномалії, наявність газів, водяні витоки, та акустичні відхилення, що можуть вказувати на несанкціонований втручання або інші аварійні ситуації;

– центр обробки даних. Контрольний пристрій функціонує як центральний вузол, який забезпечує обробку даних, зібраних сенсорною інфраструктурою, та реалізує відповідні керівні команди для активації захисних механізмів або систем попередження;

– відеоаналітичні системи. Компоненти відеоспостереження та відеоаналітики служать для візуального моніторингу просторового контексту будинку, дозволяючи не лише фіксувати події, але й реалізовувати алгоритмічний аналіз поведінки осіб та об'єктів в кадрі;

– системи сповіщення та комунікації. Системи аудіо- та візуального сповіщення, а також комунікаційні протоколи, забезпечують швидке та ефективне оповіщення мешканців та, у разі потреби, автоматичне викликання служб екстреної реакції;

– комунікаційні шлюзи. Використовуються спеціалізовані комунікаційні шлюзи та протоколи для інтеграції всіх перелічених компонентів у єдину, гомогенну систему безпеки, яка, у свою чергу, є інтегрованою частиною загальної системи розумного будинку.

Отже, система безпеки розумного будинку представляє собою комплексну, мультикомпонентну структуру, здатну реагувати на широкий спектр потенційних загроз та аварійних ситуацій, що забезпечує високий рівень захисту та комфорту для мешканців.

Система "розумний будинок" може мати два основних архітектурних підходи: централізована та децентралізована структура. У рамках централізованої моделі, ключові компоненти, включаючи елементи управління, центральний обчислювальний контролер, та виконавчі пристрої, інтегровані у єдину телекомунікаційну інфраструктуру (рис. 1.1). Це дозволяє забезпечити координоване управління та передачу керівних сигналів та команд [2].

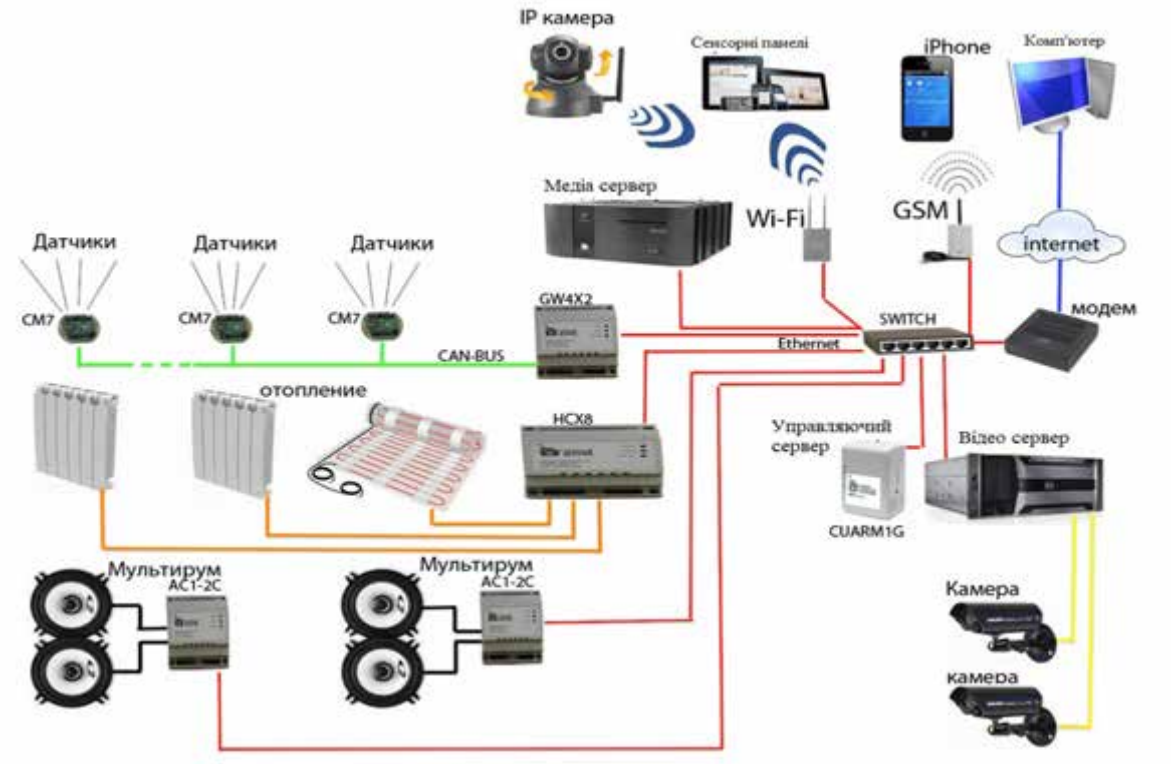


Рис. 1.1 – Концептуальна схема централізованої архітектури

Елемент управління виконує роль інтерфейсу між користувачем та системою. Управління може здійснюватися через широкий спектр девайсів, починаючи від традиційних пультів дистанційного управління, сенсорних панелей, до смартфонів. Додатково, автоматичне управління може реалізовуватися за допомогою датчиків різних типів: освітленості, присутності, температурних, вологості та інших.

Центральний контролер представляє собою обчислювальний модуль, що забезпечує централізоване управління системою. Він виконує аналіз та обробку даних від елементів управління та датчиків, зберігає в пам'яті керівні сценарії, задані користувачем або автоматично генеровані програмами, та відправляє виконавчі команди до відповідних пристроїв системи.

Отже, централізована архітектура "розумного будинку" є інтегрованою структурою, в якій центральний контролер відіграє роль "мозку" системи, а елементи управління та датчики служать органами чуття та взаємодії з користувачем. Це дозволяє досягти високого рівня координації, ефективності та гнучкості у відгуках на змінювані умови та потреби користувачів [3].

Основні переваги централізованої системи "розумний будинок" можна сформулювати наступним чином:

– єдність управління. Централізована система забезпечує уніфіковане управління всіма складовими "розумного будинку" через єдиний центральний контролер. Це спрощує процес конфігурації та моніторингу системи;

– висока координація. Завдяки центральному управлінню, можливе ефективне координування між різними пристроями та системами, що дозволяє реалізовувати складні сценарії роботи без несумісності або затримок;

– централізована обробка даних. Всі дані з датчиків та інших пристроїв збираються і обробляються в єдиному місці, що підвищує швидкість реакції на події та можливість комплексного аналізу даних;

– гнучкість та масштабованість. Централізована система легше адаптується до змін у потребах користувача або до додавання нових функціональних елементів, оскільки все управління відбувається через єдиний контролюючий модуль;

– високий рівень безпеки. Захист від несанкціонованого доступу та інших загроз може бути більш ефективно реалізований на рівні єдиного центрального контролера, що дозволяє використовувати високі стандарти криптографічної безпеки [4].

В рамках децентралізованої архітектури системи "розумний будинок", концепція управління зміщується від єдиного центрального контролера до розподіленої логіки виконання, яка імплементується на рівні окремих пристроїв та модулів (рис. 1.2).

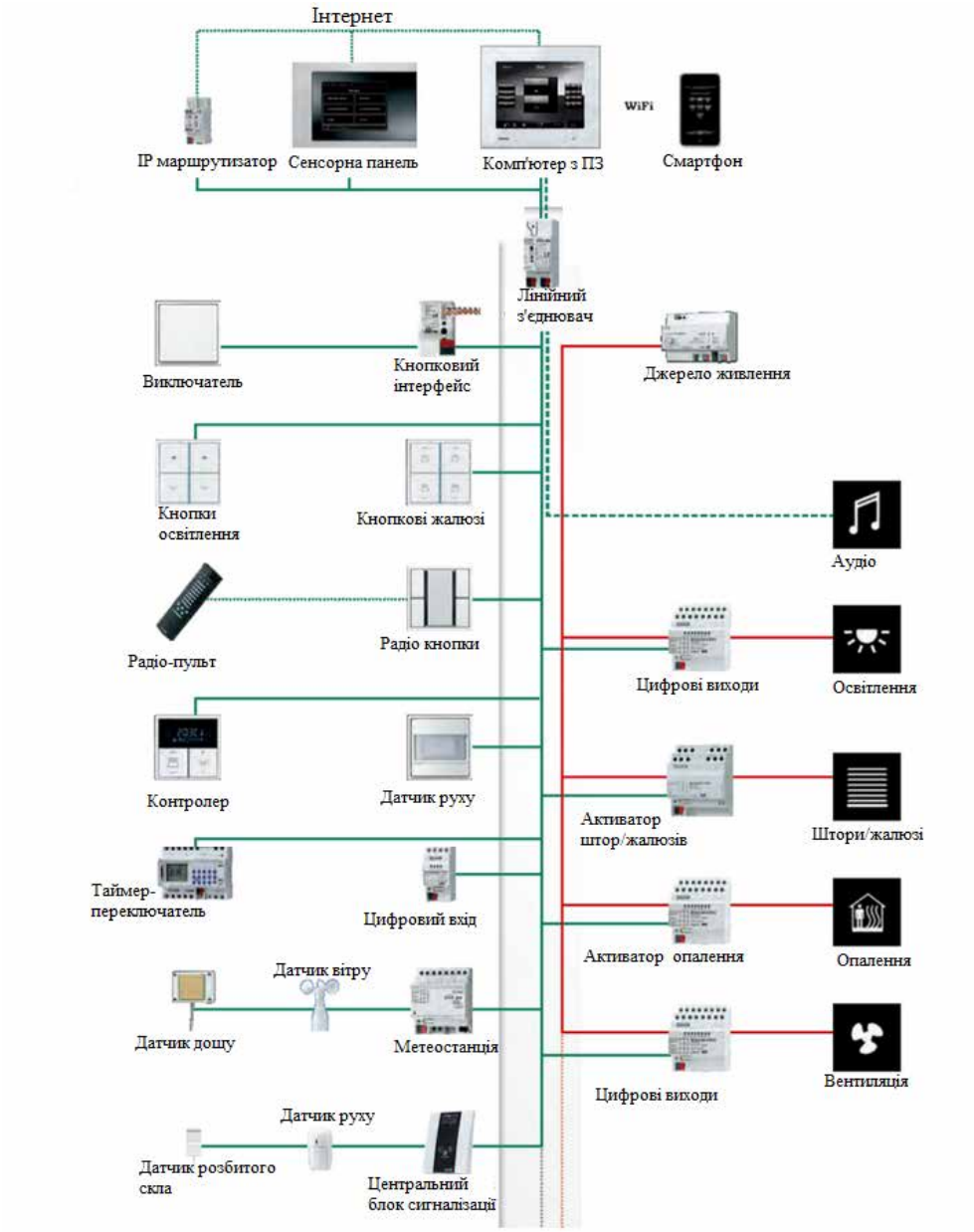


Рис. 1.2 – Схематичне зображення децентралізованої архітектури

У децентралізованій архітектурі системи "розумний будинок", датчики не просто фіксують зміни в зовнішньому середовищі або відхилення від заданих параметрів; вони також ініціюють відповідні команди до виконавчих пристроїв. Ці команди, в свою чергу, активуються через так звані активатори, які можуть

бути електромеханічними пристроями, реле або іншими виконавчими механізмами.

Цей підхід забезпечує певну гнучкість системи, оскільки кожний пристрій або модуль може адаптуватися до змін умов автономно, без потреби глобального переконфігурування. Така архітектура може пропонувати більший діапазон можливостей для кастомізації та адаптації пристроїв до специфічних потреб користувача.

Проте, варто зазначити, що децентралізований підхід може мати певні обмеження та виклики. Одним з них є потенційна низька ефективність системи, викликана необхідністю координувати взаємодію між розподіленими елементами. Крім того, розподілена логіка може ускладнити процес централізованого моніторингу та керування, а також може вимагати більш складних алгоритмів для забезпечення гармонійної роботи всієї системи.

Основні переваги децентралізованої системи автоматизації "розумний дім" можна охарактеризувати наступним чином:

– автономність компонентів. Кожен пристрій або модуль в системі може функціонувати незалежно, що робить систему більш стійкою до локальних збоїв та відмов;

– гнучкість та адаптивність. Децентралізована логіка дозволяє легко адаптувати систему до нових умов або змінених потреб користувача без необхідності глобального переконфігурування;

– швидка реакція на зміни. Оскільки рішення приймаються на локальному рівні, система може швидше реагувати на динамічні зміни в середовищі;

– зниження ризиків централізованих збоїв. Відсутність єдиного точкового вузла збою (центрального контролера) підвищує загальну надійність та стійкість системи;

– масштабованість. Додавання нових пристроїв або модулів є менш трудомістким, оскільки кожен новий елемент може бути інтегрований в систему без змін в центральній логіці управління;

– розподілена обробка даних. Збір та аналіз даних може відбуватися паралельно на кількох пристроях, що може підвищити ефективність обробки даних та виведення висновків.

Децентралізована система "розумний дім" також, пропонує ряд стратегічних переваг, які можуть бути виразно корисними в залежності від конкретних потреб та сценаріїв використання [5].

## 1.2 Дослідження використання методів штучного інтелекту в системах розумної безпеки

Архітектура інтелектуальних систем, яка базується на принципах штучного інтелекту, представляє собою комплексну інтеграцію фізичних пристроїв Internet of Things (IoT), сенсорної техніки, актуаторів та мережевих інфраструктур. Ці елементи взаємодіють з високопродуктивними обчислювальними системами та аналітичними інструментами на базі штучного інтелекту з метою автоматизації та оптимізації управлінських процесів [6].

Основні складові архітектури включають:

– фізичні пристрої IoT. Це устаткування, що відзеркалює взаємодію з навколишнім середовищем через датчики різного спектра — температурні, вологісні, оптичні, акустичні тощо;

– мережеві протоколи. Канали передачі даних можуть бути як провідними, так і безпровідними, включаючи стандарти Wi-Fi, Bluetooth, LTE та інші. Ці канали забезпечують зв'язок між пристроями IoT та обчислювальними центрами;

– обчислювальні ресурси. Ці ресурси можуть бути розміщені на реброві серверах, що знаходяться в безпосередній близькості до джерел даних, або на віддалених хмарних платформах, що забезпечують велику обчислювальну потужність для аналізу великих даних;

– аналітичні засоби на основі штучного інтелекту. Методи машинного та глибокого навчання дозволяють системі робити висновки на

основі аналізу даних, виявляти закономірності, прогнозувати події та автоматично адаптувати управлінські стратегії.

Таким чином, архітектура інтелектуальних систем представляє собою синтез фізичного та віртуального середовищ, інтегрований через мережеві технології та узгоджений за допомогою алгоритмів штучного інтелекту. Це створює умови для створення високоадаптивних, автоматизованих та інтелектуально-управлінських систем [7].

В сучасній архітектурі інтелектуальних систем, заснованих на принципах штучного інтелекту, створюється специфічна екосистема. Вона об'єднує пристрої Internet of Things, мережеві інфраструктури та обчислювальні платформи, здатні до виконання завдань на основі аналітики AI. Ця інтеграція дозволяє реалізувати автоматизоване управління, моніторинг та оптимізацію в різних доменах застосування— від домашньої автоматизації до індустріальних систем і транспортних рішень.

На наведеному прикладі архітектури RL-IoT (рис. 1.3) можна побачити, як модуль навчання з підкріпленням (Reinforcement Learning, RL) спрямований на досягнення конкретних цілей через оптимізацію налаштувань пристроїв. Тут "мета" концептуалізується як послідовність станів і дій, які пристрій або система має пройти для досягнення бажаного результату. Така архітектура відкриває широкі можливості для навчання системи, адаптації до нових умов та ефективного реагування на динамічні зміни в середовищі.

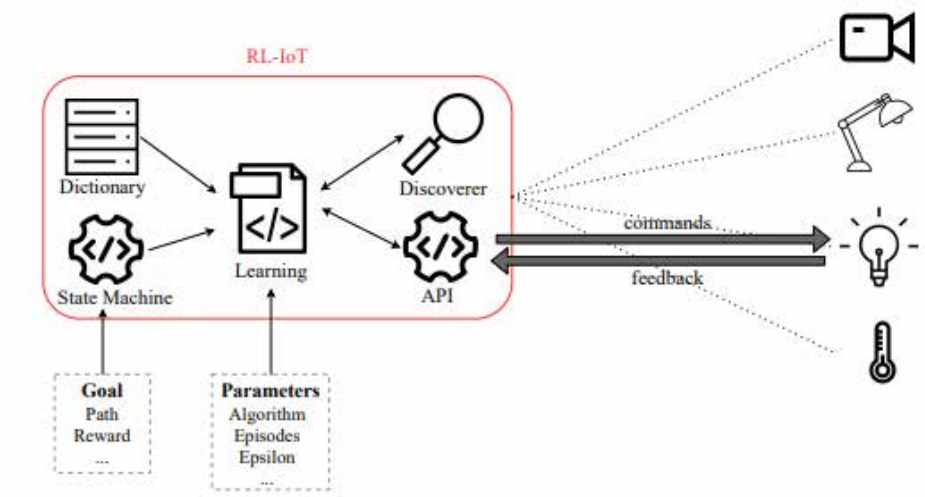

Рис. 1.3 – Огляд структури RL-IoT

В системі RL-IoT існує внутрішній лексикон комунікаційних повідомлень, який охоплює арсенал доступних команд для пристроїв Інтернету речей. Цей лексикон може бути сформований на основі офіційних специфікацій протоколів, через реверс-інжиніринг або шляхом аналізу мережевого трафіку. Він універсальний у тому сенсі, що може агрегувати команди з різних протоколів, від різних виробників та версій.

Використовуючи передові методики навчання з підкріпленням, модуль RL у системі RL-IoT здійснює конструкцію та актуалізацію внутрішнього автомата станів. Він застосовує ряд алгоритмів, зокрема Q-Learning і SARSA, кожен із специфічними параметрами, для визначення оптимальних команд з лексикону, спрямованих на зміну стану пристроїв Інтернету речей для досягнення бажаного результату. Оцінка ефективності кожної дії відбувається на основі функції винагороди, яка є унікальною для кожного потенційного шляху до цілі [8].

Система також має додаткові модулі для забезпечення її функціональності. Модуль "Discoverer" відповідає за виявлення пристроїв Інтернету речей в локальній мережі, використовуючи стандартні методи сканування, такі як nmap4. По завершенню процесу виявлення, модуль Socket API забезпечує абстракцію механізмів комунікації для взаємодії з виявленими пристроями. Цей модуль не лише здійснює відправку команд, але і може обробляти вхідні повідомлення від пристроїв, якщо це дозволено їхньою архітектурою .

Концепція Інтернету речей і автономних систем керування виходять із різних парадигм і їхнє впровадження не обов'язково залежить одне від одного. Однак, поняття автономного Інтернету речей, або AIoT, представляє собою єднання цих двох напрямків і розглядається як нова етапна віха в розвитку Інтернету речей, спрямована на дослідження додаткового потенціалу цієї технології (рис. 1.4).

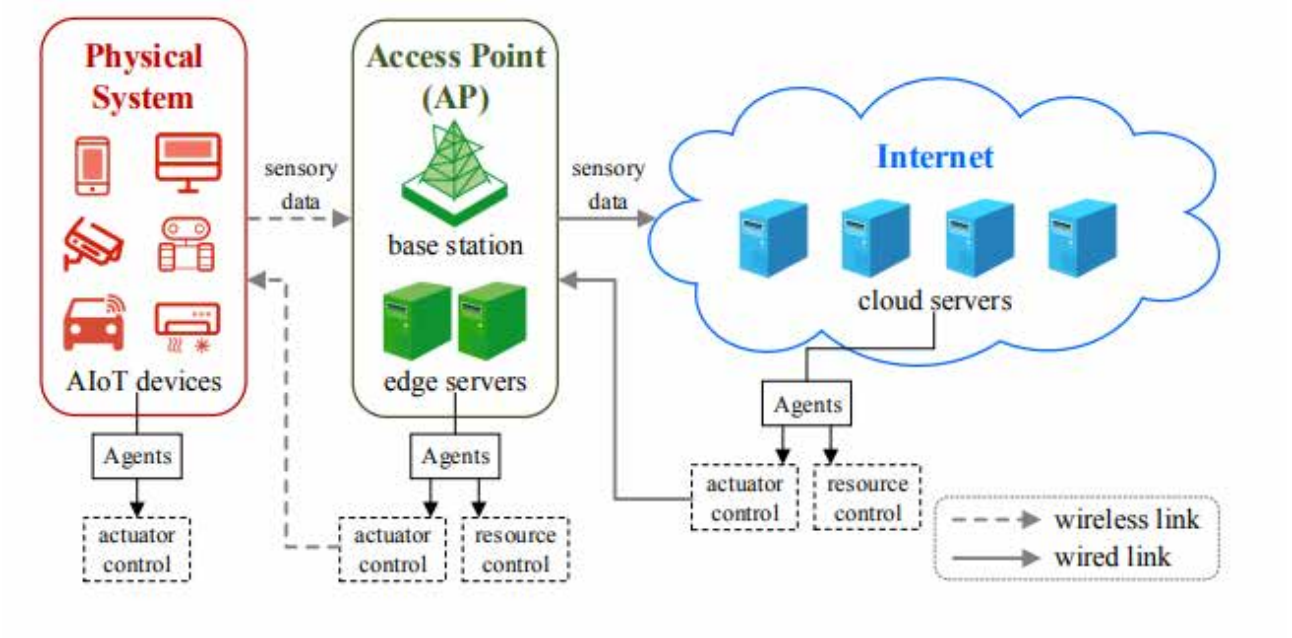


Рис. 1.4 – Концепція автономної системи Інтернету речей

В архітектурі AIoT, інтелектуальні пристрої формують динамічне середовище, в якому вони взаємодіють з фізичним світом через сенсори та виконавчі механізми. Ці пристрої, зазвичай, з'єднані з Інтернетом через мобільні базові станції або інші точки доступу, які також можуть містити обмежені обчислювальні ресурси для проміжної обробки даних. Отримані сенсорні дані аналізуються для прийняття рішень, які потім реалізуються через виконавчі пристрої. Така обробка може відбуватися як на самому пристрої, так і на віддалених хмарних серверах [9].

Модель бездротової мережі сенсорів і виконавчих пристроїв, або WSAN, можна розглядати як підмножину або спрощену версію AIoT. В цій моделі, сенсори збирають дані про навколишнє середовище, в той час як виконавчі пристрої реагують на це середовище на основі отриманих даних. Всі компоненти мережі взаємодіють через бездротові канали. Завдяки алгоритмам підсиленого глибокого навчання, агент в системі WSAN отримує інформацію від сенсорів, приймає рішення та надсилає керуючі команди до виконавчих пристроїв, змінюючи таким чином динаміку взаємодії з навколишнім середовищем.

У контексті AIoT, користувачі мають справу з більш інтегрованою та комплексною екосистемою, ніж у моделі WSAN. AIoT екосистема охоплює не лише ідентифікацію та сприйняття, але і розширює свою діяльність на комунікаційні технології, обчислювальні можливості та різноманітні сервіси.

В структурі AIoT можна виділити три ключові компоненти. По-перше, шар сприйняття, який функціонує як інтерфейс між фізичним світом та цифровою інфраструктурою. Цей шар включає пристрої Інтернету речей з сенсорами та виконавчими механізмами, які взаємодіють з навколишнім середовищем для збору даних та реалізації керуючих дій.

По-друге, мережевий шар, який служить для забезпечення комунікаційної взаємодії між різними пристроями Інтернету речей та обчислювальними ресурсами. Цей шар може включати в себе бездротові мережі доступу, мобільні базові станції та інші точки доступу до Інтернету. Ці елементи мережі не лише забезпечують трансфер даних, але і можуть виконувати роль шлюзів для з'єднання з ребровими, туманними або хмарними серверами.

По-третє, шар додатків, який представляє собою обчислювальні ресурси, розташовані на серверах, та відповідає за фінальну обробку та зберігання даних, а також за прийняття оптимальних керуючих рішень. Цей шар може використовувати розширені методи аналітики та штучного інтелекту для ефективного аналізу даних та автоматизації процесів [10].

В контексті методологій підсиленого та глибокого підсиленого навчання, агент є концептуальною одиницею, яка знаходиться в активній взаємодії з окремими елементами свого екосистемного середовища. Основна мотивація агента полягає в тому, що агент систематично здійснює реєстрацію різних станів середовища, в якому він опинився, та на основі отриманих від цього середовища винагород виробляє тактику вибору оптимальних дій. Універсальність цього процесу дозволяє його адаптацію на різних рівнях ієрархії архітектури системи, що, в свою чергу, зображено на рис. 1.5. Слід зазначити, що агент може бути розміщений та функціонувати в різних шарах,

від фізичного до абстрактного, в залежності від специфіки задачі та архітектурних рішень.

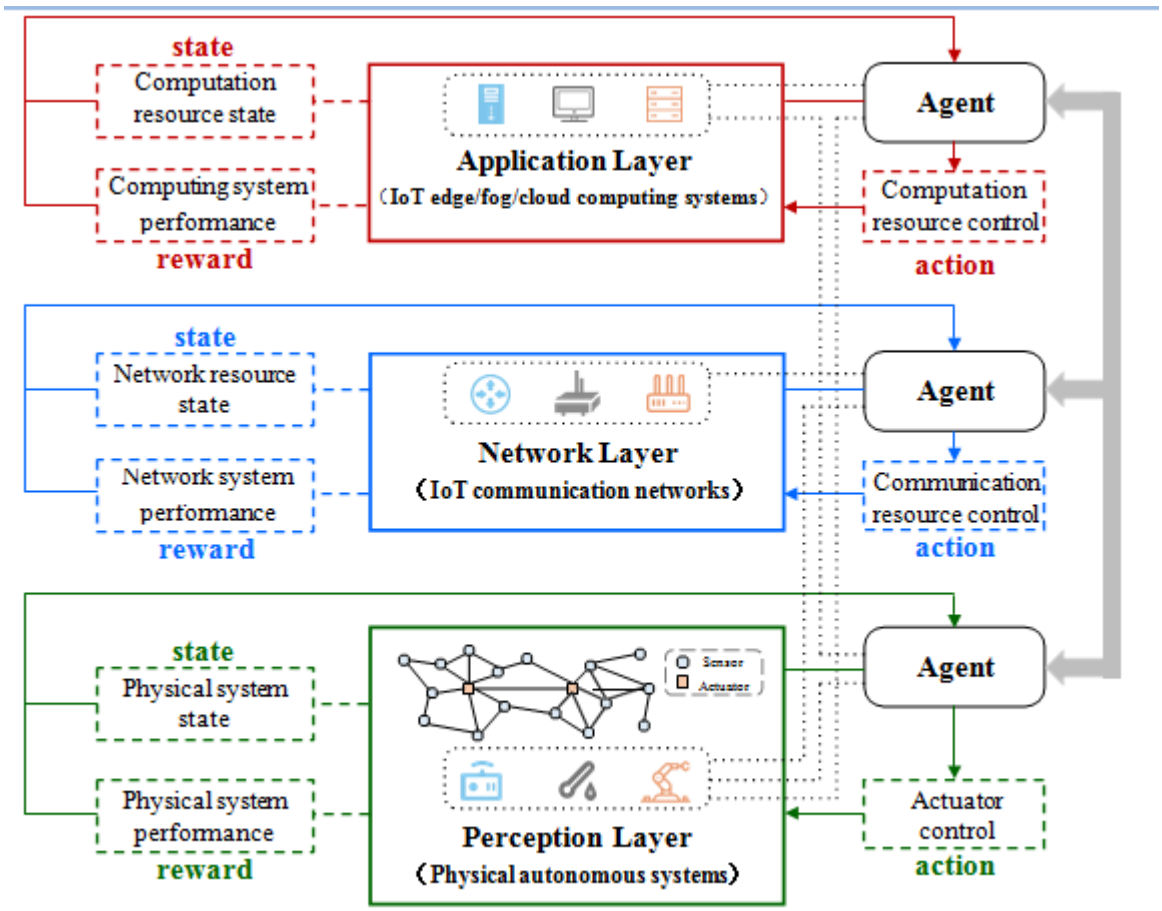


Рис. 1.5 – Загальна модель RL/DRL для автономного IoT

В пристроях Інтернету речей агент зазвичай займається збором даних через сенсори та реалізацією керуючих дій через виконавчі механізми. У бездротових точках доступу агент може оптимізувати мережевий трафік, розподіляти ресурси або забезпечувати безпеку комунікацій. На рівні хмарних серверів агент може виконувати більш складні завдання, такі як аналітика великих даних або взаємодія з різноманітними сервісами та системами.

На рис. 1.5 приведено розподіл агентів по логічних шарах та їх фізичних розташувань в контексті методів підсиленого та глибокого підсиленого навчання (RL/DRL). Агенти, що функціонують в шарі сприйняття, можуть бути інтегровані як у пристроях Інтернету речей, так і на хмарних серверах. Агенти мережевого шару знаходяться часто в бездротових точках доступу та можуть бути присутніми в пристроях IoT для забезпечення Device-to-Device

комунікації. Щодо агентів шару додатків, їх основне розташування — хмарні сервери, хоча вони також можуть бути присутніми в пристроях IoT для виконання специфічних завдань [11].

У випадку колаборації агентів з різних шарів, необхідно забезпечити ефективний обмін інформацією та координацію політик. Так, мережевий шар може надавати дані про мережеву затримку до шару сприйняття для більш точної моделі стану, а шар сприйняття, в свою чергу, може передавати свої оптимізаційні цілі для формулювання функції винагороди в мережевому шарі. У ситуаціях, коли фізичне розташування агентів співпадає, можна розглядати концепцію уніфікованого логічного агента, який відображає функціональність агентів з різних шарів.

Щодо методології DRL, вона складається з двох ключових компонентів: підсиленого навчання (RL) та глибокого навчання (DL). В RL виклик полягає в потребі великого об'єму пам'яті для збереження функцій значень та Q-функцій, особливо при великих просторах станів, що часто є характерним для реальних систем. Глибоке навчання приходить на допомогу, дозволяючи апроксимувати ці функції з значно меншим набором параметрів, тим самим роблячи систему більш обчислювально ефективною. Ця синергія між RL та DL приводить до формування більш потужних систем DRL.

У дослідженні методів глибокого підсиленого навчання (DRL) можна виокремити дві фундаментальні категорії алгоритмів: алгоритми, засновані на оцінках значення функцій, та алгоритми, що використовують градієнт політики. Вони різняться за механізмами наближення, використовуючи для цього архітектури нейронних мереж для апроксимації або функцій значення/Q-функцій, або ж функцій політики.

Однією з ключових специфічних задач, для яких застосовуються алгоритми DRL, є проблема виявлення аномалій. Ці алгоритми здійснюють скрупульозний аналіз вхідних даних, ідентифікуючи аномальні, девіаційні або підозрілі шаблони, що можуть свідчити про потенційну атаку або системні вразливості. Методики виявлення аномалій можуть бути реалізовані через

статистичні моделі, архітектури нейронних мереж, алгоритми класифікації або ж через гібридні методи, які комбінують декілька з цих підходів.

Додатково, в контексті безпеки системи, релевантним є алгоритм виявлення вторгнень (IDS). Даний алгоритм має на меті аналіз мережевого трафіку або системних логів з метою ідентифікації спроб несанкціонованого доступу, актів вторгнення чи інших форм аномальної активності. В залежності від конкретного сценарію, IDS може використовувати ряд підходів, включаючи правилові системи, евристичні моделі, техніки машинного навчання або комбінації згаданих методів для ефективного виявлення потенційно небезпечних дій [12].

### 1.3 Оцінка ключових ризиків для безпеки розумних будинків

Інтелектуальні системи безпеки в розумних будинках, незважаючи на свою технічну вдосконаленість, не є цілковито захищеними від потенційних вразливостей. Ці вразливості можуть бути використані зловмисниками для несанкціонованого доступу до системи та персональних даних користувачів. Тому, для всебічного розуміння цього аспекту, необхідно звернути увагу на ключові категорії загроз.

*Фізичні загрози*

Фізичні загрози знаходяться в сфері безпосереднього впливу на мешканців та їхню матеріальну власність. До цієї категорії можна віднести неавторизований доступ до житлової площі з намірами крадіжки або нанесення шкоди особам. Пожежні ризики, пов'язані з короткими замиканнями або несправністю побутової техніки, також підпадають під цю категорію. Додатково, неправильна експлуатація газових приладів або дефекти в газопровідній системі можуть стати причиною витоків газу, що загрожують вибухом або отруєнням. Загрози затоплення виникають через пошкодження водопровідної системи, протікання дахів та інші водні аварії. Крім того, несправність у системах життєзабезпечення, таких як опалення, вентиляція або

електропостачання, може призвести до зниження комфорту проживання та, у крайньому випадку, до загрози життю мешканців .

*Приватність та юридичні питання*

Експансія інтелектуальних систем безпеки та моніторингових технологій в домогосподарствах супроводжується складними проблемами приватності та юридичними викликами. Під чвс збору та обробки великих обсягів персональних даних, системи такого роду можуть стати інструментом порушення конфіденційності мешканців. Отже, необхідно дотримуватися національного та міжнародного законодавства щодо захисту персональних даних.

Щодо відеоспостереження, розміщення камер у приватних та публічних зонах може викликати юридичні дискусії відносно прав сусідів та інших осіб, які можуть потрапити в поле зору камер. Такий підхід вимагає обережного аналізу з точки зору права на приватність.

Додатково, існує потенціал зловживання цими системами, наприклад, для нелегітимного слідкування, шпигунства або незаконного контролю над поведінкою мешканців. Такі ситуації представляють не лише етичні, але й юридичні проблеми.

На завершення, варто звернути увагу на необхідність строгого дотримання юридичних норм та регулятивних актів, що стосуються експлуатації систем безпеки, механізмів відеоспостереження та умов зберігання даних. Це являє собою істотний аспект, який повинен бути належним чином врахований власниками інтелектуальних будинків.

*Кібернетичні загрози*

Кібернетичні ризики стають критичним фактором у контексті захисту інформаційних та управлінських систем інтелектуальних будинків. Серед найзначущіших кібернетичних загроз можна виокремити:

несанкціоноване проникнення в систему. Зловмисники мають змогу здійснити незаконний доступ до систем через недоліки в аутентифікації, вразливості в мережевій інфраструктурі або програмному забезпеченні;

– шкідливе ПЗ та віруси. Атакуючи системи, зловмисники можуть застосовувати шкідливе програмне забезпечення для крадіжки цінної інформації або втручання в нормальну роботу систем;

– атаки типу "відмова в обслуговуванні" (DoS). Ці атаки полягають у перевантаженні мережевої інфраструктури або серверів, що може призвести до зупинки роботи системи та унеможливити віддалене управління;

– інтерцепція та перехоплення даних. Неавторизоване перехоплення комунікацій між компонентами системи може призвести до витоку конфіденційної інформації або неправильної інтерпретації керуючих сигналів;

– фішинг та методики соціальної інженерії. Використання обманливих технік для отримання конфіденційних даних користувачів, таких як паролі або особиста інформація, яка може бути використана для доступу до систем.

Ці кібернетичні загрози вимагають комплексного підходу до інформаційної безпеки, який об'єднує технічні, організаційні та юридичні аспекти.

*Соціальні та психологічні аспекти загроз*

Окрім технологічних, юридичних та конфіденційних аспектів, системи безпеки інтелектуального домогосподарства мають не менш важливий вплив на соціальні та психоемоційні сфери життя осіб, які проживають в таких будинках.

Передусім, сталий відеонагляд та контроль можуть підривати психоемоційний комфорт, породжуючи стрес та почуття перманентної обережності у мешканців. Це може впливати на їхнє психічне здоров'я та загальний рівень життєзадоволення.

Автоматизація систем безпеки та їх віддалене управління може призвести до ерозії соціальних взаємин між сусідами та в місцевій громаді. Адже коли люди менше залежать від безпосереднього, традиційного спілкування для забезпечення власної безпеки, це може знизити ступінь соціальної інтеракції.

Зрештою, збільшення залежності від автоматизованих систем безпеки може компрометувати автономію мешканців, збільшуючи їхню технологічну

залежність у сфері безпеки. Така залежність може мати негативні наслідки для психоемоційного стану осіб та їхньої здатності до самостійного рішення проблем [13].

### 1.4 Огляд сучасних методів забезпечення безпеки в розумних будинках

З розширенням сфери впровадження розумних технологій, питання безпеки в розумних будинках набуває особливої актуальності. Інтелектуальні системи безпеки, розроблені для таких будинків, фокусуються на протидії різноманітним загрозам, включаючи ризик несанкціонованого доступу до користувацьких даних та системного обладнання. Важливо звернути увагу на такі ключові методики та технології забезпечення безпеки, як:

*Традиційні системи безпеки*

Традиційні системи безпеки, включаючи алармні системи, датчики руху, а також дверні та віконні сенсори, не втратили своєї актуальності в контексті розумних будинків. Ці системи надають основоположний захист від небажаних проникнень, а також активують сповіщення для мешканців чи відповідних служб безпеки у випадку підозрілих дій або вторгнення. Однак важливо відзначити, що ці системи володіють певними обмеженнями. Найбільш виразно це проявляється у їхній вузькоспеціалізованій функціональності: вони не завжди ефективні проти сучасних кіберзагроз і часто не інтегровані належним чином з іншими компонентами інтелектуальної системи розумного будинку.

*Інтегровані системи безпеки розумного будинку*

В арсеналі розумних будинків інтегровані системи безпеки представляють собою синтез традиційних заходів захисту та сучасних технологій, включаючи відеоспостереження, системи контролю доступу, а також пожежну і водяну сигналізацію. Ці системи пропонують централізовану платформу управління безпекою, що дозволяє автоматично виявляти та реагувати на широкий спектр потенційних загроз, від несанкціонованого проникнення до пожежі або водяних аварій. Особливістю інтегрованих систем

є те, що вони обладнані функціональністю віддаленого моніторингу та управління, що може бути здійснено через смартфони та інші мобільні пристрої. Така можливість значно підвищує гнучкість та реактивність системи у відповідь на динамічні зміни у загрозовому середовищі.

*Кіберзахист розумного будинку*

У зв'язку з поширенням інтернет-з'єднань у розумних будинках, питання кібербезпеки набуває особливої актуальності. Для протидії потенційним кіберзагрозам використовуються різноманітні методи, серед яких можна виділити застосування безпечних мережевих протоколів, процедури аутентифікації користувачів та шифрування передаваних даних. Додаткові засоби захисту, такі як антивірусні рішення та мережеві файрволи, слугують додатковим рівнем оборони, що мінімізує ризики компрометації пристроїв розумного будинку через дії зловмисників або вплив шкідливого програмного забезпечення. Таким чином, кіберзахист стає інтегральною частиною системи загальної безпеки розумного будинку, що забезпечує її стійкість до широкого спектру потенційних загроз.

*Приватність та контроль доступу*

Забезпечення приватності та контролю доступу представляє собою ключовий елемент інтелектуальних систем безпеки в розумних будинках. Щоб зберегти конфіденційність мешканців, застосовуються сучасні технологічні рішення, такі як анонімізація персональних даних та розподілене зберігання інформації. Щодо контролю доступу, тут імплементуються різні ступені авторизації, що дозволяє регулювати допуск до окремих зон будинку або до специфічних функціональних можливостей системи. Така гранулярна система авторизації підвищує ступінь захисту, обмежуючи можливість несанкціонованого доступу для гостей та сервісних служб.

*Автоматизація та інтелектуальні алгоритми*

З використанням автоматизованих систем та високорозвинутих алгоритмів, зокрема технологій штучного інтелекту та машинного навчання, системи безпеки розумних будинків досягають нового рівня ефективності та

точності. Ці передові технології дозволяють не лише здійснювати комплексний аналіз даних, що надходять з різноманітних датчиків та камер відеоспостереження, але й виявляти аномальні патерни поведінки, що можуть сигналізувати про потенційні загрози. На основі такого аналізу, система може автоматично ініціювати заздалегідь визначені протоколи реагування, мінімізуючи тим самим ризик помилкових спрацьовувань та непотрібних втручань служб безпеки.

*Енергоефективність та сталий розвиток*

Врахування принципів енергоефективності та сталого розвитку в інтелектуальних системах безпеки розумних будинків відіграє стратегічно важливу роль. Зокрема, використання технологій енергозбереження, таких як сучасні світлодіодні рішення в освітленні, експлуатація сонячних панелей та аккумуляторних систем зберігання енергії, сприяє сталій функціональності системи безпеки з мінімальним впливом на екосистему. Ці аспекти можуть бути додатково оптимізовані через інтеграцію з системами управління енергією розумного будинку, що дозволяє враховувати загальний енергетичний баланс об'єкта, оптимізувати енергоспоживання та знижувати витрати на експлуатацію.

*Сумісність та інтеграція з іншими системами розумного будинку*

Синергія та взаємна інтеграція інтелектуальної системи безпеки з іншими компонентами інфраструктури розумного будинку, включаючи системи освітлення, теплопостачання, вентиляції та кондиціонування повітря (HVAC), а також мультимедійні та розважальні модулі, є невід'ємною частиною оптимізації зручності та ресурсоощадності. Це у свою чергу спрощує формування уніфікованої платформи управління, що керує всіма функціональними аспектами розумного будинку, підвищуючи тим самим ефективність управління ресурсами та забезпечуючи комплексний підхід до безпеки.

Розгляд відповідних ключових елементів є критичним для концептуалізації та реалізації дієвих інтелектуальних систем забезпечення

безпеки в приміщеннях, що підлягають автоматизації. Інтеграція цих компонентів у процес створення системи безпеки сприятиме формуванню захисної архітектури для розумного будинку, яка відповідає актуальним технічним та безпековим стандартам [14].

## 1.5 Аналіз недоліків сучасних систем безпеки та потреба в нових стратегіях

Системи забезпечення безпеки в автоматизованих приміщеннях, таких як розумні будинки чи промислові комплекси, відзначаються великим рівнем технічної складності. Це ставить перед науковою та інженерною спільнотами завдання пошуку вдосконалених методологій для забезпечення їхньої надійності та безпеки в експлуатації.

Варто зазначити, що ці системи не є позбавленими певних обмежень, які можуть стати факторами ризику. Ці обмеження можуть включати в себе технічні вразливості, що відкривають можливість для несанкціонованого доступу, а також обмеженість ресурсів для швидкої та ефективної реакції на непередбачувані події. До таких обмежень можна віднести:

*Обмежена гнучкість та адаптивність*

Обмежена здатність до гнучкості та адаптації є однією з інтринсичних характеристик традиційних систем безпеки в автоматизованих приміщеннях. Ці системи можуть проявляти недостатню ефективність при спробах пристосування до динамічних змін умов екстер'єрного середовища, емерджентних загроз безпеки або особливостей експлуатації з боку користувачів. Така недолік може знизити рівень захисту, що надається, або привести до неекономічно великих витрат на обслуговування та модернізацію системи.

*Відсутність комплексного підходу*

Недостатність комплексної стратегії є однією з проблематичних характеристик деяких сучасних систем безпеки для автоматизованих приміщень. Ці системи часто сфокусовані на вирішенні окремих задач безпеки,

таких як візуальний моніторинг або превенція пожеж, при цьому знехтуванням піддається комплексний аналіз широкого спектру потенційних загроз. Така неповність підходів може створити уразливі точки у системі безпеки, які, в свою чергу, можуть бути експлуатовані особами з недоброзичливими намірами для нелегітимного доступу або реалізації деструктивних сценаріїв.

*Недостатнє забезпечення кібербезпеки*

З посиленням інтеграції Інтернету речей і зростанням числа мережево підключених пристроїв у контексті інтелектуальних будівель, питання кібербезпеки набуває вираженого критичного характеру. На жаль, значна кількість діючих систем безпеки недостатньо адекватно адresyює цю складову, створюючи тим самим потенційні вектори для кібернетичних атак та інших форм несанкціонованого доступу.

*Відсутність інтелектуальних алгоритмів та автоматизації*

В контексті сучасних технологічних можливостей, недоліком багатьох наявних систем безпеки є невикористання інтелектуальних алгоритмів та автоматизованих механізмів, базованих на принципах штучного інтелекту та машинного навчання. Ця обставина призводить до ситуацій, коли системи не здатні ефективно ідентифікувати аномальні або підозрілі активності, що може в свою чергу призвести до помилкових алармів або, навпаки, непомічених інцидентів. Така недосконаліть робить системи менш ефективними та може призвести до небажаних незручностей для користувачів.

*Проблеми приватності та контролю доступу*

Дефіцит адекватних механізмів забезпечення приватності та регулювання доступу є актуальною проблемою в контексті багатьох сучасних систем безпеки розумних будинків. Відсутність гнучких налаштувань доступу та неефективні методи забезпечення конфіденційності можуть призвести до недопустимого втручання в приватний простір мешканців та незаконного доступу до конфіденційних даних чи критичних компонентів системи безпеки. Така ситуація не тільки створює ризики для конфіденційності інформації, але й може підривати загальну ефективність системи безпеки.

*Обмеження в інтеграції з іншими системами розумного будинку*

Недоліки в сумісності та можливостях інтеграції з іншими компонентами розумного будинку представляють серйозний виклик для багатьох сучасних систем безпеки. Відсутність плавної інтеграції між системами безпеки та іншими функціональними елементами розумного будинку, такими як системи управління енергією чи мультимедійні платформи, може суттєво ускладнювати процеси управління та нагляду. Така некоординованість не лише знижує користувацьку зручність, але й може неефективно використовувати ресурси, що, в свою чергу, підриває загальну функціональність та ефективність системи безпеки.

Враховуючи ідентифіковані обмеження та недосконалості в сфері систем безпеки розумних будинків, актуальністю набуває задача розробки інноваційних методологій та технологічних рішень. Такі новітні підходи мають орієнтуватися на забезпечення адаптивності і гнучкості в контексті постійно змінюваних загроз та умов середовища. Важливим є також імплементація комплексної системи захисту, що бере до уваги широкий спектр викликів безпеки розумного будинку.

Реалізація надійного кіберзахисту та захисту персональних даних користувачів є іншим критичним вектором розвитку. Це передбачає впровадження розумних алгоритмів та автоматизованих механізмів для оптимізації функціональності та зменшення числа помилкових спрацювань.

Інтегрованість з іншими системами розумного будинку стає ключовим аспектом, що спрощує координацію та цілісність роботи різних компонент. Система має також надавати гнучкі механізми для налаштування доступу та авторизації, що дозволить індивідуалізувати захист та приватність для різних категорій користувачів.

Таким чином, інтелектуальна система безпеки розумного будинку, що відповідає згаданим вимогам, повинна не лише гарантувати високий ступінь захисту від множини потенційних загроз, але й забезпечувати оптимальні умови для управління, моніторингу та адаптації системи до змінюваних

екзогенних факторів. Реалізація такого комплексу завдань вимагає впровадження передових технологій, що максимізують рівень безпеки та комфорту користувачів [15].

### 1.6 Формулювання задачі та ключові критерії для розробки інтелектуальної системи безпеки

Головним завданням даної кваліфікаційної роботи є створення інтелектуальної системи безпеки для розумного будинку, яка б купіровала слабкі місця сучасних систем, ефективно контрувала різні види загроз і була у відповідності з актуальними технічними стандартами та вимогами. Для реалізації цього амбітного завдання передбачено вирішення ряду специфічних підзадач, включаючи:

– діагностику потреб аудиторії користувачів та класифікація потенційних загроз, які мають місце в розумних будинках;

– критичний огляд наявних на ринку технологічних рішень у контексті систем безпеки, призначених для розумних будинків;

– ідентифікацію прогалин та недосконалостей в існуючих системах безпеки, а також обґрунтування необхідності впровадження інноваційних методів;

– формулювання концептуальних засад інтелектуальної системи безпеки, що має бути відповідною до сучасних технологічних трендів та нормативів;

– розробка алгоритмічної бази та протокольної взаємодії між компонентами системи з метою оптимізації її функціональності;

– конструювання та експериментальна перевірка прототипу розробленої інтелектуальної системи безпеки для розумних будинків.

Інтелектуальна система безпеки для розумного будинку має задовольняти ряд критичних вимог, котрі є вирішальними для її ефективності, надійності та придатності до використання:

– адаптивність та гнучкість. Система безпеки повинна володіти інтелектуальними здібностями до адаптації у реальному часі, враховуючи змінні умови та емерджентні загрози. Така гнучкість забезпечує динамічний захист та можливість персоналізації налаштувань відповідно до потреб користувачів;

– комплексний підхід до захисту. Система повинна реалізовувати мультидисциплінарний підхід, комбінуючи фізичні, кібернетичні та технологічні методи безпеки. Ця інтегративність є ключовою для забезпечення всебічного захисту від різноманітних видів загроз;

– надійність кібербезпеки та приватність. Захист даних та конфіденційність інформації користувачів має бути на найвищому рівні. Система повинна включати в себе надійні механізми кібербезпеки для захисту від несанкціонованого доступу та інших кіберзагроз;

– інтелектуальні алгоритми та автоматизація. Використання передових технологій штучного інтелекту та машинного навчання для аналізу даних, виявлення аномалій та автоматизованого реагування на загрози є критично важливим для підвищення ефективності системи;

– відповідність стандартам та регулятивним вимогам. Система безпеки повинна строго відповідати сучасним національним та міжнародним стандартам, а також регулятивним вимогам, що стосуються безпеки, приватності та інших аспектів.

Згадані аспекти формують концептуальну основу для розробки та імплементації ефективної, надійної та користувацьки-орієнтованої системи безпеки розумного будинку.

# 2 ПРОЕКТУВАННЯ КОНЦЕПТУАЛЬНИХ ЗАСАД І МЕТОДИК ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ БЕЗПЕКИ ДЛЯ РОЗУМНИХ БУДИНКІВ

## 2.1 Огляд ключових методів та технологій в інтелектуальних системах безпеки

Інтелектуальна система безпеки для розумного будинку інтегрує в себе різноманітні сучасні технології та методології з метою формування надійних та ефективних стратегій захисту. Одними із ключових технологічних компонентів є:

*Хмарні технології*

Використання хмарних технологій у системах безпеки розумних будинків відкриває нові можливості для віддаленого зберігання та обробки даних. Це не тільки розширює географію доступу до системи безпеки, дозволяючи користувачам управляти нею з різних точок світу, але й сприяє більш ефективному оновленню програмного забезпечення. Окрім того, хмарні рішення надають додатковий рівень резервного копіювання, що мінімізує ризи втрати або пошкодження важливих даних через локальні негаразди, такі як збої обладнання або фізичні пошкодження. Таким чином, хмарні технології не лише полегшують управління системою безпеки, але й підвищують її надійність та резилієнтність.

*Інтернет речі (IoT)*

В інтелектуальних системах безпеки розумних будинків ключову роль відіграє технологія Інтернету Речей. Ця технологія включає в себе ряд пристроїв, що мають можливість підключення до Інтернету та можливість взаємодії між собою. Це забезпечує динамічний обмін даними та координоване управління безпекою на просторі розумного будинку. Зокрема, завдяки IoT, власник будинку має змогу дистанційно моніторити та контролювати системи безпеки, отримуючи дані про стан будинку в реальному часі. Така архітектура не тільки розширює функціональні можливості системи безпеки, але і дозволяє

реагувати на потенційні загрози з максимальною швидкістю та ефективністю [16].

*Штучний інтелект (AI) та машинне навчання (ML)*

Використання технологій штучного інтелекту та машинного навчання в системах безпеки розумних будинків дозволяє суттєво підвищити ефективність аналітики та виявлення загроз. Штучний інтелект здатний обробляти великі обсяги даних, які надходять з різноманітних датчиків та камер відеоспостереження, і в реальному часі ідентифікувати аномальні патерни поведінки або ситуації, що можуть сигналізувати про загрозу. Це може включати розпізнавання осіб на відеозаписах, детекцію незвичайних акустичних сигналів або незвичних рухів у просторі будинку. Крім того, алгоритми машинного навчання можуть аналізувати історичні дані для прогнозування потенційних проблем у майбутньому, що дозволяє вживати запобіжних заходів ще до виникнення критичних ситуацій. Таким чином, штучний інтелект та машинне навчання не лише роблять системи безпеки більш реактивними, але і прогностично ефективними [17-18].

*Бездротові комунікаційні технології*

Застосування бездротових комунікаційних технологій, включаючи Wi-Fi, Zigbee, Z-Wave та Bluetooth, є ключовим фактором у формуванні адаптивних та гнучких систем безпеки для розумних будинків. Дані технології забезпечують взаємодію між індивідуальними компонентами системи безпеки та її центральним контролером. Однією з основних переваг цього підходу є можливість легкої та швидкої інсталяції та масштабування системи без необхідності фізичного прокладання кабельних мереж. Така архітектура дозволяє з легкістю додавати нові пристрої до системи або переконфігурувати існуючі, що сприяє адаптації системи до змінних умов та потреб користувача [19].

*Криптографічний захист*

В сфері інтелектуальних систем безпеки для розумних будинків криптографічні технології відіграють ключову роль у забезпеченні довіреності

та конфіденційності. Складні алгоритми шифрування використовуються для того, щоб гарантувати, що передана чи збережена інформація залишається недоступною для несанкціонованих осіб. Це сприяє не лише збереженню приватності користувачів, але й унеможливлює витік або компрометацію їх особистих та чутливих даних. За допомогою розширених методів аутентифікації можна додатково відфільтровувати доступ до системи, визначаючи, хто має право взаємодіяти з різними елементами системи безпеки. Такий підхід значно підвищує загальний рівень безпеки та довіреності в інтелектуальних системах безпеки [20].

*Геофенсинг*

Технологія геофенсингу дозволяє створювати віртуальні периметри навколо заданої території за допомогою геолокаційних служб. Коли користувач або його мобільний пристрій перетинає ці географічні межі, система безпеки розумного будинку може автоматично активуватися або деактивуватися. Це не лише зручно, але й підвищує рівень безпеки, оскільки система може бути налаштована на реагування на конкретні сценарії залежно від місцезнаходження користувача. Наприклад, система може автоматично включати внутрішню систему спостереження, якщо визначить, що користувач покинув територію розумного будинку. Такий підхід сприяє більш ефективному та гнучкому управлінню системою безпеки [21].

*Системи аналізу відео*

В інтелектуальних системах безпеки розумних будинків використовується досить комплексний аналіз відео, який перетворює просте відеоспостереження в потужний інструмент для забезпечення безпеки. За допомогою алгоритмів машинного навчання та штучного інтелекту, система може автоматично ідентифікувати та класифікувати об'єкти, людей або навіть конкретні дії, які відбуваються на відеозапису. Такий підхід дозволяє не лише виявляти рух або присутність людей, але й аналізувати їх поведінку для визначення можливих загроз. Наприклад, система може автоматично виявити незвичайну активність або нерегулярні рухи, що можуть свідчити про наміри

злому або інші небезпечні дії. Така глибока аналітика відео значно підвищує ефективність системи безпеки, дозволяючи їй працювати не просто як пасивний інструмент спостереження, а як активний засіб виявлення та протидії потенційним загрозам.

В інтелектуальних системах безпеки розумних будинків застосовується синтез різноманітних технологій і методологій, який піднімає ефективність системи на вищий рівень. Така інтегрована архітектура забезпечує користувачам зручність управління та моніторингу на відстані, можливість отримання оперативних сповіщень про зміни стану безпеки, а також автоматизацію рутинних дій. Окрім того, система може бути індивідуально налаштована з урахуванням специфічних потреб і обставин, в яких знаходиться розумний будинок.

Така комплексна інтеграція технологічних рішень значно знижує ймовірність системних та оперативних помилок, підвищуючи надійність і стійкість системи безпеки. Це, в свою чергу, забезпечує вищий рівень захисту від різних видів загроз, включаючи незаконні вторгнення, пожежні ситуації, ризик затоплення та інші потенційні небезпеки[22].

## 2.2 Визначення фундаментальних функцій та елементів системи

### 2.2.1 Критерії вибору керуючого блоку

Для імплементації інтелектуальної системи безпеки в розумному будинку було обрано промисловий контролер Wiren Board 6. Даний контролер побудований на основі операційної системи Debian Linux версії 7.0 або вище, і характеризується відкритим кодом програмного забезпечення, що надає можливість для кастомізації та масштабування системи відповідно до специфічних потреб користувача. Зовнішній аспект даного контролера ілюстровано на рис. 2.1. Слід зазначити, що вибір контролера з відкритим кодом є стратегічним рішенням, яке забезпечує гнучкість у внесенні змін та адаптації системи до еволюційних змін у технологічному ландшафті. Це також

відкриває дорогу для розробки специфічних модулів та плагінів, що можуть враховувати локальні особливості та потреби.

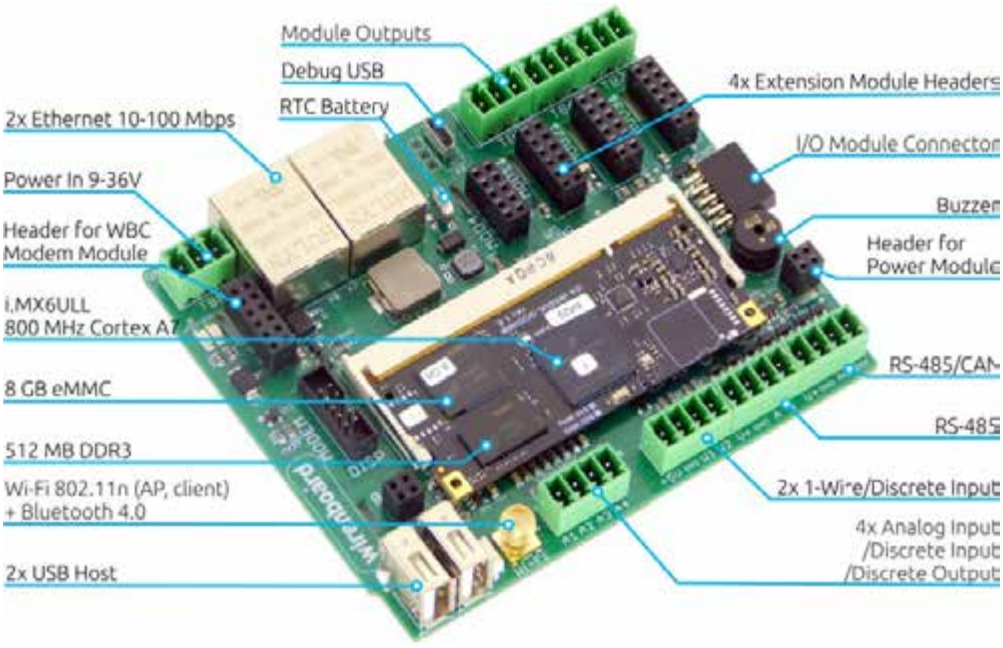


Рис. 2.1 – Зовнішній вигляд Wiren Board 6 без корпусу

Wiren Board 6 використовує протокол черги повідомлень MQTT для здійснення комунікації між різними модулями програмного забезпечення. Апаратні драйвери контролера, відповідальні за інтеракцію з підключеними зовнішніми пристроями, постійно актуалізують стан даних пристроїв, реєструючи цю інформацію у MQTT черзі у вигляді специфічних повідомлень. Веб-інтерфейс контролера, на основі цих даних, відображає актуальний стан роботи пристроїв.

Контролер дозволяє формулювати правила для управління системою за допомогою ECMAScript 5, який є одним з діалектів JavaScript. Ці правила розташовані у спеціальній директорії /etc/wb-rules. ECMAScript в даному контексті використовується з підтримкою спеціальних бібліотек, специфічних для контролера, і є об'єктно-орієнтованою мовою, що надає гнучкість у управлінні компонентами системи та їхнім взаємодією.

Основні переваги контролера Wiren Board 6 можна описати наступним чином:

– гнучкість та адаптивність. Відкритий код програмного забезпечення дозволяє кастомізувати та модифікувати систему відповідно до конкретних потреб користувачів та змінюваних умов експлуатації;

– сумісність. Робота на базі операційної системи Debian Linux забезпечує широку сумісність з різноманітними апаратними та програмними компонентами, що є важливим для інтеграції з іншими системами;

– надійність та стабільність. Debian Linux є однією з найбільш стабільних і безпечних операційних систем, що забезпечує високий рівень захисту та надійності для контролера та всієї системи в цілому;

– масштабованість. Архітектура контролера дозволяє легко додавати нові модулі та компоненти, що спрощує процес масштабування системи в майбутньому;

– інтеграція з іншими системами. Завдяки використанню стандартних протоколів та інтерфейсів, таких як MQTT, контролер може легко інтегруватися з іншими системами автоматизації та управління, включаючи зовнішні сервіси та платформи.

Ці переваги роблять контролер Wiren Board 6 оптимальним вибором для реалізації інтелектуальних систем безпеки в розумних будинках [23].

### 2.2.2 Критерії вибору модульних компонентів

У процесі розробки інтелектуальної системи безпеки для розумного будинку важливим є апаратний компонент, який включає в себе різноманітні датчики та виконавчі механізми, відомі як актуатори. Специфіка датчиків та актуаторів, які слід використовувати, напряму корелює з функціональними можливостями підсистеми, що підлягає імплементації. Для прикладу, у контексті підсистеми моніторингу енергоспоживання використання датчиків напруги та струму, а також розеток із функціональністю вимірювання споживаної енергії, вважається оптимальним. Щодо підсистеми кліматичного контролю, інтеграція термометрів, гігрометрів та датчиків рівня CO2 буде найбільш доцільною. У сценаріях, що вимагають функції виявлення

присутності осіб, такі як "пішов-прийшов", рекомендовано використання датчиків руху та магнітних контактів на вхідних дверях. Осмислений вибір датчиків та актуаторів відповідно до функціональних потреб конкретної підсистеми значуще підвищує надійність та ефективність загальної системи безпеки.

*Датчики*

В контексті підсистеми регулювання мікроклімату важливим елементом є використання багатофункціональних цифрових датчиків. Такі датчики дозволяють одночасно вимірювати параметри температури, вологості, рівня освітленості та інтенсивності шуму. Їх конструкція розрахована на високу точність вимірювань у різних діапазонах. Особливістю цих датчиків є можливість додаткового підключення двох зовнішніх датчиків температури моделі 1-wire DS18B20. Для передачі інформації використовується протокол RS-485 на базі Modbus RTU. Корпус датчиків, виготовлений з високоякісного пластику, дозволяє монтаж на DIN-рейці або за допомогою кріпильних отворів.

Стратегічне розміщення датчиків у приміщенні є ключовим фактором для забезпечення точності вимірювань. Зазвичай температурні датчики рекомендується розташовувати в центральних зонах приміщення, датчики освітленості — по периметру, вологість вимірюється на безпечній відстані від зволожувачів, а датчики шуму — в місцях, де можливе взаємодія з акустичними характеристиками приміщення. У цьому конкретному випадку використовуються датчики виробництва компанії WirenBoard, які не лише забезпечують вимірювання основних параметрів мікроклімату, але й підтримують можливість інтеграції додаткових температурних датчиків.

*Електрохімічний датчик газів WB-MSGR*

Датчик газів WB-MSGR, що працює на електрохімічній основі, представляє собою високочутливий інструмент для моніторингу концентрацій різних газів у атмосферному повітрі. Його робочий принцип ґрунтується на електрохімічному взаємодії, в ході якої взяті на аналіз гази піддаються окисненню або відновленню на чутливому електроді, що викликає коливання в

електричному струмі. Структурно, датчик є комплексом, що включає чутливий електрод, референсний електрод та електроліт, які усі знаходяться в спеціалізованому захисному корпусі.

Цей датчик може адаптуватися для вимірювання різноманітних газів, включаючи вуглекислий газ, аміак, серний діоксид, оксиди азоту та інші, залежно від конфігурації сенсора. Інтерфейс Modbus RTU забезпечує передачу даних на контролюючу панель або до центрального контролера. Такий підхід забезпечує високу точність аналітичних вимірювань та підвищує надійність системи контролю якості повітря в закритих просторах.

Щодо технічних характеристик, WB-MSGR відрізняється широким діапазоном вимірювань, коротким часом відгуку та тривалим середнім терміном служби елемента живлення, що становить понад п'ять років. Датчик може працювати в різних кліматичних умовах, в тому числі при температурах від мінус 20 до плюс 50 градусів Цельсія. Його розміри і маса роблять його легким для інтеграції в різноманітні системи. Відзначимо також, що висока надійність датчика забезпечується за рахунок використання сучасних електрохімічних методів вимірювання концентрацій газів.

*Релейні блоки*

Релейні модулі WB-MR6C від компанії WirenBoard відіграють ключову роль у системах управління розумного будинку, функціонуючи як серцевина для контролю над різноманітними виконавчими механізмами — від автоматичного відкриття і закриття вікон і штор до управління клапанами, замками та іншими пристроями. Ці модулі володіють вбудованим програмним таймером, що дозволяє здійснювати точне короткочасне керування реле, а також регулювати ступінь відкриття чи закриття виконавчих механізмів.

WB-MR6C є шестиканальним релейним модулем, розробленим для універсальної комутації силових навантажень у сферах промислової та домашньої автоматизації. Оснащений шістьма нормально відкритими виходами, кожен з яких може підтримувати струм до 20 ампер, модуль гарантує високу надійність та ефективність. Додаткова функція безпеки полягає у

вбудованому таймері, який автоматично відключає всі реле у випадку відсутності обміну даними через інтерфейс Modbus протягом заданого періоду часу.

Щодо конструкції, контакти реле модуля WB-MR6C згруповані на дві окремі підсистеми, кожна з яких має свій спільний дріт, відомий як COM1 і COM2. Це дозволяє модулю забезпечувати комутацію для навантажень до 10 ампер, комутуючи як змінний, так і постійний струм. Відзначимо також, що входи для зовнішнього управління розташовані на безгвинтових затискачах і працюють за принципом "сухий контакт", що полегшує підключення додаткових кнопок або вимикачів для ручного управління.

Модуль розрахований на роботу з напругами до 250 В в режимі змінного струму або до 30 В при використанні постійного струму. Його реле є нормально відкритими, і в модулі передбачено вхід для загального вимкнення всіх реле, що забезпечує додаткову безпеку. Щодо інтерфейсу, з'єднання з внешніми системами виконується через гвинтові затискачі.

Контакти реле модуля WB-MR6C згруповані на два сегменти, кожен із яких має свій загальний провід, відомий як COM1 та COM2. Додатково, в модулі існує вхід безпечного режиму, що дозволяє автоматично відключати всі реле у випадку відсутності обміну даними через Modbus RTU протягом визначеного періоду.

Модуль підтримує протокол Modbus RTU, що гарантує його сумісність з різними системами управління. Щодо параметрів живлення, модуль може працювати від джерела напруги 10-27 В постійного струму або 24 В змінного струму. Енергоспоживання не перевищує 0,6 Вт. Робочий діапазон температур для модуля складає від -40 до +55 градусів Цельсія.

*Виконавчі механізми*

В розумному будинку виконавчі механізми відіграють ключову роль у керуванні різноманітними пристроями та системами, спрямованими на створення оптимальних умов мікроклімату. Електроприводи, що взаємодіють із вікнами, дверима, шторами та рольшторами, підвищують зручність та

ефективність управління такими елементами. Додатково, засоби автоматичного керування тепловими потоками, наприклад, кульові клапани з електроприводами в системах опалення, сприяють раціональному використанню енергії.

Не менш важливим є використання спеціалізованих побутових приладів, таких як кондиціонери, зволожувачі, обігрівачі, очисники повітря та озонатори, які також підкоряються автоматичному управлінню. Всі ці компоненти разом формують єдину інтегровану систему, що дозволяє створити комфортні умови для проживання, а також забезпечує високу ефективність та зручність управління.

У контексті автоматизації для реалізації цих завдань застосовуються диверсифіковані електроприводи, клапани та інші устрої. Спеціфічні електричні ланцюгові приводи можуть бути використані для автоматичного відкриття та закриття віконних стулок. Що стосується систем опалення, тут активно використовуються кульові клапани з електроприводами. Системи зволоження та очищення повітря можуть включати в себе зволожувачі з ультразвуковою технологією та очисники з активованим вугіллям чи озонатори.

Управління всією системою відбувається через релейні блоки, до яких можуть бути підключені різні елементи керування, включаючи кнопки без фіксації. Для пристроїв, які підтримують інфрачервоне управління, таких як кондиціонери, можуть бути застосовані спеціалізовані модулі інфрачервоних сигналів. Така гнучкість в розміщенні елементів керування робить інтеракцію з системою максимально комфортною та зручною [24-26].

### 2.3 Аналіз методів штучного інтелекту в контексті інтелектуальних систем безпеки

За допомогою технологій машинного навчання та нейронних мереж інтелектуальні системи безпеки здатні ідентифікувати сложені шаблони поведінки та аномальні відхилення, які можуть свідчити про можливі загрози. Це значно підвищує рівень безпеки, дозволяючи системі оперативно реагувати

на потенційно небезпечні ситуації. Така обробка даних на високому рівні комплексності є можливою завдяки алгоритмам глибокого навчання, які дозволяють системі адаптуватися до змінних умов та нових, раніше невідомих загроз.

Окрім того, інтеграція технологій обробки природної мови в систему безпеки робить її більш доступною та зручною у використанні. Зокрема, можливість керування системою за допомогою голосових команд відкриває нові горизонти для інтерактивної взаємодії користувача з системою, роблячи її ефективніше та швидше реагуючою на команди.

Глибоке навчання, як складова частина машинного навчання, надає системі здатність адаптуватися до несподіваних та непередбачених ситуацій. Ця адаптивність робить систему екстремально гнучкою та стійкою до різних викликів, включаючи ті, які не були первісно враховані при її проектуванні. Така комплексна архітектура, що базується на передових методах машинного навчання, створює сильний фундамент для надійної та ефективної системи безпеки.

*Машинне навчання*

Машинне навчання представляє собою специфічну дисципліну в галузі штучного інтелекту, яка фокусується на конструюванні алгоритмічних моделей, здатних "навчатися" на основі емпіричних даних. Цей процес навчання включає аналіз великих наборів даних з метою виявлення закономірностей, які можна використовувати для прогнозування, класифікації або інших типів аналітичних задач. Основна преміса тут полягає в тому, що алгоритм, натренований на достатньо репрезентативному наборі даних, може самостійно формулювати висновки, не потребуючи додаткового програмування.

У контексті навчання з учителем, яке є однією з ключових методологій в машинному навчанні, використовується спеціально підготовлений набір даних. Цей набір містить приклади, кожен з яких включає вхідні параметри та відомі, правильні відповіді на них. Алгоритм аналізує ці пари "вхід-відповідь" для

того, щоб встановити зв'язки між ними. По завершенню цього етапу навчання, алгоритм може бути застосований для прогнозування відповідей на нових, раніше невідомих даних.

Ефективність моделі машинного навчання в значній мірі залежить від якості і об'єму навчальних даних. Після фази тренування модель піддається тестуванню на незалежному наборі даних, що дозволяє оцінити її точність та здатність до узагальнення знань на нові ситуації.

В області систем безпеки для розумних будинків машинне навчання знаходить широке застосування для аналізу поведінки мешканців, виявлення аномальних подій або непередбачуваних шаблонів активності. Моделі можуть, наприклад, бути натреновані на розпізнавання типових патернів активності та виявлення дій, які виходять за межі цих патернів і можуть свідчити про небезпечні ситуації, такі як несанкціонований доступ або інші форми вторгнення. Завдяки цьому, машинне навчання стає потужним інструментом для підвищення ефективності та надійності систем безпеки [27].

*Нейронні мережі*

Нейронні мережі представляють собою унікальний вид моделей машинного навчання, інспірованих структурою та функціонуванням біологічних нервових систем. Ці моделі містять ансамблі архітектурних елементів, званих штучними нейронами, які є організованими в ієрархічні шари. Кожен такий штучний нейрон призначений для прийому, обробки та передачі сигналів, які відповідають вхідним даним або виводам інших нейронів. Особливість цих нейронів полягає в їхній спроможності використовувати змінні ваги для модулювання сили сигналів, а також застосовувати активаційні функції для формування вихідного сигналу.

В контексті зворотньо зв'язаних нейронних мереж, процес навчання відбувається через алгоритм зворотнього поширення помилки. Вхідні дані ініціалізуються на вхідному шарі нейронів, пройдуться через один або декілька прихованих шарів, і в результаті формують вихідні сигнали на основі динамічно налаштованих ваг і активаційних функцій. Після цього вихідні

сигнали порівнюються з очікуваними результатами, і ваги нейронної мережі коригуються з метою мінімізації розбіжностей між фактичними та очікуваними вихідними даними. Цей процес повторюється, доки нейронна мережа не досягне заданого рівня точності.

В сфері систем безпеки розумних будинків нейронні мережі знаходять широкий спектр застосувань. Вони можуть служити для аналітичної обробки відеоданих, розпізнавання особистостей на основі їхніх облич, аналізу акустичних патернів для виявлення незвичайних звуків, які можуть вказувати на небезпечні ситуації, та для ряду інших завдань, що стосуються безпеки. Динамічність та гнучкість нейронних мереж дозволяють їм виявляти складні залежності та непередбачувані патерни в даних, що робить їх вкрай ефективними для задач забезпечення безпеки.

Навчання з підкріпленням є одним із підходів в галузі машинного навчання, де модель, зазвичай звана "агентом", взаємодіє з навколишнім середовищем, отримуючи позитивні або негативні винагороди з метою оптимізації певної функції винагороди. Суть даного підходу полягає в тому, що агент намагається виявити тактику дій, яка максимізує суму отриманих винагород від середовища через довгий період часу.

Для прикладу, агент у розумному будинку контролює систему відеоспостереження, датчики руху, магнітні датчики на дверях/вікнах, а також аудіо-сенсори. Середовищем є сам будинок і всі події, які в ньому відбуваються. Винагорода — це відсутність вторгнень або швидке їх виявлення. Штрафи накладаються за помилкові тривоги або невиявлені вторгнення.

Алгоритм роботи агента виглядає наступним чином:

– ініціалізація. Агент запускається з базовою стратегією, яка може бути заснована на простих правилах (наприклад, спрацьовування тривоги при відкритті дверей вночі);

– взаємодія з середовищем. Агент отримує дані з датчиків у реальному часі і вирішує, як реагувати — включити тривогу, відправити повідомлення власникові, чи ігнорувати подію;

– отримання винагороди та штрафів. Якщо агент правильно ідентифікує вторгнення, він отримує позитивну винагороду. В разі помилкової тривоги або пропущеного вторгнення — штраф;

– оновлення стратегії. Засновуючись на отриманих винагородах і штрафах, агент корегує свою стратегію за допомогою алгоритмів навчання з підкріпленням, таких як Q-learning;

– адаптація. Стратегія агента може адаптуватися до нових форм вторгнень або до зміни поведінки мешканців (наприклад, якщо вони повертаються додому пізніше, ніж зазвичай).

Такий підхід дозволяє створити гнучку систему безпеки, яка може адаптуватися до нових загроз і мінімізувати кількість помилкових тривог. На відміну від традиційних систем, які часто базуються на жорстких правилах, дана система здатна до самовдосконалення через неперервну взаємодію з середовищем.

Застосування машинного навчання, включаючи нейронні мережі та алгоритми навчання з підкріпленням, стає критичним фактором у підвищенні надійності та ефективності систем безпеки в розумних будинках. Це надає системі здатність до автономної адаптації згідно з динамічно змінюючимися умовами та сценаріями загроз. Центральну роль у цьому процесі відіграє навчання з підкріпленням, яке, керуючись системою винагород та штрафів, оптимізує стратегії реагування на різні події, отримуючи з того максимальну користь у вигляді збільшення безпеки.

Цей механізм навчання забезпечує системі гнучкість, необхідну для ефективного виявлення та нейтралізації потенційних загроз, в тому числі тих, які можуть бути нестандартними або непередбачуваними. Особливо це стає актуальним у контексті постійної еволюції засобів вторгнення та злому безпеки.

На допомогу цьому приходять нейронні мережі, які володіють здатністю до глибокого аналізу даних та виявлення складних закономірностей. Це може бути особливо корисно при аналізі відеоданих, аудіозаписів або даних з датчиків руху для виявлення незвичайних або аномальних ситуацій. Нейронні мережі можуть вивчати поведінку мешканців будинку, їх звичайні шляхи пересування та часові закономірності, щоб ефективніше виявляти дії, які виходять за рамки звичайного.

Отже, синтез алгоритмів навчання з підкріпленням та нейронних мереж в системах безпеки розумного будинку не просто підвищує ефективність цих систем, але і робить їх спроможними адаптуватися до нових видів загроз, забезпечуючи тим самим високий рівень захисту для мешканців [28].

### 2.4 Механізм роботи алгоритму ідентифікації потенційних загроз

Процес конструювання алгоритмічних складників для інтелектуальних систем безпеки розумного дому можна розглядати як комплексний багатоетапний дослідницький проєкт. Початковий етап цього процесу полягає у систематичному зборі та ретельній обробці даних, отриманих від множини сенсорів — рухомірів, акустичних датчиків, детекторів диму та інших. Ці дані підлягають прекондиціонуванню для подальшого аналізу та використання у моделях машинного навчання.

Вибір конкретних моделей машинного навчання для інтелектуальних систем безпеки розумного дому є не тривіальною задачею, що вимагає ретельного аналізу типу проблеми, сценаріїв використання, а також доступних даних. Специфічний характер задачі, такий як виявлення можливих загроз, часто вимагає застосування алгоритмів бінарної класифікації, які можуть ділити ситуації на "нормальні" та "аномальні" або "безпечні" та "небезпечні".

Логістична регресія є одним з наибільш базових, але при цьому ефективних методів для задач бінарної класифікації. Ця модель добре підходить для випадків, коли можливе лінійне відділення класів в просторі ознак. Однак, логістична регресія може бути недостатньою для виявлення

складних шаблонів в даних, особливо якщо простір ознак має високу розмірність чи взаємодію між різними ознаками.

В таких сценаріях можуть прийти на допомогу більш складні методи, такі як ансамблеві моделі. Випадковий ліс, як один з представників ансамблевих методів, забезпечує високу точність та здатність до виявлення неочевидних шаблонів завдяки комбінації прогнозів від множини дерев рішень [29].

Машини опорних векторів (SVM) також є цінним вибором для задач класифікації, особливо коли необхідно врахувати нелинійні взаємозв'язки між ознаками. Ці моделі здатні конструювати гіперплощини в багатовимірних просторах ознак для ефективного відділення класів.

Вибір конкретної моделі машинного навчання відбувається на основі комплексного аналізу, що включає в себе не лише тип задачі, але й природу даних, складність шаблонів, які необхідно виявити, а також обмеження на обчислювальні ресурси та часові рамки для навчання та валідації моделі [30].

Наступний етап — це власне навчання моделі. Цей етап охоплює застосування методів оптимізації для вибору оптимальних параметрів моделі, що максимізують її прогностичну здатність. Валідація моделі, яка зазвичай здійснюється за допомогою крос-валідації або на основі окремого тестового набору даних, дозволяє оцінити її ефективність за допомогою таких метрик як точність, відгук, а також площа під ROC-кривою.

Одержані моделі потім інтегруються в систему безпеки для оперативного розпізнавання загроз на основі стрімингових даних від датчиків. Ці моделі не є статичними, а підлягають періодичному оновленню і оптимізації з урахуванням нових даних та змінних умов експлуатації.

Також, варто врахувати можливість помилкових прогнозів, тому розробка алгоритмів для обробки неоднозначних або конфліктних ситуацій є обов'язковим етапом. Тут може бути використано такі методи, як консенсус прогнозів між декількома моделями, введення порогових значень для активації певних реакцій системи, або врахування контекстуальних факторів, таких як добовий ритм активності мешканців або їхнє розташування в приміщенні .

На рис. 2.2 представлена блок-схема, яка ілюструє робочий процес алгоритмічного рішення, базованого на моделі машинного навчання, призначеного для ідентифікації потенційних загроз.

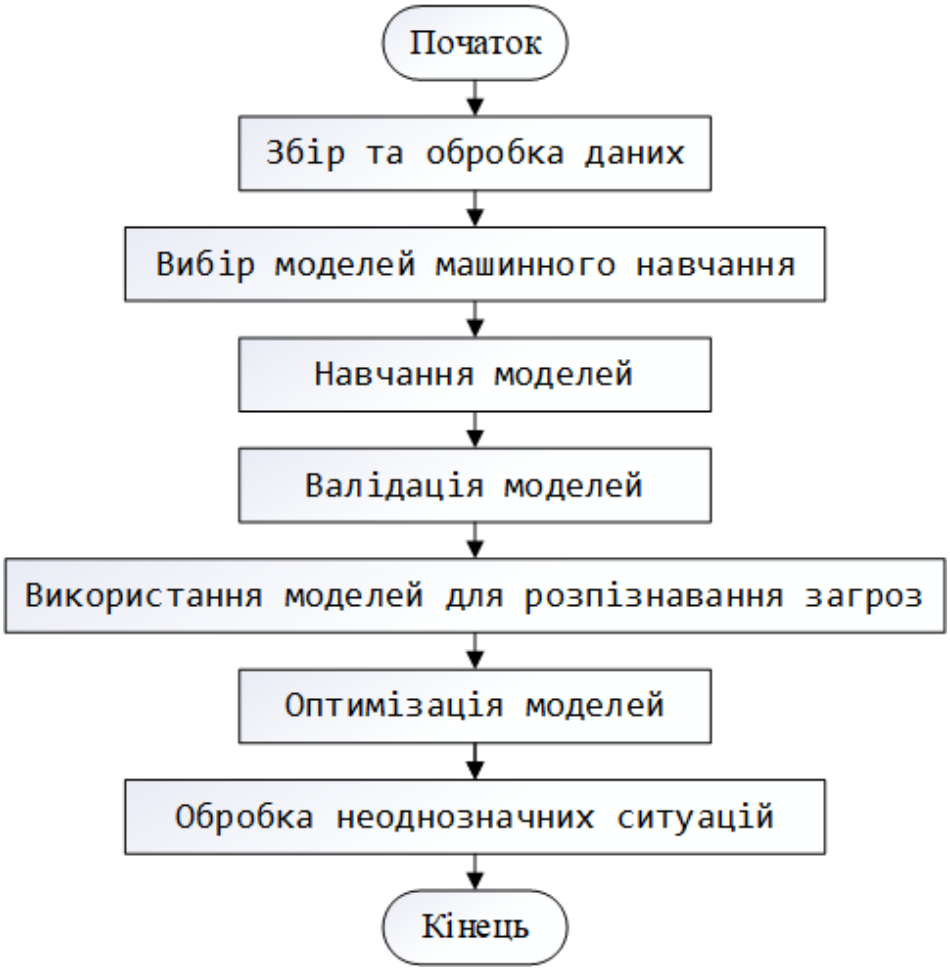


Рис. 2.2 – Процес роботи алгоритму для розпізнавання загрози

## 2.5 Обґрунтування вибору бібліотеки ML .NET для машинного навчання

ML.NET є відкритою бібліотекою для машинного навчання, яка розроблена компанією Microsoft специфічно для платформи .NET. Її основна суть полягає в наданні інструментарію для створення, тренування та розгортання моделей машинного навчання в програмах і сервісах, написаних на .NET. Завдяки цій бібліотеці, розробники, які працюють в екосистемі .NET, здатні інтегрувати алгоритми машинного навчання безпосередньо в свої застосунки без необхідності використовувати сторонні мови програмування або бібліотеки [31].

Бібліотека надає засоби для виконання ряду типових завдань машинного навчання: класифікація, регресія, кластеризація, рекомендаційні системи, та інші. Вона включає в себе широкий спектр алгоритмів для цих завдань, які оптимізовані для високої продуктивності та масштабованості.

Особливо варто відзначити, що ML.NET підтримує трансферне навчання, дозволяючи використовувати попередньо натреновані моделі, зокрема з таких популярних бібліотек машинного навчання як TensorFlow. Це дає можливість значно прискорити процес розробки і впровадження інтелектуальних систем.

Використання ML.NET може бути особливо цінним для корпоративних додатків та служб безпеки, які вже базуються на технологіях Microsoft і для яких важлива інтеграція з іншими сервісами і платформами компанії.

Вибір бібліотеки ML.NET для розробки інтелектуальної системи на основі машинного навчання обумовлений рядом ключових факторів, що є важливими для ефективної і надійної реалізації проекту:

– висока продуктивність. ML.NET є високопродуктивною бібліотекою, що розроблена з урахуванням потреб промислових застосувань. Це забезпечує швидке тренування моделей і низькі латентні показники при їх використанні в реальному часі;

– інтеграція з екосистемою .NET: ML.NET надає безперебійну інтеграцію з уже існуючими платформами і додатками на .NET, що суттєво спрощує впровадження машинного навчання в існуючі системи без необхідності застосування сторонніх бібліотек або мов програмування;

– широкий спектр алгоритмів. ML.NET надає широкий вибір алгоритмів для класифікації, регресії, кластеризації та інших задач машинного навчання, що забезпечує гнучкість у виборі оптимального методу для конкретного завдання;

– підтримка трансферного навчання. Можливість використовувати попередньо натреновані моделі для подальшого вдосконалення на специфічних для проекту даних дозволяє значно скоротити час розробки та впровадження системи;

– сумісність і розширюваність. ML.NET легко інтегрується з іншими платформами і бібліотеками для наукових досліджень і аналізу даних, такими як TensorFlow або ONNX, що забезпечує високий рівень розширюваності та сумісность;

– документація та спільнота. Наявність обширної документації, навчальних матеріалів та активної розробницької спільноти сприяє ефективному вивченню бібліотеки і швидкому вирішенню виникаючих проблем.

Таким чином, вибір ML.NET як основи для розробки інтелектуальної системи безпеки є обґрунтованим з позицій технічних можливостей, продуктивності, сумісності та підтримки з боку розробників і спільноти.

### 2.6 Імплементація ключових модулів системи

У архітектурі інтелектуальної системи безпеки розумного будинку розроблено кілька функціональних модулів. Ці модулі спеціалізуються на конкретних аспектах безпеки, включаючи, але не обмежуючись, виявленням несанкціонованого доступу, виявленням сигналів про пожежу, а також іншими механізмами, що сприяють моніторингу та контролю стану розумного будинку.

У контексті машинного навчання для підвищення ефективності системи безпеки було створено два ключові класи, зокрема SensorInput та SensorOutput, що представлені на рис. 2.3 та 2.4 відповідно. Ці класи служать для формалізації структури вхідних та вихідних даних, які будуть використовуватися моделлю машинного навчання для аналізу та виявлення потенційних загроз. Клас SensorInput представляє собою абстракцію вхідних даних від датчиків безпеки, тоді як клас SensorOutput відображає очікувані або фактичні реакції системи на різні типи входів. Ці класи становлять основу для тренування та валідації моделі машинного навчання, спрямованої на автоматичне виявлення та реагування на загрозові ситуації в розумному будинку.

```
public class SensorInput {
    [LoadColumn(0), ColumnName("WindowMotion")]
    public float WindowMotion;

    [LoadColumn(1), ColumnName("DoorMotion")]
    public float DoorMotion;

    [LoadColumn(2), ColumnName("NoiseLevel")]
    public float NoiseLevel;

    [LoadColumn(3), ColumnName("SmokeLevel")]
    public float SmokeLevel;

    [LoadColumn(4), ColumnName("Threat")]
    public bool Threat;
}
```

Рис. 2.3 – Код класу «SensorInput»

```
public class SensorOutput {
    [ColumnName("PredictedLabel")]
    public bool Prediction;

    [ColumnName("Probability")]
    public float Probability;

    [LoadColumn(4), ColumnName("Threat")]
    public bool Threat;
}
```

Рис. 2.4 – Код класу «SensorOutput»

Клас SensorInput конструє структуральний макет для вхідних даних, які подаються на обробку моделі машинного навчання. Цей клас включає в себе декілька полів, що являють собою квантитативні та категоріальні мірки безпеки в контексті розумного будинку. Деталізовані характеристики цих полів подані нижче:

– WindowMotion: це метричний показник, що квантифікує рух, зареєстрований датчиками руху, розташованими на віконних конструкціях;

– DoorMotion: аналогічно попередньому, це метрика, що реєструє рухову активність у зоні дверних проїмів на основі відповідних датчиків;

– NoiseLevel: це параметр, що відзеркалює інтенсивність акустичного поля в приміщенні, зареєстровану сенсорами звукового діапазону;

– SmokeLevel: метрика, що вказує на концентрацію димових часток в атмосфері приміщення, виміряну за допомогою спеціалізованих датчиків диму;

– Threat: це булева змінна, яка відображає наявність або відсутність загрози в системі безпеки. Ця змінна є критичною для генерації тренувального датасету та слугує міткою для навчання моделі машинного навчання.

Кожне з цих полів відіграє важливу роль у формуванні тренувальних та тестових датасетів, що використовуються для побудови та валідації моделі машинного навчання в системі безпеки розумного будинку.

Клас SensorOutput конститує структурну основу для вихідних даних, генерованих моделлю машинного навчання в контексті системи безпеки розумного будинку. Ця конструкція включає поля, що кумулюють прогнозовану мітку та асоційовану з нею ймовірність. Далі представлено деталізацію цих компонентів:

– Prediction: це булевий індикатор, що формує прогнозовану категоріальну мітку в системі безпеки. Цей параметр слугує для визначення, чи існує в даному контексті загроза, чи її немає;

– Probability: представляє собою кількісну оцінку, що показує ступінь впевненості моделі у прогнозованій мітці. Це числове значення відображає ймовірність того, що конкретна ситуація може бути класифікована як загрозлива.

Threat: це поле, яке корелює з аналогічним полем у класі SensorInput і служить для надання інформації про фактичний стан загрози на основі даних, які були первинно зібрані та проаналізовані.

Кожне з цих полів має важливу роль у контексті аналітичного розв'язання задач безпеки в розумних будинках, дозволяючи не лише класифікувати потенційні ризики, але і оцінювати ступінь впевненості моделі в її діагностичних висновках.

У процесі реалізації моделі машинного навчання за допомогою бібліотеки ML.NET було створено три ключові змінні: context, model, та data (рис. 2.5). Ці змінні функціонують як сутнісні об'єкти, що координують роботу модулів та алгоритмів в рамках даної бібліотеки.

```
private MLContext context;
TransformerChain<BinaryPredictionTransformer<CalibratedModelParametersBase
  <FastTreeBinaryModelParameters, PlattCalibrator>>> model;
private IDataView data;
```

Рис. 2.5 – Опис ключових змінних

Змінна context, типу MLContext, актує як головний управляючий об'єкт, який координує роботу різноманітних компонентів бібліотеки ML.NET. Це охоплює ініціалізацію алгоритмів машинного навчання, препроцесингових утиліт, валідаційних механізмів та додаткових компонентів. Слід підкреслити, що MLContext є концентратором, який забезпечує зв'язок між всіма необхідними компонентами для реалізації моделі машинного навчання.

model представляє фінішовий продукт процесу навчання, а саме навчену модель. Ця модель концептуалізується як послідовність трансформерів, що складається з двох основних компонентів. Перший компонент, BinaryPredictionTransformer, спеціалізується на бінарній класифікації. Другий компонент, CalibratedModelParametersBase, задіяний у калібруванні моделі для оптимізації точності прогнозування. В даному випадку, специфічна реалізація цих компонентів базується на моделі FastTreeBinaryModelParameters та калібраторі PlattCalibrator.

Змінна data використовується для зберігання даних, які подаватимуться на вхід моделі для навчання або валідації. Ця змінна є ключовою для управління даними в рамках ML.NET і дозволяє забезпечити ефективну обробку та трансформацію датасетів.

Поступово була інтегрована подійна модель "OpenBtn_Click", задачею якої є ініціація процесу відкриття файлу з тренувальними даними та подальше навчання нейронної мережі. В даній функціональності застосовується компонент інтерфейсу OpenFileDialog, що слугує інструментом для вибору відповідного файлу даних. Після здійснення успішного вибору файлу, тренувальні дані зчитуються та трансформуються у колекцію об'єктів типу SensorInput, яка отримала найменування datas (рис. 2.6).

```
List<SensorInput> datas = new List<SensorInput>();
using (StreamReader reader = new StreamReader(_Path, Encoding.GetEncoding(1251))) {
  // Пропускаємо перший рядок з іменами стовпців
  reader.ReadLine();

  while (!reader.EndOfStream) {
    var line = reader.ReadLine();
    var values = line.Split(';');
    SensorInput item = new SensorInput() {
      WindowMotion = float.Parse(values[0]),
      DoorMotion = float.Parse(values[1]),
      NoiseLevel = float.Parse(values[2]),
      SmokeLevel = float.Parse(values[3]),
      Threat = bool.Parse(values[4])
    };
    datas.Add(item);
  }
}
```

Рис. 2.6 – Зчитування даних із файлу

В рамках використання бібліотеки ML.NET, імперативно провести етап підготовки даних, що включає їх розділення на відповідні підмножини (див. рис. 2.7).

```
data = context.Data.LoadFromEnumerable<SensorInput>(datas);
var trainTestData = context.Data.TrainTestSplit(data, testFraction: 0.2, seed: 0);
var trainData = trainTestData.TrainSet;
var testData = trainTestData.TestSet;
```

Рис. 2.7 – Підготовка та розподіл даних

У вихідному коді імплементовано механізм завантаження даних з колекції та їх подальшого розподілу на тренувальну та тестову підмножини. Зберігання цих підмножин здійснюється у змінних з назвами trainData та testData, що згодом використовуватимуться для процесів навчання та валідації моделі машинного навчання. Після завершення цього етапу, необхідно ініціювати конвеєр даних, який призначений для попередньої обробки даних перед входом їх в модель машинного навчання в рамках бібліотеки ML.NET (рис. 2.8).

```
var pipeline = context.Transforms.Concatenate(
  "Features", nameof(SensorInput.WindowMotion), nameof(SensorInput.DoorMotion),
  nameof(SensorInput.NoiseLevel), nameof(SensorInput.SmokeLevel))
  .Append(context.Transforms.NormalizeMinMax("Features"))
  .Append(context.Transforms.CopyColumns(inputColumnName: "Threat", outputColumnName: "Label"))
  .Append(context.BinaryClassification.Trainers.FastTree());
```

Рис. 2.8 – Створення конвеєра обробки даних

Вказаний у вихідному коді конвеєр даних спрямований на підготовку інформації для подальшого навчання моделі в рамках бібліотеки ML.NET. Цей конвеєр включає в себе кілька ключових операцій:

– агрегація вхідних атрибутів WindowMotion, DoorMotion, NoiseLevel та SmokeLevel в єдиний атрибут, який отримує назву "Features";

– нормування агрегованого атрибуту "Features" з метою перетворення його значень до діапазону [0, 1];

– клонування колонки "Threat" в новостворену колонку, яка отримує назву "Label" і представляє цільову змінну або мітку для моделі;

– застосування алгоритму FastTree для реалізації процесу навчання моделі бінарної класифікації.

Цей конвеєр дозволяє впоратися з вхідними даними, обробити їх та підготувати для тренування моделі машинного навчання. Створивши конвейєр, необхідно провести навчання нейронної мережі та зберегти модель (рис 2.9).

```
// Навчання моделі
model = pipeline.Fit(trainData);
```

Рис. 2.9 – Навчання моделі НМ

Модель, яка пройшла етап навчання, консервується в змінній з назвою "model" з метою її подальшої аплікації в системі. Наступним обов'язковим кроком є верифікація якості навченої моделі (рис. 2.10).

```
var predictions = model.Transform(testData);
var metrics = context.BinaryClassification.Evaluate(predictions);
RaportTBox.Text += ($"Accuracy: {metrics.Accuracy:P2}") + "\r\n";
RaportTBox.Text += ($"F1Score: {metrics.F1Score:P2}") + "\r\n";
```

Рис. 2.10 – Перевірка навченої моделі нейронної мережі

Цей блок програмного коду здійснює оцінювання результатів роботи навченої моделі на основі тестового набору даних. В якості метрик для оцінки ефективності використовуються точність (Accuracy) та F1-міра (F1 Score).

Функціональність збереження моделі в базі даних реалізована через подію "AddBtn_Click", яка активізується при натисканні кнопки "Додати" (рис. 2.11).

```
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);
    _NeuralProvider.InsertNeural(NeuralNamesTBox.Text,
      Convert.ToInt32(ScenariosCBox.SelectedValue), pathName);
    context.Model.Save(model, data.Schema, localProj + pathName);
    //context.Model.Save(model, data.Schema, "model.zip");
    ClearAllData();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
      "Було навчено нейронну мережу " +
      NeuralNamesTBox.Text, DateTime.Now);
    MessageBox.Show("Дані успішно збережено!");
  }
}
```

Рис. 2.11 – Код події «AddBtn_Click»

Цей сегмент програмного коду здійснює ряд послідовних операцій:

– валідація вхідних даних: Початково здійснюється аналіз коректності вхідних даних за допомогою методу IsDataEnteringCorrect(). У випадку невідповідності вхідних даних встановленим критеріям, подальше виконання коду припиняється;

– формування місця зберігання моделі: При позитивному результаті валідації програма генерує шлях для збереження файлу моделі. Це робиться шляхом комбінації базової директорії ("\teach"), випадково згенерованої назви файлу (за допомогою методу GenerateFileName()) та розширення файлу (".zip");

– реєстрація моделі в базі даних: Далі відбувається вставка інформації про навчену нейронну мережу в базу даних через метод _NeuralProvider.InsertNeural. В параметрах передаються назва нейронної мережі, обраний сценарій використання та локація файлу моделі;

– фізичне зберігання моделі: Оперативне зберігання навченої моделі в файл здійснюється за допомогою методу context.Model.Save();

– очищення полів введених даних. Викликається метод ClearAllData(), який здійснює очищення полів вводу в графічному інтерфейсі програми;

– логування подій. Виконується додавання запису в журнал системи про успішне завершення процесу навчання нейронної мережі через метод _LogsProvider.InsertLogs;

– інформування користувача. Згодом виводиться повідомлення, що сповіщає користувача про успішне збереження даних.

Для здійснення моніторингу загроз в оперативному режимі була створена спеціалізована візуальна форма, яка надає можливість імпортувати вже натреновану модель аналізу даних. Функціонал завантаження моделі реалізований через обробник події «ScenariosCBox_SelectedValueChanged», активація якого відбувається автоматично після вибору конкретної моделі із випадкового списку на графічному інтерфейсі. Деталізований лістинг даного обробника подій можна знайти на рис. 2.12.

```
private void ScenariosCBox_SelectedValueChanged(object sender, EventArgs e) {
  if (_IsThemesLoad) {
    _SelectedNeural = _NeuralProvider.SelectedNeuralByScenariosId
      (Convert.ToInt32(ScenariosCBox.SelectedValue));
    LoadData(_SelectedNeural.NeuralFileModel);
    _LocalProj = Application.StartupPath +
      _SelectedNeural.NeuralFileModel;
  }
}
```

Рис. 2.12 – Код обробник події «ScenariosCBox_SelectedValueChanged»

Процес виконання кодової бази характеризується наступними послідовними фазами:

– верифікація стану ініціалізації. Перед ініціацією будь-яких подальших операцій проводиться перевірка статусу завантаження сценаріїв загроз за допомогою змінної _IsThemesLoad. У випадку, коли сценарії не є завантаженими, подальші дії алгоритму не ініціюються;

– селекція моделі нейронної мережі. У випадку позитивного результату верифікації стану ініціалізації, викликається метод _NeuralProvider.SelectedNeuralByScenariosId(). Даний метод реалізує вибір конкретної нейронної мережі на основі ідентифікатора сценарію, який був вибраний користувачем через графічний інтерфейс (ScenariosCBox).

Ідентифікатор сценарію передається в метод як вхідний параметр. Результат методу зберігається у змінній _SelectedNeural;

– ініціалізація датасету. Завершальна фаза передбачає виклик методу LoadData(), який реалізує завантаження даних відповідно до файлу моделі, асоційованого з вибраною нейронною мережею.

Для симуляції вхідних даних від сенсорів системи безпеки була сконструйована функція під назвою "GenerateRandomInput", алгоритмічна реалізація якої ілюстрована на рис. 2.13.

```
1 reference
public SensorInput GenerateRandomInput() {
  return new SensorInput {
    WindowMotion = (float)random.NextDouble(),
    DoorMotion = (float)random.NextDouble(),
    NoiseLevel = (float)random.NextDouble(),
    SmokeLevel = (float)random.NextDouble(),
    Threat = random.Next(2) == 0
  };
}
```

Рис. 2.13 – Код генерації даних датчиків

Для моделювання активності різних типів датчиків, що входять до складу системи безпеки розумного будинку, випадковим методом генеруються числові значення. Ці значення пізніше використовуються як вхідні дані для аналізу. Крім того, для кожного згенерованого набору даних формується булевий параметр, який відображає наявність або відсутність імовірної загрози.

З метою надання можливості візуалізації цих синтетичних даних, їх аналізу в контексті потенційних загроз, а також для перевірки ефективності та адекватності раніше навченої моделі машинного навчання, була реалізована спеціалізована обробка події таймера. Ця обробка події, яка отримала назву «timer1_Tick», забезпечує періодичну активацію вказаних процесів аналізу та відображення. Детальніше цей механізм представлено на рис. 2.14.

```
private void timer1_Tick(object sender, EventArgs e) {
  try {
    var generator = new SensorInputGenerator();
    var predictor = new SensorPredictor(_LocalProj);
    var randomInput = generator.GenerateRandomInput();
    var isThreat = predictor.Predict(randomInput);
    RaportTBox.Text += $"WindowMotion: {randomInput.WindowMotion}," +
      $" DoorMotion: {randomInput.DoorMotion}, " +
  $"NoiseLevel: {randomInput.NoiseLevel}, SmokeLevel: {randomInput.SmokeLevel}," +
  $" Threat: {randomInput.Threat}" + "\r\n";
    if (isThreat == true) {
      RaportTBox.Text += "Спрацювала тривога\r\n";
    }
    RaportTBox.SelectionStart = RaportTBox.Text.Length;
    RaportTBox.ScrollToCaret();
  } catch (Exception ex) {
    RaportTBox.Text = ex.ToString();
  }
}
```

Рис. 2.14 – Код методу «timer1_Tick»

Обробник події таймера ініціюється з періодичністю 300 мс після активації кнопки "Запустити". Він відповідає за генерацію синтетичних даних, які емулюють реальні сигнали від датчиків системи безпеки. Ці даних подаються на вхід предварительно навченої моделі машинного навчання для оцінки потенційних загроз.

Якщо модель ідентифікує можливу загрозу, це відображається у динамічному системному звіті, доступному через графічний інтерфейс користувача. Отже, цей підрозділ зосереджується на аналізі та описі важливих аспектів коду, що відносяться до моделювання та аналізу ризиків.

### 2.7 Детальна специфікація розроблених алгоритмів

В цьому підрозділі здійснено деталізований аналіз алгоритмічної структури, критичної для функціональності інтелектуальної безпекової системи в контексті розумного будинку. Рис. 2.15 відображає блок-схему, на якій ілюстровано послідовність операцій щодо інтеграції даних про навчену модель на основі тестового набору даних у системну архітектуру.

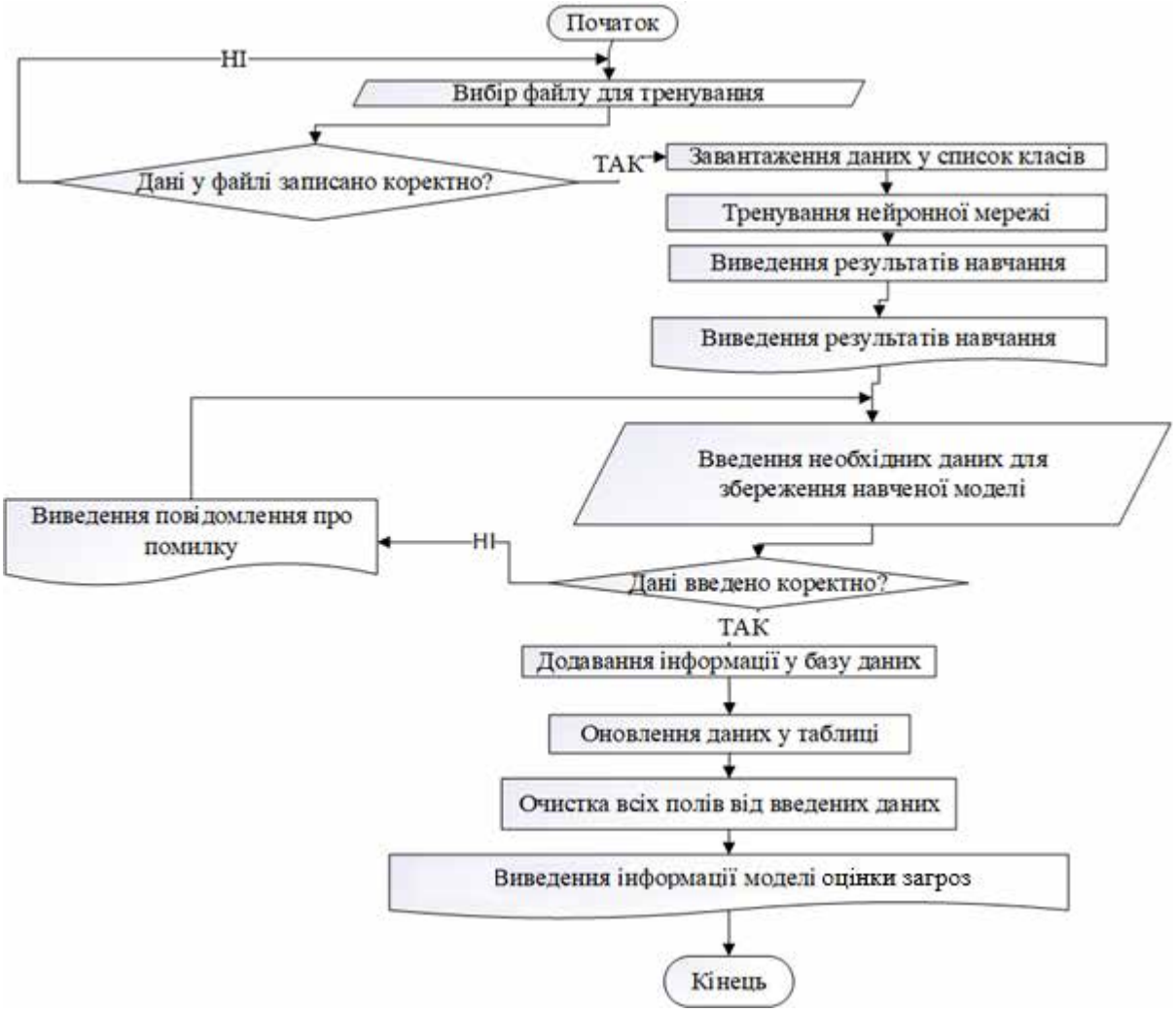


Рис. 2.15 – Додавання інформації про модель нейронної мережі

В ході виконання алгоритму, що ілюстровано розробленою блок-схемою, спершу здійснюється процедура вибору відповідного файлу, який містить тренувальний набір даних. Цей етап критичний для подальшої коректності моделі. Після цього ініціюється процес верифікації цих даних на предмет їхньої валідності та консистентності. У разі виявлення некоректних даних, система повертає користувача до етапу вибору файлу, щоб уникнути подальших помилок у моделі.

По завершенню верифікації, дані з вибраного файлу імпортуються в систему та алокуються у відповідному списку класів для подальшого використання. Це послідовно переходить до фази тренування нейронної мережі, де модель оптимізується на основі завантаженого тренувального набору даних.

Після завершення процедури тренування, результати аналізуються та виводяться для оцінки ефективності та точності моделі. У цей момент користувачу надається можливість ввести метадані для моделі, такі як її ім'я, для її подальшого збереження.

Ще одна фаза верифікації відбувається при введенні цих метаданих. Якщо користувач вводить некоректну інформацію, система генерує повідомлення про помилку і користувач має можливість коректувати введені дані. По завершенню цього етапу, інформація про нову навчену модель інтегрується в базу даних.

Далі, система автоматично оновлює таблицю, яка відображає актуальний стан збережених моделей. Всі поля для вводу даних очищаються для можливості ініціації нового циклу тренування.

На завершальному етапі, користувачу предоставляється агрегована інформація про всі моделі, які були раніше створені та збережені в системі. Таким чином, алгоритм забезпечує цілісний процес від вибору тренувального набору даних до інтеграції навченої моделі в системну архітектуру.

На рис. 2.16 представлена блок-схема, яка деталізовано ілюструє механізми та послідовності дій, що реалізуються для забезпечення функціональності навченої моделі в режимі реального часу. Схема включає в себе критичні компоненти та алгоритмічні рішення, які спільно забезпечують високу ефективність та надійність системи.

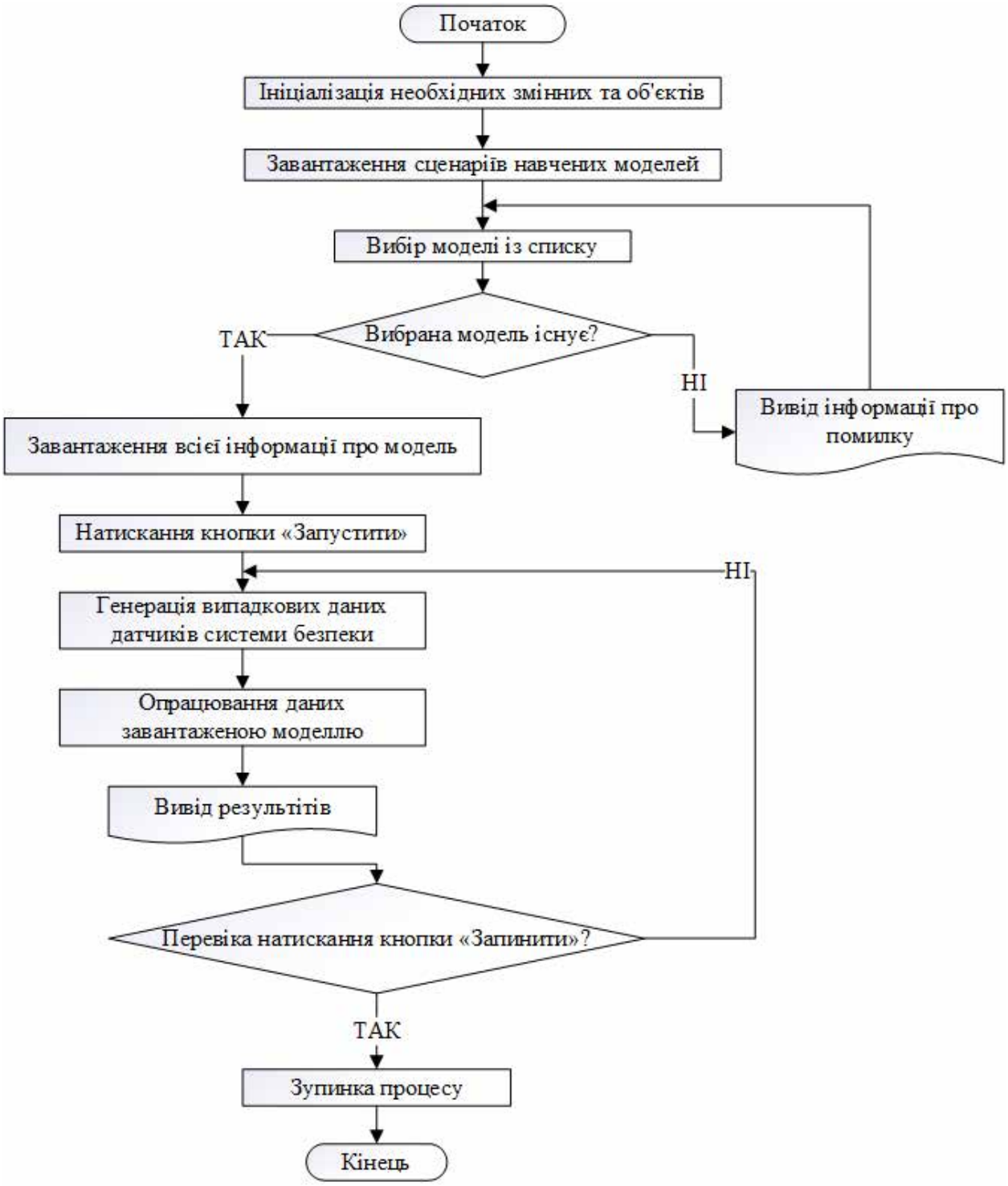


Рис. 2.16 – Процес роботи навченої моделі нейронної мережі

У процесі роботи з навченою моделлю для опрацювання даних в реальному часі ініціюється ряд ключових змінних та об'єктів, які задають параметри робочого середовища. Після цього виконується загрузка сценаріїв, асоційованих з раніше навченими моделями, для подальшого їх використання.

Користувачу надається можливість вибору конкретної моделі зі списку доступних. При цьому здійснюється перевірка наявності вибраної моделі у

системі. У випадку відсутності моделі користувач інформується про це через з'явлення відповідного повідомлення, і процес відкочується до етапу вибору моделі.

Якщо вибрана модель існує, здійснюється загрузка всієї відповідної інформації про неї. Після цього користувач ініціює початок опрацювання даних шляхом натискання кнопки "Запустити".

На наступному етапі система автоматично генерує випадкові дані, що імітують сигнали датчиків системи безпеки. Ці дані подаються на вхід завантаженій моделі для їх подальшого аналізу та опрацювання.

Результати аналізу виводяться користувачу для оцінки. Протягом всього циклу опрацювання даних система періодично перевіряє, чи була натиснута кнопка "Запинити" для призупинення процесу. У випадку її активації цикл опрацювання даних припиняється.

Таким чином, весь процес є взаємозв'язаним і динамічним, із можливістю адаптації до потреб користувача та змінних умов операційного середовища.

### 2.8 Конфігурація та налаштування системи

Для конфігурації пристроїв через веб-інтерфейс контролера Wiren Board 6 ініціюється процес, який розпочинається з авторизації у веб-інтерфейсі. Це здійснюється шляхом введення IP-адреси контролера у адресному рядку веб-браузера. Після успішного доступу до веб-інтерфейсу наступним етапом є зміна рівня доступу користувача до статусу "Administrator". Це необхідно для отримання повного контролю над налаштуваннями пристроїв та системи в цілому, як це відображено на рис. 2.17.

Після внесення всіх необхідних змін та адаптацій в налаштуваннях контролера, важливо пам'ятати про зміну рівня доступу до веб-інтерфейсу на більш обмежений, наприклад "User" або "Operator". Це робиться з метою запобігання несанкціонованим змінам або випадковим помилкам, які можуть виникнути в процесі подальшої експлуатації системи. Така послідовність дій забезпечує високий рівень безпеки та стабільність роботи контролера.

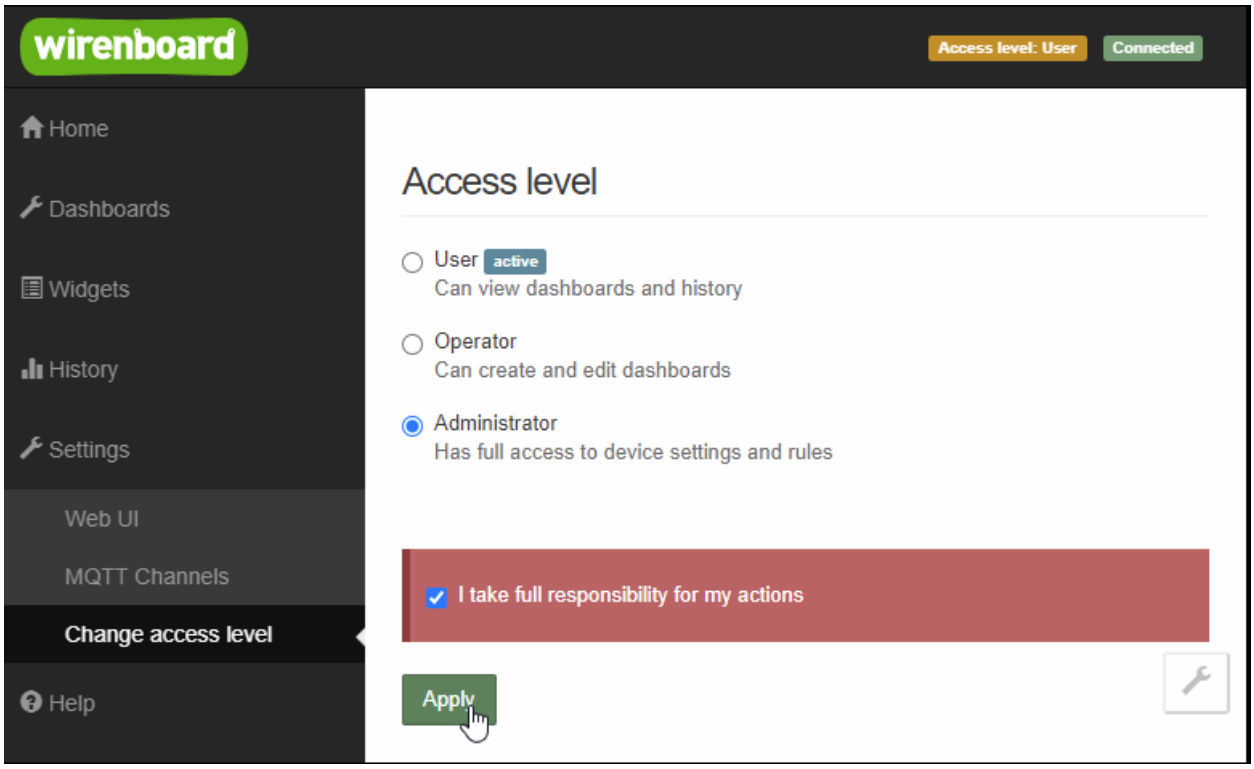


Рис. 2.17 – Заміна рівня доступу до веб-інтерфейсу

Для оптимальної конфігурації порту на контролері Wiren Board 6 за допомогою шини даних RS-485 ініціюється цілеспрямований процес через веб-інтерфейс. Цей процес включає в себе авторизацію в системі, здійснену шляхом введення IP-адреси контролера. Однак, перед внесенням будь-яких змін в конфігурації, необхідно переконатися, що рівень доступу до системи встановлено як "Administrator". Це можна зробити у вкладці "Settings", вибравши підменю "Change access level".

Після цього корисно перейти до секції "Settings", далі до підсекції "Configs" і знайти "Serial Device Driver Configuration". Тут можна вибрати конкретний порт, до якого фізично підключений пристрій. На цьому етапі активується порт, ставлячи відмітку в полі "Enable port" і вказуючи відповідні параметри підключення. У випадку, коли до одного порту підключено декілька пристроїв, важливо забезпечити їхню сумісність, налаштувавши однакові параметри підключення для всіх пристроїв. Це детально відображено на рис. 2.18.

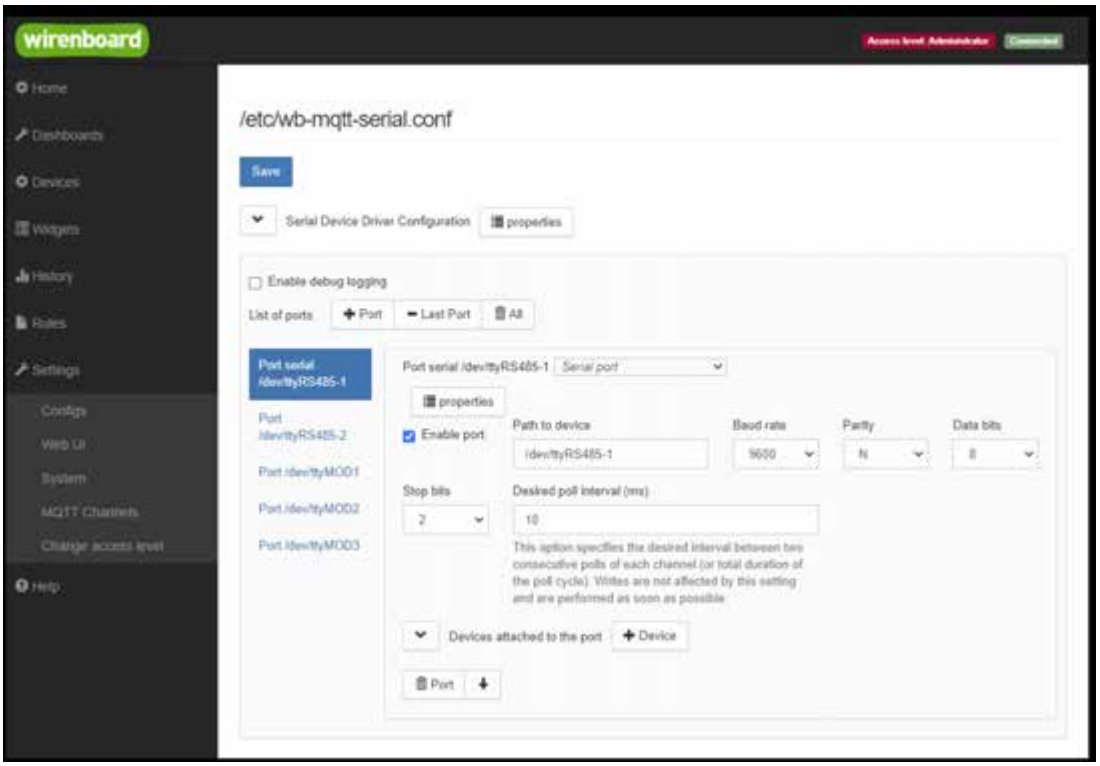


Рис. 2.18 – Налаштування підключень

По завершенню всіх налаштувань і для уникнення можливих помилок в роботі системи рекомендується знизити рівень доступу до веб-інтерфейсу, обравши роль "User" або "Operator". Така послідовність кроків не тільки забезпечує правильну конфігурацію порту, але і сприяє підвищенню рівня безпеки системи.

Для конфігурації пристроїв, інтегрованих з контролером Wiren Board 6, за допомогою веб-інтерфейсу та з використанням шини даних RS-485, необхідно виконати декілька послідовних кроків. Ініціація процесу налаштувань відбувається через авторизацію в веб-інтерфейсі контролера. Після цього для можливості внесення змін у системні налаштування необхідно переключити рівень доступу на "Administrator".

Після переключення рівня доступу слід перейти до конфігурації порту. Це реалізується у розділі "Settings", де вибирається підменю "Configs" та "Serial Device Driver Configuration". Тут активується порт, ставлячи відмітку у полі "Enable port", а також визначаються параметри підключення пристрою.

Наступним етапом є інтеграція пристрою у веб-інтерфейс контролера. Для цього у налаштуваннях порту вибирається опція "Devices attached to the

port" і через інтерфейс натискається кнопка "+Device". Якщо пристрій базується на відомому шаблоні, його можна легко знайти у каталозі підтримуваних пристроїв від сторонніх виробників.

Після успішного додавання пристрою його функціональність можна перевірити у вкладці "Devices". Для фіксації усіх внесених налаштувань і завершення процесу конфігурації необхідно використати кнопку "Save", що зображена на рис. 2.19. Таким чином, реалізована послідовність дій не лише забезпечує ефективну конфігурацію порту та пристроїв, але і сприяє загальній безпеці та стабільності системи.

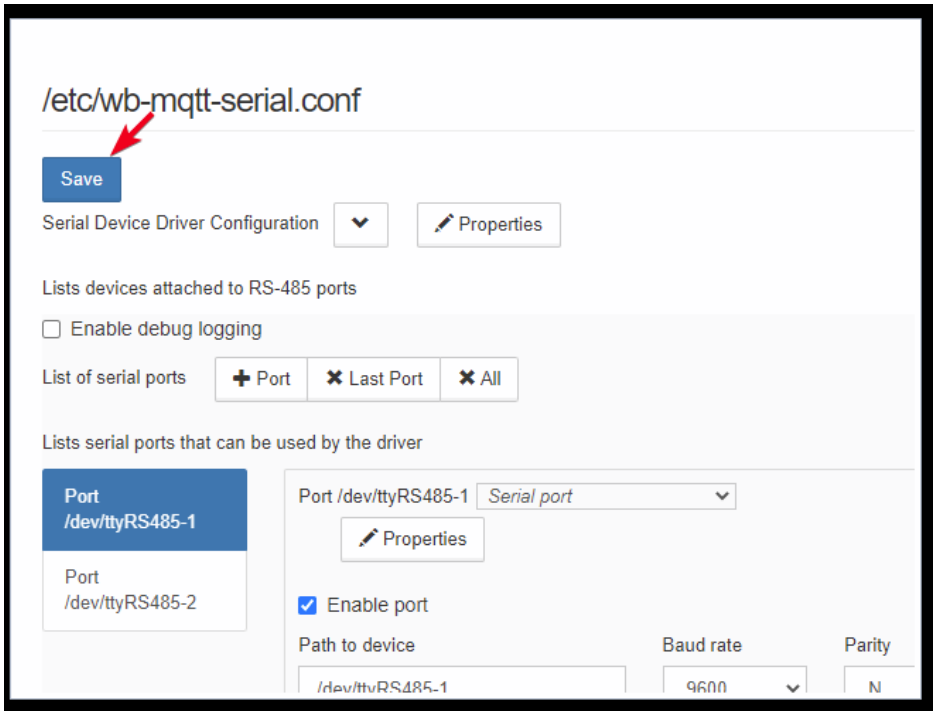


Рис. 2.19 – Збереження проведеного налаштування контролера

Процес конфігурації пристроїв в контексті використання контролера Wiren Board 6 через веб-інтерфейс відзначається високою ступенем зручності та інтуїтивності. Цей аспект є особливо значущим, оскільки інтерфейс надає користувачу можливість здійснювати глибоку персоналізацію системи, не обмежуючись лише налаштуванням портів і пристроїв, що до них підключені. Спектр доступних параметрів для конфігурації охоплює мережеві налаштування, часову зону, параметри апаратної частини та інші ключові аспекти системи.

Важливим доповненням до цього є наявність вбудованих шаблонів для роботи з пристроїв від сторонніх виробників, що додатково спрощує процес налаштування і робить його більш доступним. Така конфігураційна гнучкість дозволяє адаптувати систему до специфічних потреб користувача.

З огляду на вищезгадані фактори можна стверджувати, що веб-інтерфейс контролера Wiren Board 6 представляє собою інтуітивно зрозумілий та ефективний інструмент для налаштування системи. Це робить його підходящим для користувачів різного рівня технічної підготовки, включаючи тих, хто не має глибокого досвіду в конфігурації подібних систем.

# 3 ПЕРЕВІРКА ТА ВАЛІДАЦІЯ ПРОЕКТОВАНОЇ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ БЕЗПЕКИ

## 3.1 Дослідження характеристик проектованої системи

По завершенню етапу розробки критично важливим є проведення глибокого аналізу характеристик інтелектуальної системи моніторингу безпеки в домогосподарстві. Аналіз повинен охоплювати не лише технічні аспекти, але й ефективність, надійність, гнучкість системи, а також її спроможність до адаптації в різноманітних екстраординарних ситуаціях.

*Ефективність системи*

Проектована інтелектуальна система безпеки для розумного будинку демонструє високу ступінь ефективності, яка досягається через комплексну інтеграцію передових технологічних рішень. Серед таких рішень — контролер Wiren Board 6, високочутливі датчики, а також розширені алгоритми обробки даних і автоматизовані сценарії реагування на потенційні загрози. Ці компоненти спільно забезпечують не лише високу оперативність у зборі та аналізі даних, але і в сфері невідкладних реакцій на загрозові ситуації, що в цілому підвищує ефективність системи в контексті забезпечення безпеки житлового простору.

*Надійність системи*

У інтелектуальній системі безпеки розумного будинку, слід акцентувати увагу на збалансованому виборі якісних компонентів та передових технологій. Додатково, потрібно відзначити, що система працює на базі розроблених алгоритмів, що пройшли множинні етапи тестування та оптимізації. Така комплексна підготовка сприяла забезпеченню високої стабільності в режимі неперервної роботи. Ці фактори колективно знижують ймовірність несподіваних відмов та збоїв у системі, забезпечуючи тим самим високий рівень надійності.

*Гнучкість*

Гнучкість системи безпеки розумного будинку досягається за рахунок інтеграції веб-інтерфейсу, який надає користувачам засоби для деталізованого

налаштування ряду параметрів. Ця архітектурна рішення забезпечує можливість глибокої кастомізації, зокрема, в аспектах збору даних, управління виконавчими механізмами, та алгоритмів реагування на екстрені ситуації. Такий підхід не тільки збільшує адаптивні можливості системи до специфічних потреб користувача, але й відкриває шлях для подальших модифікацій і масштабування системи в майбутньому.

*Адаптивність*

Адаптивність системи безпеки розумного будинку проявляється в її спроможності до гнучкої інтеграції з різноманітними датчиками та виконавчими пристроями, які можуть бути додані до інфраструктури "розумного будинку". Така архітектурна відкритість не тільки спрощує швидке впровадження нових технологічних рішень, але й забезпечує можливість адаптації системи до змінюваних умов та потреб користувача. Додатково, система безпеки розроблена таким чином, що може легко синхронізуватися з іншими компонентами екосистеми "розумного будинку", що робить її вище адаптивною до широкого спектру технологічних сценаріїв.

*Масштабованість*

Масштабованість даної системи безпеки розумного будинку є однією з її ключових переваг, виражаючись у спроможності ефективно функціонувати в різних об'єктових умовах — від компактних житлових приміщень до обширних архітектурних комплексів. Це стає можливим завдяки її модульній архітектурі, яка передбачає гнучкість в додаванні нових компонентів або модифікації існуючих. Така архітектурна пластичність не лише сприяє адаптації системи до специфічних потреб та розмірів різних об'єктів, але також відкриває можливості для її вертикальної та горизонтальної масштабованості відповідно до зростаючих вимог безпеки.

*Зручність використання*

Зручність управління та ергономічність інтерфейсу є важливими аспектами розробленої інтелектуальної системи безпеки для розумного будинку. Інтуїтивно зрозумілий веб-інтерфейс забезпечує користувачам

швидкий доступ до всіх необхідних параметрів системи, дозволяючи з легкістю налаштувати її роботу та моніторити актуальний стан об'єкта. Така конструктивна простота не тільки полегшує процес первинного ознайомлення з системою, але і робить її експлуатацію максимально комфортною для користувача, незалежно від його технічної кваліфікації.

*Безпека*

Розроблена інтелектуальна система безпеки розумного будинку використовує алгоритм шифрування Advanced Encryption Standard (AES). Цей алгоритм відомий своєю високою стійкістю до криптографічних атак і є стандартом в індустрії кібербезпеки. Використання AES у системі значуще підвищує рівень захисту даних від несанкціонованого доступу та забезпечує інтегрітет персональної інформації користувачів. Така технологічна орієнтація на застосування передових методів шифрування дозволяє системі відповідати сучасним вимогам до інформаційної безпеки, а також забезпечує довіру до системи з боку кінцевих користувачів.

Комплексна оцінка атрибутів інтелектуальної системи моніторингу безпеки розумного будинку атестує її як високоефективний, надійний, гнучкий, адаптивний та масштабований інструментарій. Система не лише забезпечує інтуїтивну зручність управління та високий рівень захисту інформації для користувачів, але і володіє значущим потенціалом для імплементації в діверсифікованих оперативних контекстах. Це робить її ідеально придатною для використання як автономного заходу у сфері забезпечення безпеки приміщень, так і для інтеграції в більш об'ємні системні рішення в області "розумного будинку", орієнтовані на комплексне забезпечення комфорту, безпеки та енергоефективності.

Отже, розроблена інтелектуальна система моніторингу безпеки розумного будинку представляє собою передову технологічну концепцію, здатну реалізувати високий ступінь безпеки в резиденційних та комерційних структурах. Система є в повній відповідності з актуальними нормативами та прогностичними тенденціями в сфері технологій "розумного будинку" і може

стати ключовим компонентом для гарантованого забезпечення безпеки та комфортних умов проживання користувачів.

### 3.2 Експериментальна перевірка ефективності системи

Експериментальна перевірка є ключовим елементом у розробці програмних рішень, спрямованим на атестацію функціональних можливостей системи та її відповідності заздалегідь визначеним технічним вимогам. У контексті нашої роботи, основна увага була приділена тестуванню ефективності роботи навченої моделі на декількох наборах тестових даних. Головна задача цього етапу полягала у переконанні, що алгоритмічна модель коректно та адекватно реагує на потенційні безпекові загрози.

Для проведення тестування було сформовано спеціалізований датасет, який включав у себе вибіркові показники від датчиків, а також відповідні класифікаційні лейбли. Ці дані були архівовані в форматі CSV, що спростило їх подальшу обробку та інтеграцію в тестувальний процес (рис. 3.1).

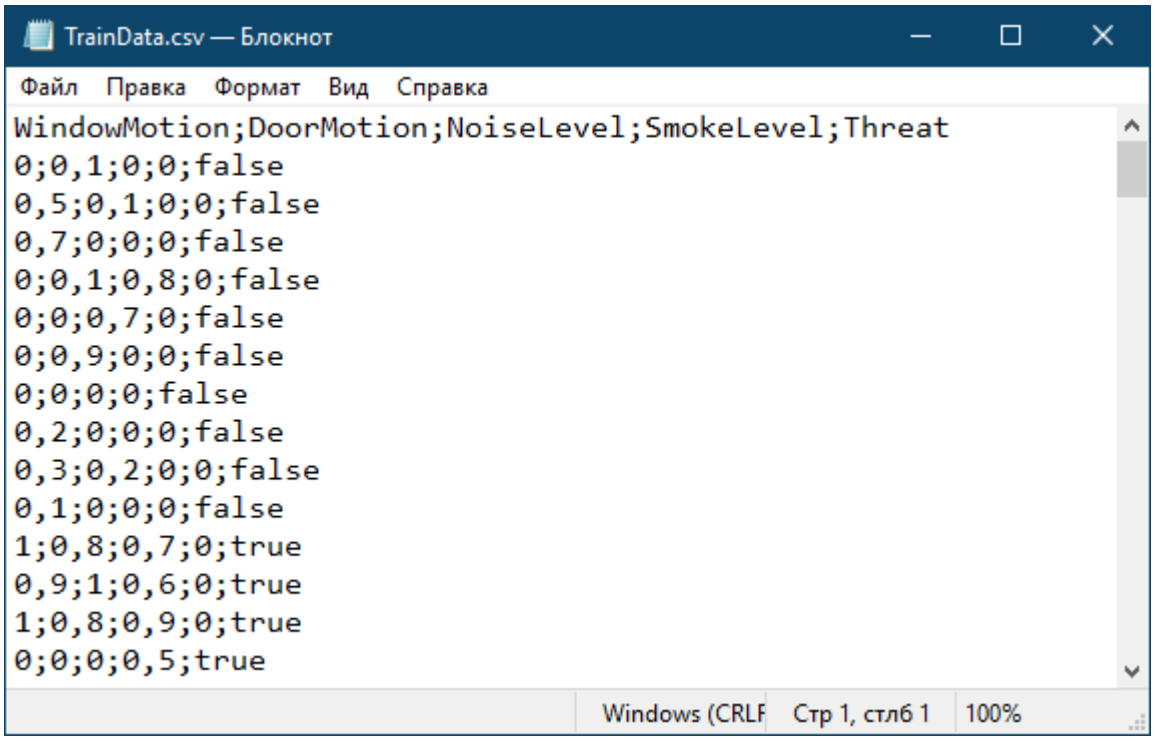


Рис. 3.1– Фрагмент навчальних даних НМ

Після завантаження відповідного CSV-файлу, було здійснено процес навчання нейронної мережі із використанням прописаного методологічного

підходу, деталі якого можна спостерігати у лівій частині інтерфейсу програми (рис. 3.2).

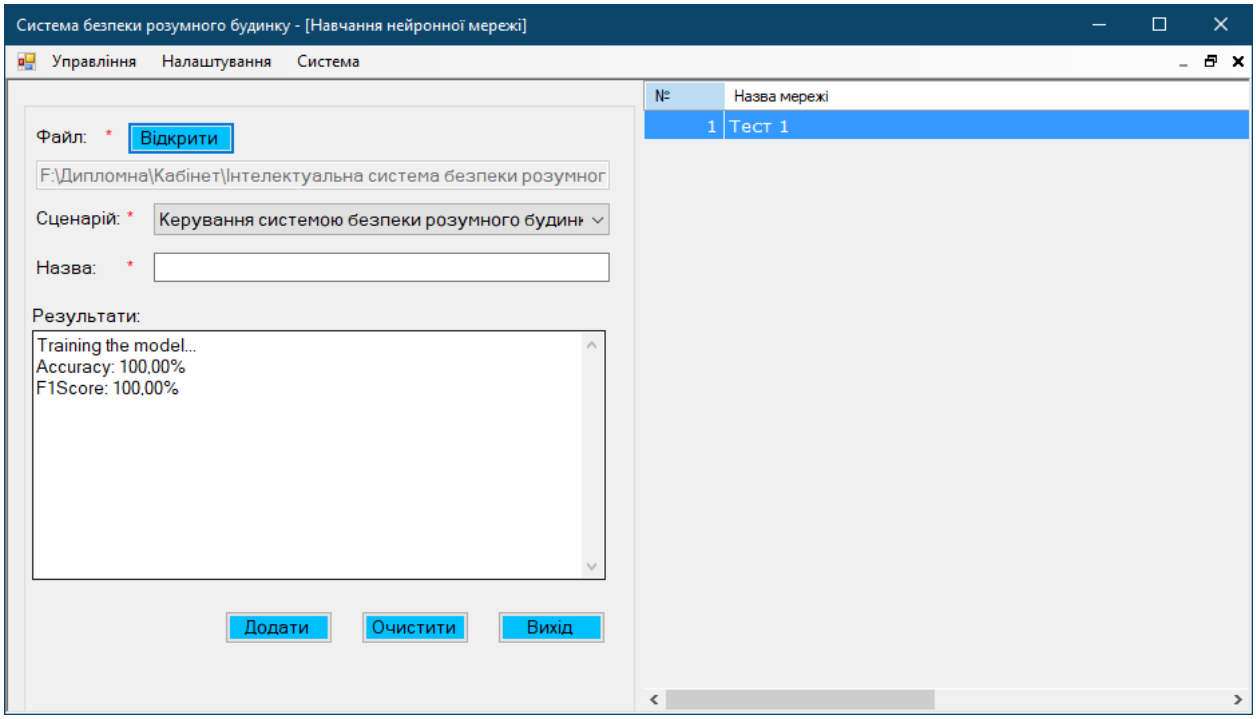


Рис. 3.2– Форма досягнутого результату навчання НМ

З метою кількісної характеристики ефективності навчання, система проводить розрахунок ключових метрик. Серед них:

– метрика точності (Accuracy) є показником, що визначає відсоткове співвідношення коректно класифікованих випадків до загальної кількості. У контексті даного дослідження, ця метрика досягла 100%, що на перший погляд може вважатися ідеальним показником. Втім, це не завжди є індикатором високої якості моделі, особливо у присутності незбалансованого набору даних;

– F1-оцінка (F1-Score) представляє собою гармонійне середнє між точністю та повнотою класифікаційного алгоритму. Ця метрика може приймати значення від 0 до 1, де 1 символізує найвищий рівень точності та повноти. У рамках проведеного аналізу, F1-оцінка також досягла показника в 100%, вказуючи на відмінну збалансованість та ефективність моделі.

По завершенню етапу навчання, наступним кроком була експериментальна верифікація роботи нейронної мережі. Для цього було здійснено тестування на основі даних, сгенерованих випадковим методом, які

імітували показники реальних датчиків (рис. 3.3). Цей експеримент мав на меті кількісно оцінити адекватність та точність роботи навченої моделі в умовах, що наближені до реального оперативного середовища.

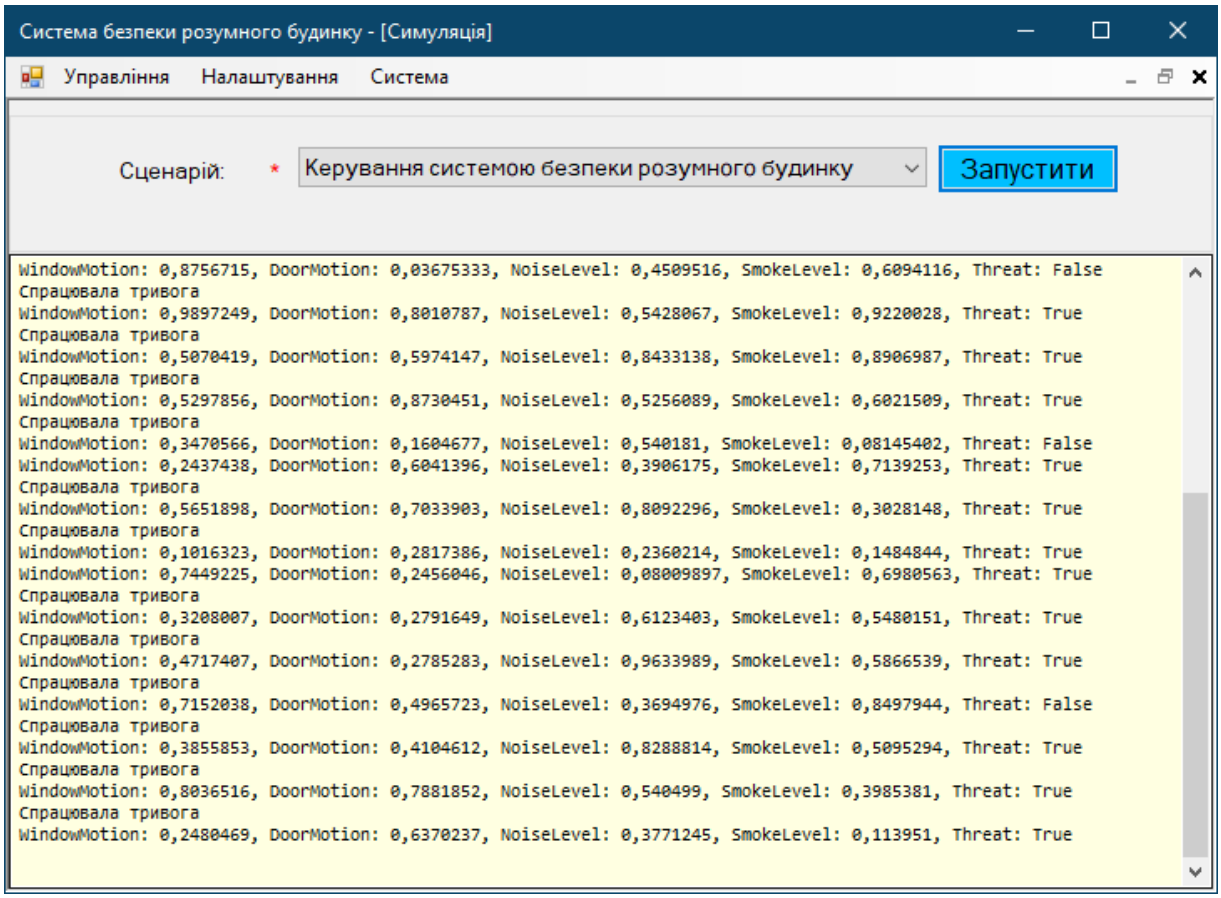


Рис. 3.3– Проведення експериментів із згенерованими даними

За допомогою дослідницького методу було оцінено реакцію системи на різні комбінації параметрів, таких як рух вікон (WindowMotion), рух дверей (DoorMotion), рівень шуму (NoiseLevel) та рівень диму (SmokeLevel). Параметр "Threat" служив для індикації потенційної загрози.

Аналіз результатів показує, що система в більшості випадків коректно ідентифікує стан загрози, активуючи тривогу. Однак, є інстанції, де система спрацьовує невірно — активує тривогу, незважаючи на відсутність реальної загрози, або не активує її при її наявності.

Специфічно, тривога була активована в 12 з 14 сценаріїв, включаючи випадки, де "Threat" було встановлено як "False". Це може свідчити про підвищену чутливість моделі та необхідність її додаткової калібруванні для зменшення кількості хибно-позитивних сигналів.

Особливо варто звернути увагу на метрику "Threat: True" у комбінації з активацією тривоги, яка свідчить про високу здатність системи правильно ідентифікувати реальні загрози.

Загалом, результати тестування нейронної мережі підкреслюють її потенціал для ефективного моніторингу безпеки, але також вказують на необхідність подальших налаштувань для оптимізації її роботи.

### 3.3 Аналіз потенційних ризиків для проектованої системи

Потенціальні загрози для системи інформаційної безпеки в архітектурі "розумного будинку" в значній мірі визначаються застосованими технологічними рішеннями та конфігурацією апаратних компонентів. В рамках цього дослідження, ми провели систематичний аналіз потенційних загроз, зосереджуючись на технологічних аспектах, які використовуються в централізованих системах управління.

Наступним етапом аналізу є корелятивна оцінка виявлених загроз з існуючими вразливостями, а також визначення можливого впливу на ключові атрибути інформаційної безпеки: конфіденційність (К), цілісність (Ц), та доступність (Д). Деталі представлено в табл. 3.1.

Таблиця 3.1 – Основні загрози безпеки для системи «розумний будинок»

| № | Тип загрози | Асоційована вразливість | К | Ц | Д |
|---|---|---|---|---|---|
| 1 | Інтрузії на сервер | Недостатній мережний захист | + | + | + |
| 2 | Шкідливе ПЗ | Відсутність захисних механізмів | + | + | + |
| 3 | Інтерцепція даних | Незахищена мережева комунікація | + | + | - |
| 4 | Неавторизований доступ | Недостатня автентифікація | + | - | - |
| 5 | Зловживання розробником | Слабка автентифікація | + | + | - |
| 6 | Виток ресурсів | Неоптимізоване розподілення | - | - | + |

| | | | | | |
|---|---|---|---|---|---|
| | | навантаження | | | |
| 7 | Обхід обмежень доступу | Недоліки в розмежуванні прав | + | + | - |
| 8 | Дезактивація сенсорів | Несправжній контроль доступу | - | - | + |
| 9 | Фізичний проникнення | Недостатній фізичний захист | + | + | + |
| 10 | Виведення обладнання з ладу | Незахищене зберігання | + | + | + |
| 11 | Руйнування обладнання | Недостатні фізичні заходи захисту | - | + | + |
| 12 | Відсутність електроенергії | Недостатнє резервне живлення | - | - | + |
| 13 | Операційні помилки користувача | Складний користувацький інтерфейс | - | + | + |
| 14 | Помилки програмного забезпечення | Використання неліцензійного програмного забезпечення | - | - | + |
| 15 | Катастрофи | Недостатній рівень фізичної безпеки | - | + | + |

Таблиця ризиків була розроблена згідно з методологічними принципами корпорації "Microsoft" для оцінки інформаційних ризиків. В даному контексті були враховані три ключові параметри: ймовірність маніфестації загрози, потенціальний рівень впливу на активи та категорія активів, які можуть бути зачеплені. Ймовірність виникнення кожної загрози була класифікована як висока, середня або низька, виходячи з частоти її потенційної реалізації. Щодо рівня впливу на активи, він також був ранжований на високий, середній та низький, з урахуванням можливого економічного та оперативного збитку.

Класифікація активів виконана на основі аналізу їх значущості для діяльності організації, в тому числі впливу на конфіденційність, цілісність та доступність інформаційних ресурсів.

Ці синтезовані дані з оцінки ризиків були застосовані для комплексної оцінки інформаційних ризиків в системі автоматизованого управління "Розумний будинок", що дозволило визначити ключові напрямки для подальшого забезпечення її інформаційної безпеки.

За висновками аналізу, сформовано таблицю, що слугує для якісної характеристики рівнів ризиків в системі "Розумний будинок" (табл. 3.2).

Таблиця 3.2 – Визначення рівнів впливу

| Класи | Високий вплив | Середній вплив | Низький вплив |
|---|---|---|---|
| Високий рівень схильності | Високий | Високий | Середній |
| Середній рівень схильності | Високий | Середній | Низький |
| Низький рівень схильності | Середній | Низький | Низький |

В табл. 3.2 перетин класу активу та рівня схильності до впливу дає можливість оцінити потенціальний рівень ризику. Наприклад, актив з високим рівнем схильності та високим впливом на організацію віднесено до категорії "Високий рівень ризику".

Таблиця 3.3 – Визначення підсумкового ризику

| Ймовірність | Високий вплив | Середній вплив | Низький вплив |
|---|---|---|---|
| Висока ймовірність | Високий | Високий | Середній |
| Середня ймовірність | Високий | Середній | Низький |
| Низька ймовірність | Середній | Низький | Низький |

У цій таблиці, аналогічно попередній, перетин впливу та рівня ймовірності реалізації загрози дозволяє класифікувати підсумковий ризик. Зокрема, якщо вплив на актив є високим, а ймовірність реалізації загрози також висока, ризик класифікується як високий.

Таблиця 3.4 – Рівень ризику для загроз «розумного будинку»

| Загроза | Вірогідність виникнення | Схильність виникнення | Категорія активу | Ступінь ризику |
|---|---|---|---|---|
| Атака на сервер | Висока | Висока | Високий | Високий |
| Інфекція шкідливим кодом | Висока | Висока | Високий | Високий |
| Перехоплення комунікацій | Висока | Середня | Середній | Високий |
| Несанкціонований доступ | Висока | Середня | Середній | Середній |
| Втручання розробника | Висока | Висока | Середній | Високий |
| Завантаження ресурсів | Висока | Середня | Низький | Середній |
| Взламування | Середня | Середня | Середній | Середній |
| Відключення сенсорів | Висока | Середня | Середній | Високий |
| Фізичне проникнення | Середня | Висока | Високий | Високий |
| Вихід з ладу | Низька | Висока | Високий | Середній |
| Знищення обладнання | Низька | Висока | Високий | Середній |
| Проблеми з електропостачанням | Середня | Середня | Середній | Середній |
| Помилки оператора | Середня | Середня | Середній | Середній |
| Помилки в коді | Середня | Середня | Середній | Середній |
| Катаклізми | Низька | Висока | Високий | Середній |

Згідно з проведеною аналітичною роботою з оцінки ризиків інформаційної безпеки системи моніторингу "Розумний будинок", можна виокремити декілька ключових загроз, які представляють наибільший ризик для стабільності та безпеки системи. Серед них:

– несанкціонований доступ до центрального сервера. Висока ймовірність такої атаки може призвести до повного або часткового зламу системи, що зробить її неефективною або недоступною для користувачів;

– імплантація шкідливого коду. Потенційна можливість внедрення в систему вірусів або троянських програм, які можуть збити роботу системи та отримати доступ до конфіденційної інформації;

– маніпуляція та перехоплення комунікаційних сигналів. Сценарії, при яких зловмисник може перехопити та/або підмінити сигнали між компонентами системи, що може призвести до неконтрольованих дій системи;

– експлуатація програмних засобів розробника. Ризик зловживання адміністративними правами для зміни конфігурації системи або внесення змін до її функціональності;

– неавторизоване відключення або маніпуляція з контрольними датчиками. Це може призвести до фальшивих тривожних сигналів або, навпаки, до нереагування системи на реальні загрози;

– компрометація фізичних захисних механізмів. Включає в себе ризик проникнення на об'єкт через подолання фізичних бар'єрів, таких як двері, вікна або інші засоби обмеження доступу.

Додатково до вищезазначених ключових загроз, не менш важливими є ризики, асоційовані з непередбачуваними або неконтрольованими факторами. Сюди входять:

– неадекватна робота систем електроживлення. Нестабільність або відсутність електропостачання може серйозно підривати функціональність системи, зробивши її неефективною або навіть недоступною;

– людський фактор у вигляді помилок користувача. Некоректна діяльність користувачів може призвести до неправильної роботи системи або до її злому;

– дефекти у програмному забезпеченні. Помилки в коді, некоректна робота програмного забезпечення може стати причиною не лише технічних, але й інформаційних ризиків.

В якості цих додаткових ризиків, рекомендується реалізація наступних захисних механізмів для атенуації потенційних загроз:

– механізми ідентифікації та аутентифікації. Необхідність реалізації сильних механізмів ідентифікації для забезпечення того, що доступ до системи мають лише уповноважені особи;

– криптографічні механізми. Використання сучасних алгоритмів шифрування для захисту конфіденційності та цілісності даних;

– антивірусне програмне забезпечення. Основний акцент на використанні надійних антивірусних програм для сканування та нейтралізації шкідливого коду;

– система контролю доступу. Реалізація деталізованих правил для керування доступом до різних компонентів системи;

– механізми балансування навантаження. Оптимізація роботи системи через розподіл ресурсів, особливо в пікові періоди;

– періодична аудиторська перевірка. Регулярний моніторинг та оцінка працездатності всіх елементів системи для виявлення можливих слабких місць;

– резервне електроживлення. Встановлення надійних джерел резервного живлення для забезпечення неперервної роботи системи в умовах нестабільності або відсутності основного електропостачання.

Базуючись на проведеному аналізі ризиків, можна виокремити наступні найбільш критичні загрози:

– несанкціонований доступ до центрального сервера. Це може призвести до порушення конфіденційності, цілісності та доступності інформації;

– внесення шкідливого коду або програм. Шкідливий код може викликати непередбачувані наслідки, зокрема знищення даних або несанкціонований доступ до системи;

– інтерцепція та альтерація комунікаційних сигналів. Перехоплення сигналів може призвести до втрати конфіденційності або цілісності переданих даних;

– зловживання розробницькими привілеями. Це може забезпечити доступ до конфіденційної інформації або можливість змінити функціональність системи;

– вимкнення або маніпуляція контрольними датчиками. Таке втручання може призвести до невірної роботи системи, в тому числі спрацювання тривожних сигналів;

– компрометація фізичної безпеки об'єкта. Несанкціонований фізичний доступ може призвести до тотальної компрометації системи;

– нестабільність електропостачання та програмні дефекти. Ці фактори можуть впливати на стабільність роботи системи, і, як наслідок, на інформаційну безпеку.

Для мінімізації ризиків, асоційованих з цими загрозами, рекомендується:

– впровадження системи аутентифікації та ідентифікації. Для забезпечення того, що лише уповноважені користувачі мають доступ до системи;

– застосування криптографічних методів. Для захисту конфіденційності та цілісності переданих даних;

– імплементація антивірусних програм. Для захисту від шкідливого коду;

– створення системи управління доступом. Для контролю і моніторингу доступу до ресурсів системи;

– використання механізмів балансування навантаження. Для забезпечення стабільної роботи системи;

– періодична діагностика системи. Для виявлення та виправлення можливих несправностей.

– впровадження резервних систем живлення. Для забезпечення неперервної роботи в умовах нестабільності основного джерела енергії.

## 3.4 Оцінка практичної придатності та шляхи подальшого вдосконалення

Система моніторингу безпеки для "розумного будинку" представляє собою важливий науково-технічний напрямок з великим потенціалом для практичної реалізації та наукового удосконалення. Ця технологія може служити ефективним інструментом для забезпечення безпеки не лише в житловому секторі, але і в сфері комерції та громадських просторах. Розглянемо детальніше можливості практичного застосування та перспективи розвитку системи.

*Практичне застосування*

Система моніторингу безпеки розумного будинку може бути впроваджена у наступних сферах:

– житловий сектор. Інтелектуальна система безпеки може значно покращити рівень захисту приватних домогосподарств. Система може автоматизувати процеси контролю доступу, активувати тривожні сповіщення у випадках незаконного вторгнення та інших небезпечних ситуацій;

– комерційні об'єкти та офісні простори. В даному випадку, система може служити для забезпечення конфіденційності корпоративних даних, інтегритету технічного обладнання, а також для захисту персоналу. Додатково, система може контролювати доступ до стратегічно важливих зон приміщення;

– громадська інфраструктура. Застосування інтелектуальної системи безпеки на міському рівні може сприяти підвищенню рівня безпеки в громадських локаціях, таких як парки, площі, транспортні переплетення.

*Перспективи для подальшого вдосконалення та розвитку*

Система моніторингу безпеки розумного будинку також має потенціал для подальшого розвитку та вдосконалення, зокрема:

– мультидисциплінарна інтеграція. Система може бути синтезована з іншими інтелектуальними технологіями, такими як системи енергоефективності, автоматичне регулювання освітлення та клімат-контроль.

– розширення функціоналу. Нові модулі можуть бути добавлені до системи, включаючи біометричні методи аутентифікації, відеоаналітичні інструменти, та автоматизовану систему оцінки ризиків;

– використання технологій інтернету речей. Інтеграція IoT-технологій може дозволити більш ефективний збір даних та автоматизацію управління системою;

– оптимізація комунікативних протоколів. Вдосконалення стандартів комунікації між компонентами може поліпшити надійність системи та ефективність передачі даних;

– посилення кібербезпеки. У контексті зростання кіберзагроз, актуалізується необхідність постійного удосконалення механізмів кіберзахисту;

– стандартизація та сертифікація. Розробка уніфікованих стандартів та процедур оцінки якості може сприяти глобалізації та широкому впровадженню технологій "розумного будинку".

Отже, інтелектуальна система безпеки "розумного будинку" представляє значний інтерес для наукової спільноти та має великий потенціал для практичної реалізації, що може суттєво вдосконалити якість життя населення.

# ВИСНОВКИ

Завдання дослідження системи моніторингу безпеки будівлі було максимально комплексним і охоплювало не лише теоретичну, але і практичну складові. У першому розділі роботи акцент зроблено на ретельному аналізі основних теоретичних принципів систем безпеки у розумних будинках. Було вивчено два ключові архітектурні підходи: централізована та децентралізована структура, а також здійснено глибокий аналіз методів штучного інтелекту в контексті систем розумної безпеки. Це дало можливість не лише оцінити потенційні ризики, але й зрозуміти, на які аспекти безпеки слід зосередитися.

Особливу увагу приділено оцінці ключових ризиків, включаючи фізичні, кібернетичні, а також соціальні та психологічні аспекти загроз. Це включало аналіз сучасних методів забезпечення безпеки, а також аналіз недоліків сучасних систем, що підкреслило потребу в нових стратегіях та інноваційних підходах.

У другому розділі роботи основний фокус був зроблений на проектуванні концептуальних засад інтелектуальної системи безпеки. Розроблено алгоритм ідентифікації потенційних загроз на основі машинного навчання, що використовувало бібліотеку ML .NET. Було визначено фундаментальні функції та елементи системи, зокрема критерії вибору керуючого блоку та модульних компонентів. Це дало можливість здійснити імплементацію ключових модулів системи, описану на мові програмування C#.

Третій розділ роботи було присвячено детальній перевірці та валідації розробленої системи. Було проведено ряд експериментів, які не лише підтвердили високу ефективність системи, але і виявили потенційні вразливості, що забезпечило можливість для подальшого її вдосконалення.

Розроблена інтелектуальна система безпеки розумного будинку демонструє високий рівень ефективності та надійності, що підтверджено реальними експериментами. Отримані результати дозволяють говорити про великий потенціал системи не лише в академічному, але і в практичному аспекті. Виконана робота стала основою для розробки конкретних

рекомендацій щодо подальшого вдосконалення системи, її адаптації до нових технологій та відповіді на сучасні загрози. Особливо важливою є можливість широкого використання розробленої системи в різних сферах: від приватних будинків до бізнес-центрів та громадських просторів. Це забезпечує не лише підвищення рівня безпеки, але і дає змогу оптимізувати ресурси та знизити економічні витрати на забезпечення безпеки.

Важливо підкреслити, що розроблена система є гнучкою та адаптивною, що дозволяє їй інтегруватися з іншими системами розумного будинку, включаючи системи енергозбереження, автоматизації освітлення та системи контролю клімату. Це відкриває нові перспективи для створення єдиної, інтегрованої системи безпеки, що може ефективно протистояти різноманітним загрозам на різних рівнях.

Також слід зазначити, що проведені експерименти та аналіз потенційних ризиків стали основою для розробки комплексу захисних заходів, що спрямовані на зниження ризиків та підвищення стабільності системи. Отже, в рамках даної роботи було зроблено великий крок вперед у розробці ефективних і надійних систем безпеки для розумних будинків. Розроблена система не лише відповідає сучасним вимогам і тенденціям, але й покладає основу для подальших наукових досліджень та практичних реалізацій в цій сфері.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Patrascu M. Integrating Services and Agents for Control and Monitoring: Managing Emergencies in Smart Buildings. Service Orientation in Holonic and MultiAgent Manufacturing and Robotics. / Patrascu., 2014. – 544 с.

2. Granzer W. P. Security in Building Automation Systems / Wolfgang Praus Granzer. Munich: Appress, 2018. – 578 с.

3. Dickson B. How to prevent your IoT devices from being forced into botnet bondage / Dickson. – 2015. URL: https://techcrunch.com/2016/08/16/how-to-prevent-your-iot-devices-from-being- forcedinto-botnet-slavery/ (дата звернення: 10.09.2023).

4. Power Load Event Detection and Classification Based on Edge Symbol Analysis and Support Vector Machine . – 2014. URL: https://www.hindawi.com/journals/acisc/2012/742461/ (дата звернення: 15.09.2023).

5. An Overview of Home Automation Systems– 2017. URL: https://ieeexplore.ieee.org/document/7791223/ (дата звернення: 14.09.2023).

6. Internet of Things (IoT). URL: https://www.it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot (дата звернення: 19.09.2023).

7. Radivojevic G., Bjelic N., Popovic D. Internet of things in logistics. 3rd Logistics International Conference. Serbia, Belgrade 2019. №3. Pp. 185–190.

8. RL-IoT. URL: https://ieeexplore.ieee.org/document/9905718 (дата звернення: 17.09.2023).

9. What is artificial intelligence of things (AIoT)? URL: https://www.techtarget.com/iotagenda/definition/Artificial-Intelligence-of-Things-AIoT (дата звернення: 04.10.2023).

10. Artificial Intelligence of Things (AIoT) – Trends and Applications in 2023. URL: https://viso.ai/edge-ai/artificial-intelligence-of-things-aiot/ (дата звернення: 21.09.2023).

11. L. Huang et al., “A deep reinforcement learning-based method applied for solving multi-agent defense and attack problems,” Expert Syst. Appl., vol. 176.

12. A Review of Techniques and Policies on Cybersecurity Using Artificial Intelligence and Reinforcement Learning Algorithms, September 2023IEEE Technology and Society Magazine 42(3):57-68

13. МОРОЗОВА, О. ., ТЕЦЬКИЙ, А. ., НІЧЕПОРУК, А. ., КРИВАК, Д. ., & ТКАЧОВ, В. . (2022). ОЦІНКА РИЗИКІВ БЕЗПЕКИ СИСТЕМИ РОЗУМНОГО БУДИНКУ. *Computer Systems and Information Technologies*, (3), 81–88. URL: https://doi.org/10.31891/CSIT-2021-5-11 (дата звернення: 27.09.2023).

14. Belova, A., & Onischenko, V. (2019). МЕТОДИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ РОЗУМНОГО БУДИНКУ. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, *2*(6), 134–141. URL: https://doi.org/10.28925/2663-4023.2019.6.134141 (дата звернення: 22.09.2023).

15. Дерев’янко Ю. В. Дослідження можливостей «інтелектуального будинку» / Ю. В. Дерев’янко, О. Л. Краснікова // Право і Безпека. – 2010. – № 1. – С. 223-226. – URL: https://cutt.ly/DbmPIO9 (дата звернення: 23.09.2023).

16. Інтернет речей (IoT) – що це таке і як працює, суть, технології. URL: https://termin.in.ua/internet-rechey-iot/ (дата звернення: 26.09.2023).

17. *Artificial intelligence, AI)*. URL: https://www.it.ua/knowledge-base/technology-innovation/artificial-intelligence (дата звернення: 30.09.2023).

18. Machine Learning, ML *)*. URL: https://www.it.ua/knowledge-base/technology-innovation/machine-learning (дата звернення: 01.10.2023).

19. Волошко С.В., Курца Д.О. Інформаційна безпека в безпроводових сенсорних мережах. ПолтНТУ, 201

20. Інтеграція засобів криптографічного захисту інформації. URL: https://iit.com.ua/ (дата звернення: 19.09.2023).

21. Geo-fence. URL: https://www.techtarget.com/whatis/definition/geofencing (дата звернення: 03.09.2023).

22. Кучеренко А. О. «Бездротові передавачі камер відеоспостереження в інформаційно-охоронній системі «Розумний будинок» // Science Online. International Electronic Scientific Journal. – 2018. – № 7. – 5 с. – URL: https://u.to/03VGGw (дата звернення: 25.09.2023).

23. Wiren Board 6. URL: https://wirenboard.com/ru/product/wiren-board-6/ (дата звернення: 25.09.2023).

24. Qadori H.Q. et al. A spawn mobile agent itinerary planning approach for energy-efficient data gathering in wireless sensor networks // Sensors (Switzerland). 2017. Vol. 17, № 6.

25. Ponde S., Lomte S. An Energy-Efficient MAC Protocols for Wireless Sensor Networks // Advances in Intelligent Systems and Computing. 2020. Vol. 1025.

26. Kuntz R., Gallais A., Noël T. Auto-adaptive MAC for energy-efficient burst transmissions in wireless sensor networks // 2011 IEEE Wireless Communications and Networking Conference, WCNC 2011. 2011.

27. Донцов І. Д. Використання штучного інтелекту в домашній автоматизації та енергозбереженні / І. Д. Донцов, О. М. Безвесільна // Погляд у майбутнє приладобудування : XI всеукр. наук.-практ. конф. студентів та аспірантів (15-16 трав. 2018 р., КПІ ім. І. Сікорського). – К., 2018. – С. 505-508. – URL: https://u.to/MXRGGw (дата звернення: 27.09.2023).

28. Abdallah Z. Activity Recognition with Evolving Data Streams: A Review / Z. Abdallah, M. Medhat // ACM Computing Surveys (CSUR) / Z. Abdallah, M. Medhat., 2018. – С. 1–36.

29. Adams, C. E. (2002). Home area network technologies. BT Technology Journal, 20(2), 53–72.

30. What is ML.NET and how does it work? URL: https://learn.microsoft.com/uk-ua/dotnet/machine-learning/how-does-mldotnet-work (дата звернення: 28.09.2023).

# ДОДАТКИ

## Додаток А. Скрипти створення бази даних

```
USE [master]
GO
/****** Object:  Database [DB]    Script Date: 12.10.2023 16:53:22 ******/
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
ALTER DATABASE [DB] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [DB] SET ANSI_NULLS OFF
GO
ALTER DATABASE [DB] SET ANSI_PADDING OFF
GO
ALTER DATABASE [DB] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [DB] SET ARITHABORT OFF
GO
ALTER DATABASE [DB] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [DB] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [DB] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [DB] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [DB] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [DB] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [DB] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [DB] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [DB] SET RECURSIVE_TRIGGERS OFF
```

```
GO
ALTER DATABASE [DB] SET  DISABLE_BROKER
GO
ALTER DATABASE [DB] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [DB] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [DB] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [DB] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [DB] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [DB] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [DB] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [DB] SET RECOVERY SIMPLE
GO
ALTER DATABASE [DB] SET  MULTI_USER
GO
ALTER DATABASE [DB] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [DB] SET DB_CHAINING OFF
GO
ALTER DATABASE [DB] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [DB] SET TARGET_RECOVERY_TIME = 0 SECONDS
GO
ALTER DATABASE [DB] SET DELAYED_DURABILITY = DISABLED
GO
USE [DB]
GO
/****** Object:  Table [dbo].[Logs]    Script Date: 12.10.2023 16:53:22 ******/
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,
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].[Neural]    Script Date: 12.10.2023 16:53:22 ******/
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Neural](
	[NeuralId] [int] IDENTITY(1,1) NOT NULL,
	[NeuralNames] [nvarchar](50) NULL,
	[ScenariosId] [int] NULL,
	[NeuralFileModel] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[NeuralId] 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].[Scenarios]    Script Date: 12.10.2023 16:53:22 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Scenarios](
	[ScenariosId] [int] IDENTITY(1,1) NOT NULL,
	[ScenariosName] [nvarchar](50) NULL,
	[Description] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[ScenariosId] 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: 12.10.2023 16:53:22 ******/
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](50) NULL,
	[RoleId] [int] NULL,
	[Description] [nvarchar](1000) NULL,
	[Email] [nvarchar](150) 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]
GO
```

```
ALTER TABLE [dbo].[Logs]  WITH CHECK ADD  CONSTRAINT [FK_Logs_Users] FOREIGN
KEY([UsersId])
REFERENCES [dbo].[Users] ([UsersId])
GO
ALTER TABLE [dbo].[Logs] CHECK CONSTRAINT [FK_Logs_Users]
GO
ALTER TABLE [dbo].[Neural]  WITH CHECK ADD  CONSTRAINT [FK_Neural_Scenarios]
FOREIGN KEY([ScenariosId])
REFERENCES [dbo].[Scenarios] ([ScenariosId])
GO
ALTER TABLE [dbo].[Neural] CHECK CONSTRAINT [FK_Neural_Scenarios]
GO
USE [master]
GO
ALTER DATABASE [DB] SET  READ_WRITE
GO
```

## Додаток Б. Код для керування контролером системи моніторингу

```
const ModbusRTU = require("modbus-serial");
const axios = require("axios");
const client = new ModbusRTU();

const sensorIDs = {
    GAS_SENS_ID: 1,
   WINDOW_SENS_ID: 2,
   DOOR_SENS_ID: 3,
   NOISE_SENS_ID: 4
};

const portConfig = {
   port: "/dev/ttyUSB0",
   baudRate: 9600,
   dataBits: 8,
   parity: "none",
   stopBits: 1
};

function defineDevice(deviceName, deviceTitle) {
   defineVirtualDevice(deviceName, {
      title: deviceTitle,
      cells: {
         value: {
            type: "value",
            value: 0
         }
      }
   });
}

function defineAlertRule(deviceName) {
   defineRule(`${deviceName}_rule`, {
      whenChanged: `${deviceName}/value`,
      then: async function (newValue, devName, cellName) {
         await axios.post('http://myHome.com/data', {
            device: devName,
            cell: cellName,
            value: newValue
         });
      }
   });
}

["gas_detector", "window_detector", "door_detector", "noise_detector"].forEach(device => {
   defineDevice(device, `${device.replace("_", " ").toUpperCase()} Detector`);
   defineAlertRule(device);
});
```

```
const nodemailer = require('nodemailer');

async function sendAlert(message) {
    try {
        // Зчитування налаштувань з зовнішнього файлу або змінних середовища
        const emailConfig = require('./emailConfig.json');  // або process.env.EMAIL_CONFIG

        const transporter = nodemailer.createTransport({
            service: emailConfig.service,
            auth: {
                user: emailConfig.user,
                pass: emailConfig.pass
            }
        });

        const mailOptions = {
            from: emailConfig.user,
            to: 'destination@example.com',  // Одержувач
            subject: 'Security Alert', // Тема
            text: message // Текст повідомлення
        };

        const info = await transporter.sendMail(mailOptions);

        console.log(`Email sent: ${info.response}`);
    } catch (error) {
        console.error(`Error sending email: ${error}`);
    }
}
module.exports = {
    getSensorData,
    sendAlert
};

const express = require("express");
const app = express();
const port = 3000;

const { getSensorData, sendAlert } = require("./controller");

app.use(express.static("public"));
app.use(express.json());

app.get("/api/sensorData", async (req, res) => {
    const sensorData = await getSensorData();
    res.json(sensorData);
});

app.post("/api/sendAlert", async (req, res) => {
    await sendAlert();
    res.sendStatus(200);
});
```

```
app.listen(port, () => {
   console.log(`Server listening at http://localhost:${port}`);
});

async function getSensorData() {
   await client.connectRTUBuffered(portConfig.port, portConfig);
   try {
      const slaveID = 1;
      const readings = {};

      for (const [sensorName, sensorID] of Object.entries(sensorIDs)) {
         const sensorData = await client.readInputRegisters(slaveID, sensorID, 1);
         readings[sensorName] = sensorData.data[0];
      }

      client.close();
      return readings;
   } catch (error) {
      console.error("Error reading sensor data:", error);
      client.close();
      throw error;
   }
}
```

## Додаток В. Лістинги програми

Лістинг 1. Код класу «NeuralProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Configuration;
using System.Linq;

namespace SecurityApp.Providers
{
   class NeuralProvider
   {
      private readonly string _ConnString = ConfigurationManager.AppSettings["CONNECT"];

      public void InsertNeural(string NeuralNames, int ScenariosId, string NeuralFileModel)
      {
         string SqlString = "INSERT INTO Neural (NeuralNames, ScenariosId, NeuralFileModel)
VALUES (?, ?, ?)";

         using (OleDbConnection conn = new OleDbConnection(_ConnString))
         using (OleDbCommand cmd = new OleDbCommand(SqlString, conn))
         {
            cmd.CommandType = CommandType.Text;
            cmd.Parameters.AddWithValue("NeuralNames", NeuralNames);
            cmd.Parameters.AddWithValue("ScenariosId", ScenariosId);
            cmd.Parameters.AddWithValue("NeuralFileModel", NeuralFileModel);
            conn.Open();
            cmd.ExecuteNonQuery();
         }
      }

      public List<Neural> GetAllNeural()
      {
         string SqlString = "SELECT * FROM Neural";

         using (OleDbConnection conn = new OleDbConnection(_ConnString))
         using (OleDbCommand cmd = new OleDbCommand(SqlString, conn))
         {
            conn.Open();
            using (OleDbDataReader reader = cmd.ExecuteReader())
            {
               return Enumerable.Range(1, int.MaxValue)
                  .Select(i => new Neural { Number = i })
                  .TakeWhile(reader.Read())
                  .Select(oneNeural => new Neural
                  {
                     NeuralId = Convert.ToInt32(reader["NeuralId"]),
                     NeuralNames = reader["NeuralNames"].ToString(),
                     ScenariosId = Convert.ToInt32(reader["ScenariosId"]),
                     NeuralFileModel = reader["NeuralFileModel"].ToString()
```

```
                })
                .ToList();
            }
        }
    }

    // Інші методи залишаються без змін.

  }
}

public class Neural
{
  public int Number { get; set; }
  public int NeuralId { get; set; }
  public string NeuralFileModel { get; set; }
  public int ScenariosId { get; set; }
  public string NeuralNames { get; set; }
  public string Message { get; set; }
}
```

Лістинг 2. Код класу «ScenariosProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Configuration;
using System.Linq;

namespace SecurityApp.Providers
{
  class ScenariosProvider
  {
    private readonly string _ConnString = ConfigurationManager.AppSettings["CONNECT"];

    public void InsertScenarios(string ScenariosName, string Description)
    {
      string SqlString = "INSERT INTO Scenarios (ScenariosName, Description) VALUES (?,
?)";

      using (OleDbConnection conn = new OleDbConnection(_ConnString))
      using (OleDbCommand cmd = new OleDbCommand(SqlString, conn))
      {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("ScenariosName", ScenariosName);
        cmd.Parameters.AddWithValue("Description", Description);
        conn.Open();
        cmd.ExecuteNonQuery();
      }
    }

    public List<Scenarios> GetAllScenarios()
```

```
{
   string SqlString = "SELECT * FROM Scenarios ORDER BY ScenariosName ASC";

   using (OleDbConnection conn = new OleDbConnection(_ConnString))
   using (OleDbCommand cmd = new OleDbCommand(SqlString, conn))
   {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader())
      {
         return Enumerable.Range(1, int.MaxValue)
            .Select(i => new Scenarios { Number = i })
            .TakeWhile(reader.Read())
            .Select(oneScenarios => new Scenarios
            {
               ScenariosId = Convert.ToInt32(reader["ScenariosId"]),
               ScenariosName = reader["ScenariosName"].ToString(),
               Description = reader["Description"].ToString()
            })
            .ToList();
      }
   }
}

public Scenarios SelectedScenariosByScenariosId(int ScenariosId)
{
   string SqlString = "SELECT * FROM Scenarios WHERE ScenariosId = ?";

   using (OleDbConnection conn = new OleDbConnection(_ConnString))
   using (OleDbCommand cmd = new OleDbCommand(SqlString, conn))
   {
      cmd.Parameters.AddWithValue("ScenariosId", ScenariosId);
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader())
      {
         if (reader.Read())
         {
            return new Scenarios
            {
               ScenariosId = Convert.ToInt32(reader["ScenariosId"]),
               ScenariosName = reader["ScenariosName"].ToString(),
               Description = reader["Description"].ToString()
            };
         }
         else
         {
            return new Scenarios
            {
               ScenariosId = 0,
               Message = NamesMy.NoDataNames.NoDataInScenarios
            };
         }
      }
```

```
        }
    }

    public void UpdateScenarios(string ScenariosName, string Description, int ScenariosId)
    {
        string SqlString = "UPDATE Scenarios SET ScenariosName=?, Description=? WHERE
ScenariosId=?";

        using (OleDbConnection conn = new OleDbConnection(_ConnString))
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn))
        {
            cmd.CommandType = CommandType.Text;
            cmd.Parameters.AddWithValue("ScenariosName", ScenariosName);
            cmd.Parameters.AddWithValue("Description", Description);
            cmd.Parameters.AddWithValue("ScenariosId", ScenariosId);
            conn.Open();
            cmd.ExecuteNonQuery();
        }
    }

    public void DeleteScenariosByScenariosId(int ScenariosId)
    {
        string SqlString = "DELETE FROM Scenarios WHERE ScenariosId = ?";

        using (OleDbConnection conn = new OleDbConnection(_ConnString))
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn))
        {
            cmd.Parameters.AddWithValue("ScenariosId", ScenariosId);
            conn.Open();
            cmd.ExecuteNonQuery();
        }
    }
  }
}

public class Scenarios
{
  public int Number { get; set; }
  public int ScenariosId { get; set; }
  public string Description { get; set; }
  public string ScenariosName { get; set; }
  public string Message { get; set; }

  public Scenarios()
  {
    Number = 0;
    ScenariosId = 0;
    Description = String.Empty;
    ScenariosName = String.Empty;
    Message = String.Empty;
  }
}
```

Лістинг 3. Код класу «SimulationForm»

```
using Microsoft.ML;
using SecurityApp.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SecurityApp.Forms.Controls
{
   public partial class SimulationForm : Form
   {
      private static Random random = new Random();
      private PredictionEngine<SensorInput, SensorOutput> predictor;
      private readonly NeuralProvider _NeuralProvider = new NeuralProvider();
      private Neural _SelectedNeural = new Neural();

      private readonly ScenariosProvider _ScenariosProvider = new ScenariosProvider();
      private List<Scenarios> _ScenariosList = new List<Scenarios>();
      private bool _IsThemesLoad = false;

      public SimulationForm()
      {
         InitializeComponent();
         LoadAllData();
      }

      private void LoadData(string FilePath)
      {
      }

      private void LoadAllData()
      {
         _ScenariosList = _ScenariosProvider.GetAllScenarios();
         ScenariosCBox.DataSource = _ScenariosList;
         ScenariosCBox.ValueMember = "ScenariosId";
         ScenariosCBox.DisplayMember = "ScenariosName";
         _IsThemesLoad = true;
         ScenariosCBox_SelectedValueChanged(ScenariosCBox, EventArgs.Empty);
      }

      public SensorInput GenerateRandomInput()
      {
         return new SensorInput
         {
            WindowMotion = (float)random.NextDouble(),
```

```
            DoorMotion = (float)random.NextDouble(),
            NoiseLevel = (float)random.NextDouble(),
            SmokeLevel = (float)random.NextDouble(),
            Threat = random.Next(2) == 0
        };
    }

    public bool Predict(SensorInput input)
    {
        var result = predictor.Predict(input);
        return Convert.ToBoolean(result.Prediction);
    }

    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 ScenariosCBox_SelectedValueChanged(object sender, EventArgs e)
    {
        if (_IsThemesLoad)
        {
            _SelectedNeural =
_NeuralProvider.SelectedNeuralByScenariosId(Convert.ToInt32(ScenariosCBox.SelectedValue));
            LoadData(_SelectedNeural.NeuralFileModel);
        }
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        try
        {
            string localProj = Application.StartupPath + _SelectedNeural.NeuralFileModel;
            var generator = new SensorInputGenerator();
            var predictor = new SensorPredictor(localProj);
            var randomInput = generator.GenerateRandomInput();
            var isThreat = predictor.Predict(randomInput);

            RaportTBox.Text += $"WindowMotion: {randomInput.WindowMotion}, DoorMotion:
{randomInput.DoorMotion}, " +
                               $"NoiseLevel: {randomInput.NoiseLevel}, SmokeLevel:
{randomInput.SmokeLevel}, Threat: {randomInput.Threat}" + "\r\n";
```

```
            if (isThreat == true)
            {
                RaportTBox.Text += "Спрацювала тривога\r\n";
            }

            RaportTBox.SelectionStart = RaportTBox.Text.Length;
            RaportTBox.ScrollToCaret();
        }
        catch (Exception ex)
        {
            RaportTBox.Text = ex.ToString();
        }
    }
}

public class SensorPredictor
{
    private readonly MLContext mlContext;
    private ITransformer model;

    public SensorPredictor(string modelPath)
    {
        mlContext = new MLContext();
        model = mlContext.Model.Load(modelPath, out var modelSchema);
    }

    public bool Predict(SensorInput input)
    {
        var predictionEngine = mlContext.Model.CreatePredictionEngine<SensorInput,
SensorOutput>(model);
        var result = predictionEngine.Predict(input);
        return Convert.ToBoolean(result.Prediction);
    }
}

public class SensorInputGenerator
{
    private static Random random = new Random();

    public SensorInput GenerateRandomInput()
    {
        return new SensorInput
        {
            WindowMotion = (float)random.NextDouble(),
            DoorMotion = (float)random.NextDouble(),
            NoiseLevel = (float)random.NextDouble(),
            SmokeLevel = (float)random.NextDouble(),
            Threat = random.Next(2) == 0
        };
    }
}
```

Лістинг 4. Код класу «ScenariosForm»

```
using SecurityApp.AppCode;
using SecurityApp.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SecurityApp.Forms.Dictionary
{
   public partial class ScenariosForm : Form
   {
      private int _selectedRowIndex = 0;
      private readonly ValidationMy _validation = new ValidationMy();
      private readonly ScenariosProvider _ScenariosProvider = new ScenariosProvider();
      private List<Scenarios> _ScenariosList = new List<Scenarios>();

      public ScenariosForm()
      {
         InitializeComponent();
         LoadData();
      }

      private void AddBtn_Click(object sender, EventArgs e)
      {
         if (IsDataEnteringCorrect())
         {
            _ScenariosProvider.InsertScenarios(ScenariosNameTBox.Text, DescriptionTBox.Text);
            DataLoad();
            ClearAllControls();
         }
      }

      private void ClearBtn_Click(object sender, EventArgs e)
      {
         ClearAllControls();
      }

      private void ExitBtn_Click(object sender, EventArgs e)
      {
         this.Close();
      }

      private void DataLoad()
      {
         int firstRowIndex = 0;
         if (ScenariosGridView.FirstDisplayedScrollingRowIndex > 0)
```

```
            {
                firstRowIndex = ScenariosGridView.FirstDisplayedScrollingRowIndex;
            }
            try
            {
                _ScenariosList = _ScenariosProvider.GetAllScenarios();
                LoadDataInScenariosGridView(_ScenariosList);
                if (_selectedRowIndex == ScenariosGridView.Rows.Count)
                {
                    _selectedRowIndex = ScenariosGridView.Rows.Count - 1;
                }
                if (_selectedRowIndex >= 0)
                {
                    ScenariosGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
                    ScenariosGridView.Rows[_selectedRowIndex].Selected = true;
                }
            }
            catch { }
        }

        private void LoadDataInScenariosGridView(List<Scenarios> ScenariosList)
        {
            ScenariosGridView.DataSource = null;
            ScenariosGridView.Columns.Clear();
            ScenariosGridView.AutoGenerateColumns = false;
            ScenariosGridView.RowHeadersVisible = false;

            ScenariosGridView.DataSource = ScenariosList;

            if (ScenariosList.Count > 0)
            {
                if (ScenariosList[0].Message == NamesMy.NoDataNames.NoDataInScenarios)
                {
                    DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
                    messageColumn.DataPropertyName = "Message";
                    messageColumn.Width = ScenariosGridView.Width -
NamesMy.SizeOptions.MinusSizePanel;
                    ScenariosGridView.Columns.Add(messageColumn);
                }
                else
                {
                    DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
                    DetailIdColumn.DataPropertyName = "ScenariosId";
                    ScenariosGridView.Columns.Add(DetailIdColumn);
                    ScenariosGridView.Columns[0].Visible = false;

                    DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
                    numberColumn.HeaderText = "№ ";
                    numberColumn.DataPropertyName = "Number";
                    numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
                    numberColumn.Width = NamesMy.SizeOptions.NumberSize;
```

```
                ScenariosGridView.Columns.Add(numberColumn);

                DataGridViewColumn ScenariosNameColumn = new
DataGridViewTextBoxColumn();
                ScenariosNameColumn.HeaderText = "Сценарій";
                ScenariosNameColumn.DataPropertyName = "ScenariosName";
                ScenariosNameColumn.Width = NamesMy.SizeOptions.NameSize;
                ScenariosGridView.Columns.Add(ScenariosNameColumn);
            }
            for (int i = 0; i < ScenariosGridView.Columns.Count; i++)
            {
                ScenariosGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
            }
        }
    }

    private void ClearAllControls()
    {
        ScenariosNameTBox.Text = String.Empty;
        DescriptionTBox.Text = String.Empty;
    }

    private bool IsDataEnteringCorrect()
    {
        bool isCorrect = true;
        if (_validation.IsDataEntering(ScenariosNameTBox.Text))
        {
            ScenariosNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        }
        else
        {
            ScenariosNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
            isCorrect = false;
        }
        return isCorrect;
    }

    private void ScenariosGridView_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        if (e.RowIndex >= 0 && ScenariosGridView[0, e.RowIndex].Value.ToString() !=
_ScenariosList[0].Message)
        {
            _selectedRowIndex = e.RowIndex;
            UpdateScenariosForm updateScenariosForm = new
UpdateScenariosForm(Convert.ToInt32(ScenariosGridView[0, e.RowIndex].Value.ToString()));
            updateScenariosForm.ShowDialog();
            DataLoad();
        }
    }
  }
}
```

Лістинг 5. Код класу «TeachingNeuralForm»

```
using Microsoft.ML;
using Microsoft.ML.Calibrators;
using Microsoft.ML.Data;
using Microsoft.ML.Trainers.FastTree;
using SecurityApp.AppCode;
using SecurityApp.Forms.Systems;
using SecurityApp.Providers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SecurityApp.Forms.Dictionary
{
   public partial class TeachingNeuralForm : Form
   {
      private string _Path = "";
      private MLContext context;

TransformerChain<BinaryPredictionTransformer<CalibratedModelParametersBase<FastTreeBinar
yModelParameters, PlattCalibrator>>> model;
      private IDataView data;

      private int _selectedRowIndex = 0;
      private ScenariosProvider _ScenariosProvider = new ScenariosProvider();
      private List<Scenarios> _ScenariosList = new List<Scenarios>();
      private ValidationMy _Validation = new ValidationMy();
      private NeuralProvider _NeuralProvider = new NeuralProvider();
      private List<Neural> _NeuralList = new List<Neural>();
      private LogsProvider _LogsProvider = new LogsProvider();
      private bool _IsModelTrain = false;

      public TeachingNeuralForm()
      {
         InitializeComponent();
         LoadAllData();
         DataLoad();
      }

      private void LoadAllData()
      {
         _ScenariosList = _ScenariosProvider.GetAllScenarios();
         ScenariosCBox.DataSource = _ScenariosList;
         ScenariosCBox.ValueMember = "ScenariosId";
         ScenariosCBox.DisplayMember = "ScenariosName";
```

```
        }

        private void DataLoad()
        {
            int firstRowIndex = 0;
            if (NeuralGridView.FirstDisplayedScrollingRowIndex > 0)
            {
                firstRowIndex = NeuralGridView.FirstDisplayedScrollingRowIndex;
            }
            try
            {
                _NeuralList = _NeuralProvider.GetAllNeural();
                LoadDataInNeuralGridView(_NeuralList);
                if (_selectedRowIndex == NeuralGridView.Rows.Count)
                {
                    _selectedRowIndex = NeuralGridView.Rows.Count - 1;
                }
                if (_selectedRowIndex >= 0)
                {
                    NeuralGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
                    NeuralGridView.Rows[_selectedRowIndex].Selected = true;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
        }

        private void LoadDataInNeuralGridView(List<Neural> NeuralList)
        {
            NeuralGridView.DataSource = null;
            NeuralGridView.Columns.Clear();
            NeuralGridView.AutoGenerateColumns = false;
            NeuralGridView.RowHeadersVisible = false;

            NeuralGridView.DataSource = NeuralList;

            if (NeuralList.Count > 0)
            {
                if (NeuralList[0].Message == NamesMy.NoDataNames.NoDataInNeural)
                {
                    DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
                    messageColumn.DataPropertyName = "Message";
                    messageColumn.Width = NeuralGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
                    NeuralGridView.Columns.Add(messageColumn);
                }
                else
                {
                    DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
                    DetailIdColumn.DataPropertyName = "NeuralId";
```

```
            NeuralGridView.Columns.Add(DetailIdColumn);
            NeuralGridView.Columns[0].Visible = false;

            DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
            numberColumn.HeaderText = "№ ";
            numberColumn.DataPropertyName = "Number";
            numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
            numberColumn.Width = NamesMy.SizeOptins.NumberSize;
            NeuralGridView.Columns.Add(numberColumn);

            DataGridViewColumn NeuralNamesColumn = new DataGridViewTextBoxColumn();
            NeuralNamesColumn.HeaderText = "Назва мережі";
            NeuralNamesColumn.DataPropertyName = "NeuralNames";
            NeuralNamesColumn.Width = 500;
            NeuralGridView.Columns.Add(NeuralNamesColumn);

            DataGridViewColumn NeuralFileModelColumn = new
DataGridViewTextBoxColumn();
            NeuralFileModelColumn.HeaderText = "Файл";
            NeuralFileModelColumn.DataPropertyName = "NeuralFileModel";
            NeuralFileModelColumn.Width = 300;
            NeuralGridView.Columns.Add(NeuralFileModelColumn);

            DataGridViewButtonColumn IsResidesBtn = new DataGridViewButtonColumn();
            IsResidesBtn.HeaderText = "Видалити";
            IsResidesBtn.Text = "Видалити";
            IsResidesBtn.UseColumnTextForButtonValue = true;
            IsResidesBtn.ToolTipText = "Видалити";
            IsResidesBtn.Width = NamesMy.SizeOptins.DeleteBtnSize;
            NeuralGridView.Columns.Add(IsResidesBtn);
        }
        for (int i = 0; i < NeuralGridView.Columns.Count; i++)
        {
            NeuralGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
        }
    }
}

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;
            context = new MLContext(seed: 0);

            List<SensorInput> datas = new List<SensorInput>();
            using (StreamReader reader = new StreamReader(_Path,
Encoding.GetEncoding(1251)))
            {
               reader.ReadLine(); // Пропускаємо перший рядок з іменами стовпців

               while (!reader.EndOfStream)
               {
                  var line = reader.ReadLine();
                  var values = line.Split(';');
                  SensorInput item = new SensorInput()
                  {
                     WindowMotion = float.Parse(values[0]),
                     DoorMotion = float.Parse(values[1]),
                     NoiseLevel = float.Parse(values[2]),
                     SmokeLevel = float.Parse(values[3]),
                     Threat = bool.Parse(values[4])
                  };
                  datas.Add(item);
               }
            }

            data = context.Data.LoadFromEnumerable<SensorInput>(datas);
            var trainTestData = context.Data.TrainTestSplit(data, testFraction: 0.2, seed: 0);
            var trainData = trainTestData.TrainSet;
            var testData = trainTestData.TestSet;

            // Define data process pipeline
            var pipeline = context.Transforms.Concatenate("Features",
nameof(SensorInput.WindowMotion), nameof(SensorInput.DoorMotion),
                                                nameof(SensorInput.NoiseLevel), nameof(SmokeLevel))
            .Append(context.Transforms.NormalizeMinMax("Features"))
            .Append(context.Transforms.CopyColumns(inputColumnName: "Threat",
outputColumnName: "Label"))
            .Append(context.BinaryClassification.Trainers.FastTree());

            RaportTBox.Text = "Training the model..." + "\r\n";

            // Навчання моделі
            model = pipeline.Fit(trainData);

            // Перевірка моделі
            var predictions = model.Transform(testData);
            var metrics = context.BinaryClassification.Evaluate(predictions);

            RaportTBox.Text += ($"Accuracy: {metrics.Accuracy:P2}") + "\r\n";
            RaportTBox.Text += ($"F1Score: {metrics.F1Score:P2}") + "\r\n";
            _IsModelTrain = true;
         }
```

```
            catch (Exception ex)
            {
                RaportTBox.Text = ex.Message;
            }
        }
    }

    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().Locatio
n);
            _NeuralProvider.InsertNeural(NeuralNamesTBox.Text,
Convert.ToInt32(ScenariosCBox.SelectedValue), pathName);
            context.Model.Save(model, data.Schema, localProj + pathName);
            ClearAllData();
            _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId, "Було навчено нейронну
мережу " +
              NeuralNamesTBox.Text, DateTime.Now);
            MessageBox.Show("Дані успішно збережено!");
        }
    }

    private void ClearBtn_Click(object sender, EventArgs e)
    {
        ClearAllData();
    }

    private void ExitBtn_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    public string GenerateFileName()
    {
        DateTime now = DateTime.Now;
        string fileName = string.Format("{0}_{1}_{2}_{3}_{4}_{5}",
            now.Year, now.Month, now.Day, now.Hour, now.Minute, now.Second);

        return fileName;
    }

    private void ClearAllData()
    {
        _IsModelTrain = false;
        FileNameTBox.Text = String.Empty;
        NeuralNamesTBox.Text = String.Empty;
        RaportTBox.Text = String.Empty;
```

```
            DataLoad();
        }

        private bool IsDataEnteringCorrect()
        {
            bool isCorrect = true;
            if (!_IsModelTrain)
            {
                MessageBox.Show("Неможливо зберегти дані. \r\nЩе не навчено нейронну
мережу!", "Увага!");
                isCorrect = false;
            }
            if (Convert.ToInt32(ScenariosCBox.SelectedValue) <= 0)
            {
                ScenariosValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
                isCorrect = false;
            }
            else
            {
                ScenariosValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
            }

            if (_Validation.IsDataEntering(NeuralNamesTBox.Text))
            {
                NeuralNamesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
                isCorrect = false;
            }
            else
            {
                NeuralNamesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
            }

            return isCorrect;
        }

        // Додатковий код для класів SensorInput і SensorOutput...
    }

    public class SensorInput
    {
        [LoadColumn(0), ColumnName("WindowMotion")]
        public float WindowMotion;

        [LoadColumn(1), ColumnName("DoorMotion")]
        public float DoorMotion;

        [LoadColumn(2), ColumnName("NoiseLevel")]
        public float NoiseLevel;

        [LoadColumn(3), ColumnName("SmokeLevel")]
        public float SmokeLevel;
```

```
    [LoadColumn(4), ColumnName("Threat")]
    public bool Threat;
  }

  public class SensorOutput
  {
    [ColumnName("PredictedLabel")]
    public bool Prediction;

    [ColumnName("Probability")]
    public float Probability;

    [LoadColumn(4), ColumnName("Threat")]
    public bool Threat;
  }

  public class SensorDataGenerator
  {
    private static Random random = new Random();

    public List<SensorInput> GenerateData(int numberOfSamples)
    {
      var data = new List<SensorInput>();

      for (int i = 0; i < numberOfSamples; i++)
      {
        var sensorInput = new SensorInput
        {
          WindowMotion = (float)random.NextDouble(),
          DoorMotion = (float)random.NextDouble(),
          NoiseLevel = (float)random.NextDouble(),
          SmokeLevel = (float)random.NextDouble(),
          Threat = GenerateThreat()
        };

        data.Add(sensorInput);
      }

      return data;
    }
  }
}
```