НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
Завідувач кафедри
_____ **Дмитро ЛАНДЕ**
(підпис)

«_____»_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека»

на тему: Метод виявлення вторгнень у розумні будинки на основі аналізу поведінкових патернів

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи ФБ-05
Нікітський Іван Володимирович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник к. ф-м. н., доцент, Смирнов С. А. _____ _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент ас. каф. ММАД НН ФТІ Крохальов І. Д. _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ – 2024 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Дмитро ЛАНДЕ
(підпис)

«___»_____2024 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

Нікітському Івану Володимировичу

1. Тема роботи: Метод виявлення вторгнень у розумні будинки на основі аналізу поведінкових патернів, керівник роботи к. ф-м. н., доцент, Смирнов С. А., затверджені наказом по університету від «31» травня 2024 р. №2251-с

2. Термін подання здобувачем вищої освіти роботи 14 червня 2024 р.

3. Вихідні дані до роботи: Тенденції розвитку систем розумного будинку, методи виявлення вторгнень на основі аналізу поведінкових патернів, алгоритми машинного навчання (БФГШ, МОВ, МВЛ), інструменти збору даних з IoT, інтеграція AIoT для обробки даних з сенсорів.

4. Зміст роботи: дослідження концепцій та архітектур розумних будинків, аналіз ключових компонентів систем IoT, вивчення поведінкових патернів мешканців, огляд існуючих методів виявлення вторгнень. Розробка математичної моделі та алгоритмів для виявлення аномалій у поведінці, вибір технологічних рішень, реалізація прототипу системи. Експериментальне дослідження методу, верифікація моделі через різні сценарії, аналіз перспектив подальших досліджень.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) Презентація до захисту дипломної роботи.

6. Дата видачі завдання 22.09.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Визначення теми ДР	23.09.2023	виконано
2	Огляд літератури та аналіз концепцій	23.02.2023 - 10.03.2023	виконано
3	Дослідження існуючих методів виявлення вторгнень	11.03.2023 - 31.03.2023	виконано
4	Вибір методу машинного навчання та формулювання завдання	01.04.2023 - 07.04.2023	виконано
5	Розробка математичної моделі, алгоритмів та прототипу	08.04.2023 - 20.04.2023	виконано
6	Налаштування середовища, збір даних та верифікація методу	21.04.2023 - 05.05.2023	виконано
7	Експериментальне дослідження та оцінка ефективності методу	06.05.2023 - 12.05.2023	виконано
8	Оформлення роботи	13.05.2023 - 31.05.2024	виконано

Здобувач вищої освіти _____ Іван Нікітський
(підпис)

Керівник роботи _____ Сергій Смирнов
(підпис)

РЕФЕРАТ

Дипломна робота обсягом 107 сторінок включає 40 ілюстрацій, 10 таблиць, 2 додатки та 34 бібліографічних найменувань. Метою роботи є розробка методу виявлення вторгнень у розумні будинки на основі аналізу поведінкових патернів. Використано методи машинного навчання, зокрема метод Бройдена-Флетчера-Гольдфарба-Шанно, для аналізу поведінкових даних, зібраних за допомогою різноманітних сенсорів.

У процесі роботи було розглянуто концепції розумних будинків, архітектуру та ключові компоненти систем, а також методи виявлення вторгнень. Розроблено математичну модель виявлення вторгнень, алгоритми для навчання та зберігання даних моделі, а також прототип системи з трьохрівневою архітектурою. Вибір технологічних рішень, зокрема мови програмування C#, забезпечив оптимальну реалізацію методу.

Проведено експериментальне дослідження та оцінку ефективності методу на основі зібраних даних з платформи Kaggle. Отримані метрики свідчать про високу ефективність моделі: логарифмічні втрати – 0,65, макро точність – 0,81, мікро точність – 0,81, редукція логарифмічних втрат – 0,7. Верифікація моделі через різні сценарії виявлення вторгнень показала, що модель успішно ідентифікує підозрілу діяльність та визначає можливі вторгнення.

Результати дослідження рекомендовано для впровадження в системи розумних будинків з метою підвищення рівня безпеки. Подальші дослідження можуть включати інтеграцію з додатковими датчиками, розробку гібридних моделей штучного інтелекту, адаптивне навчання та забезпечення приватності та безпеки даних.

Ключові слова: безпека даних, БФГШ, виявлення вторгнень, машинне навчання, поведінкові патерни, розумний будинок, сенсори, трьохрівнева архітектура, C#, Kaggle.

ABSTRACT

The diploma thesis consists of 107 pages, including 40 illustrations, 10 tables, 2 appendices, and 34 bibliographic references. The aim of the thesis is to develop a method for detecting intrusions in smart homes based on the analysis of behavioral patterns. Machine learning methods, particularly the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, were used to analyze behavioral data collected from various sensors.

The study examined the concepts of smart homes, the architecture and key components of systems, as well as intrusion detection methods. A mathematical model for intrusion detection was developed, along with algorithms for training and storing the model's data, and a prototype system with a three-tier architecture. The choice of technological solutions, including the C# programming language, ensured the optimal implementation of the method.

An experimental study and evaluation of the method's effectiveness were conducted based on data collected from the Kaggle platform. The obtained metrics indicate high model efficiency: logarithmic loss – 0.65, macro precision – 0.81, micro precision – 0.81, logarithmic loss reduction – 0.7. Verification of the model through various intrusion detection scenarios demonstrated that the model successfully identifies suspicious activity and detects potential intrusions.

The results of the study are recommended for implementation in smart home systems to enhance security levels. Future research may include integration with additional sensors, development of hybrid artificial intelligence models, adaptive learning, and ensuring data privacy and security.

Keywords: data security, BFGS, intrusion detection, machine learning, behavioral patterns, smart home, sensors, three-tier architecture, C#, Kaggle.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БФГШ - Бройден-Флетчер-Гольдфарб-Шанно

МОВ - метод опорних векторів

МВЛ - метод випадкового лісу

IoT - Інтернет речей (Internet of Things)

RL-IoT - Архітектура реінфорсмент-навчання для Інтернету речей

AIoT - Архітектура штучного інтелекту для Інтернету речей

DHT22 - цифровий датчик температури та вологості

PIR - пасивний інфрачервоний датчик (Passive Infrared Sensor)

KY-018 - фоторезистор

MAX9814 - мікрофонний модуль

ВСТУП

У сучасному світі, де технології розумного дому все більше входять в наше повсякденне життя, забезпечення безпеки цих систем стає надзвичайно важливим завданням. Розумні домівки, оснащені безліччю IoT-пристроїв, збирають величезні обсяги даних про поведінку та звички своїх мешканців. Ці дані, при неналежному захисті, можуть стати мішенню для кіберзлочинців. Виявлення вторгнень у ці системи не тільки захищає особисту інформацію, але й запобігає потенційним загрозам фізичній безпеці осіб.

Зростаюча кількість кібератак на розумні домівки змушує науковців та розробників шукати нові підходи до виявлення та запобігання таких інцидентів. Традиційні методи кібербезпеки часто виявляються неефективними у випадку IoT, оскільки вони не враховують специфіку і взаємодію множини пристроїв і сервісів в рамках єдиної екосистеми. Це створює потребу в розробці нових методів, які змогли б ефективно адаптуватися до динамічного середовища розумних домівок.

Підхід, заснований на машинному навчанні, дозволяє використовувати зібрані дані для створення моделей, що можуть самостійно навчатися виявляти аномальні або непередбачені поведінкові патерни, які можуть вказувати на спроби вторгнення. Це не тільки підвищує точність виявлення, але й забезпечує швидшу реакцію на потенційні загрози. Важливість розвитку таких технологій для України полягає в зміцненні національної безпеки та підвищенні рівня довіри до технологій розумного дому серед споживачів.

Впровадження інноваційних методів захисту для розумних домівок є значущим не тільки з наукової точки зору, але й має велике практичне значення. Поліпшення систем безпеки відіграє критичну роль у захисті кінцевих користувачів і може сприяти розширенню застосування цих технологій на території країни, що, в свою чергу, позитивно вплине на економічний розвиток і технологічне вдосконалення.

Метою роботи є розробка методу виявлення вторгнень у розумні будинки на основі аналізу поведінкових патернів.

Для досягнення цієї мети потрібно виконати наступні задачі:

– провести теоретичний аналіз існуючих рішень та методів виявлення вторгнень в розумних домівках, включаючи вивчення архітектури систем «Розумний дім» та ролі поведінкових патернів у контексті забезпечення безпеки;

– розробити і валідувати математичну модель, що дозволить з високою точністю ідентифікувати аномальні поведінкові патерни, характерні для спроб вторгнення, з використанням вибраного методу машинного навчання;

– створити прототип системи для виявлення вторгнення, що включає вибір та інтеграцію технологічних рішень і інструментів, а також розробку алгоритмів для обробки та аналізу поведінкових даних;

– провести експериментальне дослідження з метою оцінки ефективності розробленого методу через реалізацію різних сценаріїв вторгнення та збір та аналіз відповідних даних для верифікації методу.

Об'єктом дослідження є процес забезпечення безпеки в системах розумних домівок.

Предметом дослідження є розробка методу машинного навчання для ідентифікації потенційних вторгнень на основі цих поведінкових патернів.

У процесі дослідження було використано наступний ряд методів:

– метод теоретичного аналізу літературних джерел. Застосований для всебічного огляду існуючих концепцій розумних домівок, архітектур та ключових компонентів систем;

– аналітичний метод. Використовувався для аналізу поведінкових паттернів та визначення аномальної поведінки на основі зібраних даних;

– метод машинного навчання. Використаний для розробки та тренування моделей, що можуть ідентифікувати аномалії в поведінці користувачів розумних домівок;

– експериментальний метод. Застосовувався для верифікації та оцінки ефективності розробленої системи шляхом проведення тестувань в контрольованих умовах.

Наукова новизна одержаних результатів полягає в розробці та впровадженні методики виявлення кібервторгнень у системах розумних домівок з використанням методу квазіньютона. Зокрема, розроблено модель, яка інтегрує аналіз поведінкових патернів мешканців з даними з різноманітних датчиків розумного дому, дозволяючи не тільки ідентифікувати поточні загрози, але й прогнозувати потенційні вторгнення.

Практичне значення одержаних результатів дослідження полягає в розробці та впровадженні ефективного інструменту для підвищення рівня безпеки в системах розумних домівок, що вже демонструє свою готовність до використання та масштабування. Розроблена система дозволяє не тільки виявляти існуючі загрози, але й прогнозувати потенційні вторгнення, надаючи можливість заздалегідь адаптувати захисні механізми системи. Окрім того, результати дослідження мають велике значення для подальшого розвитку сфери кібербезпеки в контексті IoT та можуть бути використані у академічних курсах і наукових роботах, що вивчають проблематику безпеки інтелектуальних будинкових систем.

1 ТЕОРЕТИЧНИЙ ОГЛЯД І ФОРМУЛЮВАННЯ ПРОБЛЕМИ

1.1 Концепції розумних будинків

Концепція «Розумного Дому» представляє собою прогресивне втілення ідеї автоматизованого управління житловим простором, опираючись на передові розробки у сферах інженерії та інформаційних технологій (рис. 1.1). Система розумного дому об'єднує в собі різні пристрої, починаючи від елементарних температурних сенсорів до складних мультимедійних систем, що сприяє синергії технічних компонентів, підвищуючи комфортність, безпеку та енергетичну ефективність житла.

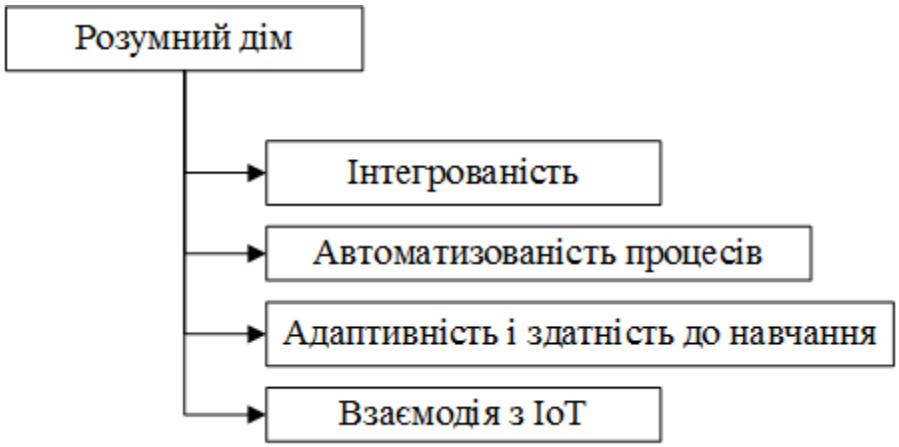


Рисунок 1.1 – Приклад концепції «Розумний Дім»

Фундаментальним принципом розумного дому є його здатність до максимальної автономії: система сама регулює енергопостачання, клімат-контроль, освітлення, безпеку, та інші домогосподарські функції. Таке управління забезпечується завдяки комплексним алгоритмам, які аналізують актуальні умови і попередній досвід взаємодії з мешканцями, прогнозуючи їхні потреби і відповідно адаптуючи роботу системи [1].

«Розумний Дім» не лише оптимізує побут, але й значно покращує якість життя мешканців, забезпечуючи раціональне управління ресурсами, що дозволяє знижувати енергоспоживання та оптимізувати витрати. Ця інноваційна система

відображає еволюцію традиційного житла, адаптуючись до сучасних вимог екологічності, економічності та індивідуального підходу до кожного мешканця.

Інтеграція технологій штучного інтелекту, зокрема мереж на базі машинного навчання, значно розширює можливості системи. Завдяки різноманітним алгоритмам, які здатні розпізнавати складні взірці у масивах даних, система може передбачати потенційні інциденти та заздалегідь реагувати на них. Це дозволяє системі вчитися на поведінкових моделях мешканців і адаптувати налаштування безпеки, що забезпечує високий рівень захисту, враховуючи індивідуальні особливості кожного користувача та зміни в їхній поведінці [2].

Архітектура «розумного будинку» може бути втілена через два основних підходи: централізований та децентралізований. У централізованій моделі, всі ключові компоненти системи, включно з керувальними механізмами та центральним процесором, згруповані в єдину телекомунікаційну мережу (рис. 1.2). Це забезпечує ефективне та синхронізоване управління, а також безперервну передачу сигналів по всьому об'єкту.

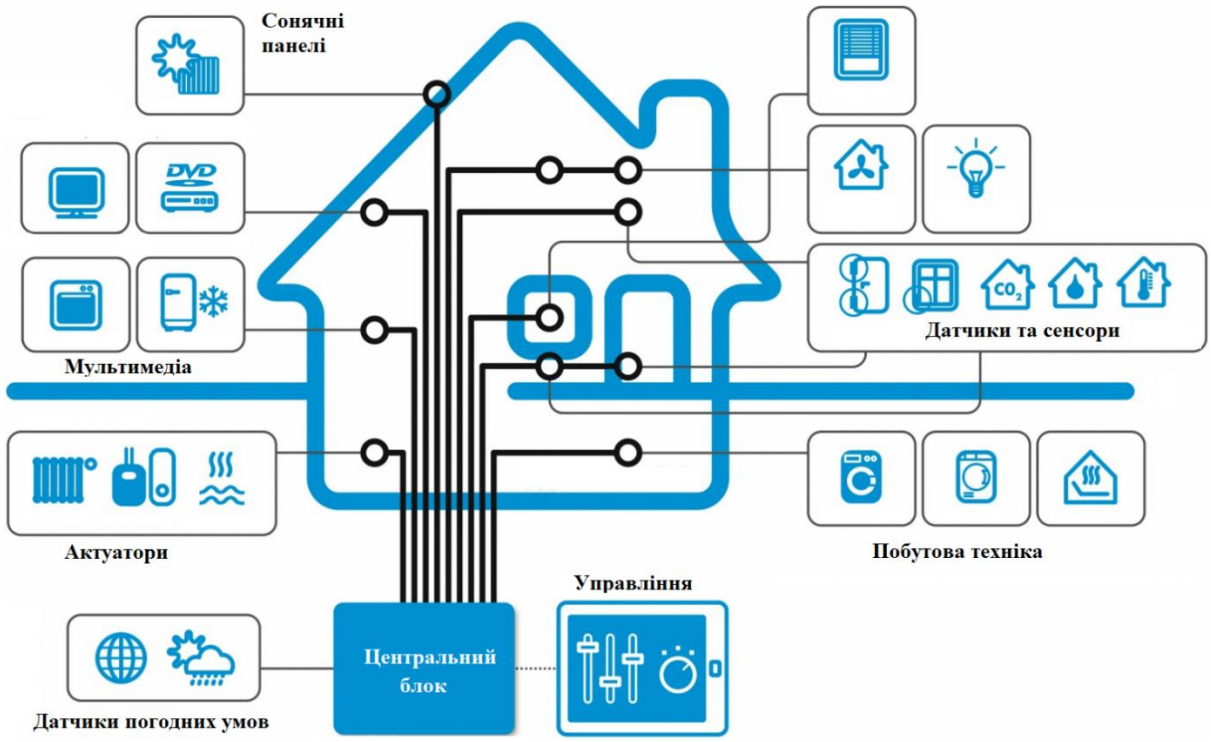


Рисунок 1.2 – Приклад централізованої архітектури «Розумний Дім»

Інтерфейси управління служать зв'язком між користувачами та системою, дозволяючи керувати функціями розумного будинку через різноманітні пристрої – від традиційних пультів до інноваційних сенсорних панелей і мобільних додатків. Система також автоматично відгукується на зміни у середовищі за допомогою вбудованих датчиків, які моніторять різні умови, включаючи освітленість, наявність людей, температуру та вологість.

Центральний обчислювальний контролер відіграє роль командного центру системи, керуючи збором, аналізом та обробкою даних, які надходять як від керувальних інтерфейсів, так і від датчиків. Він зберігає задані користувачем сценарії та генерує команди засновані на алгоритмах штучного інтелекту, координуючи діяльність усіх підсистем розумного будинку для виконання завдань [3].

Централізована архітектура розумного будинку має низку переваг, які варто врахувати при виборі системи автоматизації, включно з:

– уніфікацією управління, що дозволяє інтегрувати всі підсистеми в єдиний інтерфейс, спрощуючи управління;

– ефективною координацією роботи системи, забезпечуючи синхронізовані дії;

– легкістю масштабування через додавання нових пристроїв або оновлення компонентів через один центральний вузол;

– зручністю моніторингу, оскільки стан усіх компонентів можна відслідковувати з одного місця;

– спрощенням технічної підтримки, полегшуючи діагностику та усунення несправностей.

В децентралізованій архітектурі інтелектуальних систем управління житлом концепція централізованого контролю замінюється на розподілену модель управління, яка надає кожному компоненту, включно з окремими пристроями та модулями, можливість виконувати свої функції незалежно від інших (рис. 1.3). Кожен елемент обладнано власною обчислювальною логікою, що дозволяє їм

самостійно реагувати на зміни у домашньому середовищі. Так, датчики, реагуючи на різні події або зміни, можуть автономно активувати виконавчі механізми, як-от реле чи електромеханічні системи, без потреби центрального втручання. Ця модель забезпечує більшу гнучкість у відповідях на локальні події та може значно покращувати час реакції системи на потреби користувачів.

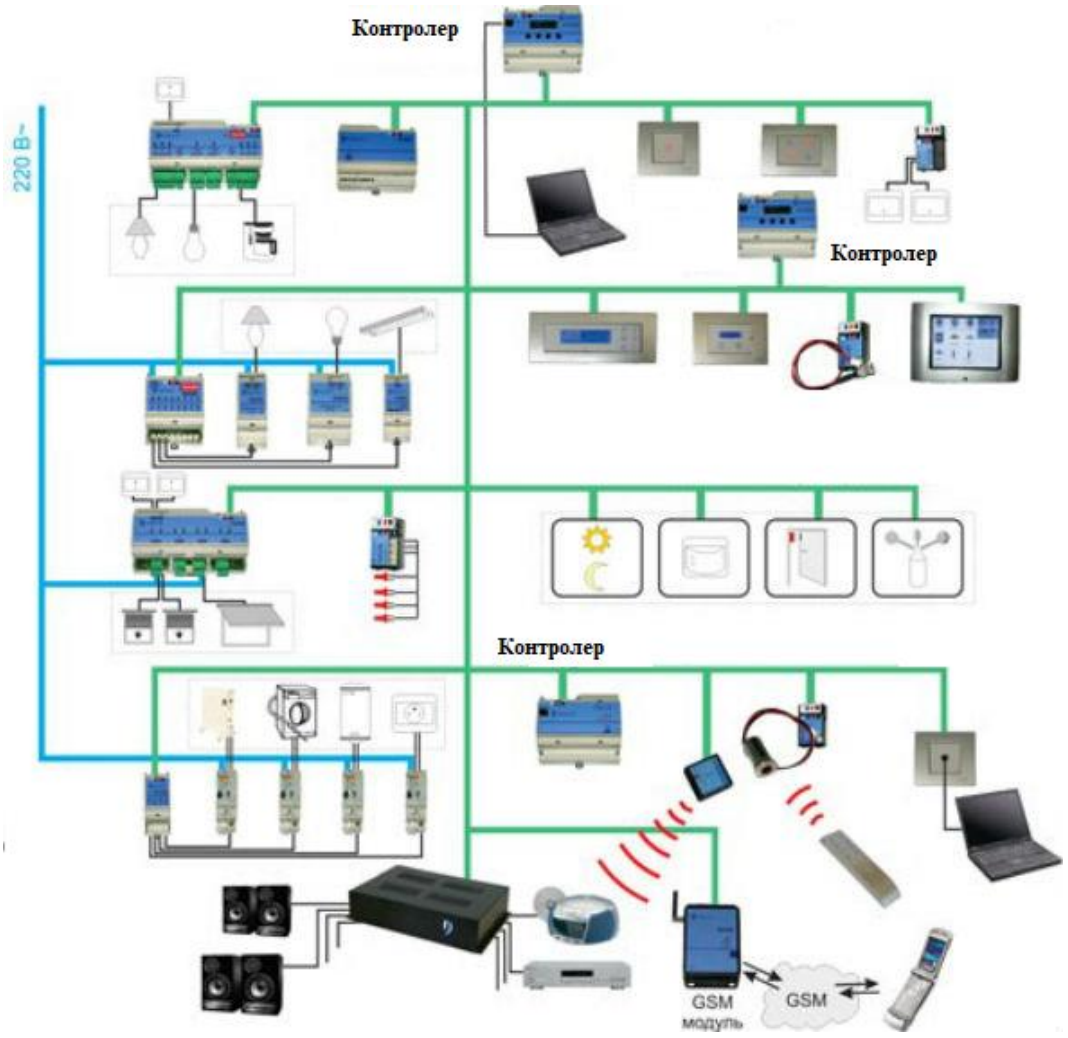


Рисунок 1.3 – Приклад децентралізованої архітектури

Ця архітектура надає системі здатність до самостійної регуляції на локальному рівні, дозволяючи окремим компонентам самостійно адаптуватися до змін умов довкілля. Така властивість підвищує гнучкість та адаптивність системи, дозволяючи їй відповідати на індивідуальні потреби користувачів. Водночас, необхідність координації між розподіленими компонентами може ускладнити управління та моніторинг загальної системи [4]. Ця конфігурація може вимагати

розвинутішої інфраструктури для забезпечення ефективної взаємодії, а також створює додаткові виклики для централізованої діагностики та обслуговування.

Таким чином, хоча децентралізована архітектура забезпечує компонентам більшу незалежність, вона також вимагає більш складних технічних рішень для їх інтеграції та управління. Ця структура може бути особливо корисною в ситуаціях, де важлива швидка реакція на місцеві зміни та висока автономія окремих зон або пристроїв. Однак, у великих та комплексних системах, де необхідний загальний моніторинг і управління, потрібно обережно оцінювати потенційні складнощі, які може виникнути від використання децентралізованої моделі.

1.2 Архітектура та ключові компоненти систем «Розумний дім»

Архітектурний дизайн інтелектуальних систем використовує принципи штучного інтелекту і охоплює взаємопов'язані компоненти, які включають IoT пристрої, сенсори, актуатори та розширені мережеві технології (рис. 1.4). Ці елементи з'єднані з потужними обчислювальними платформами, які застосовують алгоритми штучного інтелекту для аналізу даних та автоматизації управлінських процесів.

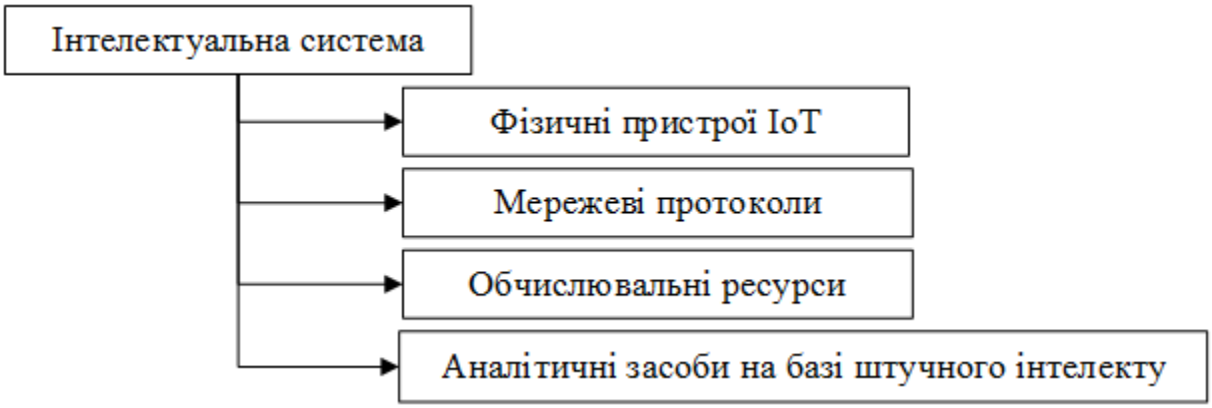


Рисунок 1.4 – Основні компоненти інтелектуальних систем

Як показано на рис. 1.4, ключовими елементами архітектури інтелектуальних систем є:

– фізичні пристрої IoT, які відслідковують зміни у навколишньому середовищі через різноманітні датчики, надаючи системі інформацію про фізичний стан;

– мережеві протоколи, які підтримують як дротові, так і бездротові зв'язки між IoT пристроями та обчислювальними центрами, включаючи Wi-Fi, Bluetooth, LTE, що дозволяють передавати дані;

– обчислювальні ресурси, які можуть бути розміщені локально на edge серверах або в хмарних середовищах, забезпечуючи значні обчислювальні можливості для складного аналізу даних;

– аналітичні інструменти на базі штучного інтелекту, включаючи техніки машинного та глибокого навчання, які дозволяють системі виявляти закономірності, аналізувати тренди, прогнозувати події та автоматично налаштовувати стратегії управління для адаптації до змін.

Ця архітектура підвищує інтерактивність та адаптивність просторів, дозволяючи системі автономно реагувати на користувацькі потреби та зміни у середовищі, що сприяє більш ефективному управлінню та рівню автоматизації в житлових та комерційних зонах [5].

Сучасна архітектура інтелектуальних систем створює комплексний екосистемний підхід, інтегруючи IoT пристрої, складні мережеві структури та великі обчислювальні потужності для оптимізації та автоматизації управлінських процесів. Зображено на рис. 1.5, архітектура RL-IoT демонструє, як за допомогою алгоритмів навчання з підкріпленням, система налаштовує IoT пристрої до досягнення конкретних цілей через виконання визначених дій.

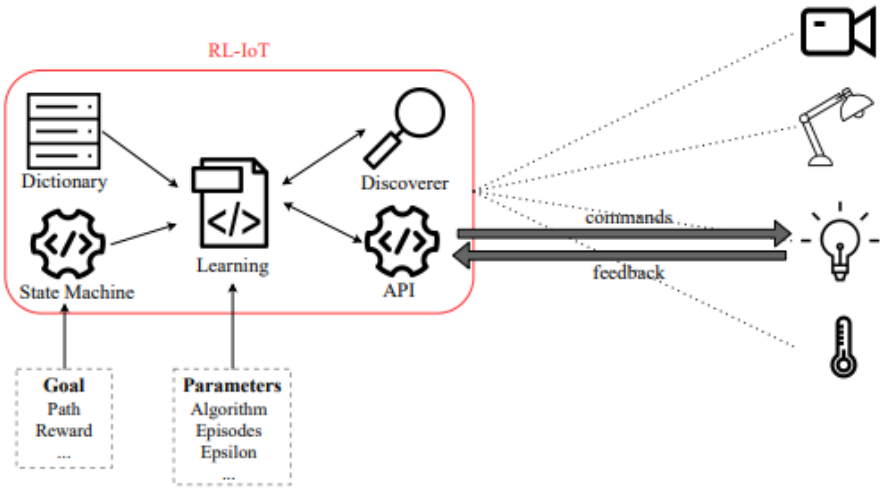


Рисунок 1.5 – Архітектура RL-IoT

В системі RL-IoT, взаємодія між пристроями відбувається через детально розроблений набір комунікаційних інструкцій, що включає в себе повний спектр команд, який може бути сформований через офіційні протокольні специфікації чи реверс-інжиніринг. Алгоритми навчання з підкріпленням, такі як Q-Learning чи SARSA, дозволяють системі оптимізувати свої дії відповідно до системи винагород, спрямовуючи її до досягнення визначеної мети [6].

Модулі, як «Discoverer», ідентифікують IoT пристрої в мережі за допомогою сканування. Після їх виявлення, модуль Socket API надає абстракцію для взаємодії з пристроями, дозволяючи не лише відправляти, але й приймати команди відповідно до їхніх можливостей приймати запити [7].

Інтернет речей (IoT) та автономні системи, які спочатку розвивалися як окремі технологічні сфери, зливаються в концепцію AIoT (Автономний Інтернет речей). Це створює новий рівень інтеграції, значно розширюючи можливості IoT. В архітектурі AIoT, пристрої активно взаємодіють з фізичним світом через сенсори та виконавчі механізми, збираючи дані та реалізуючи рішення на їх основі (рис. 1.6) [8].

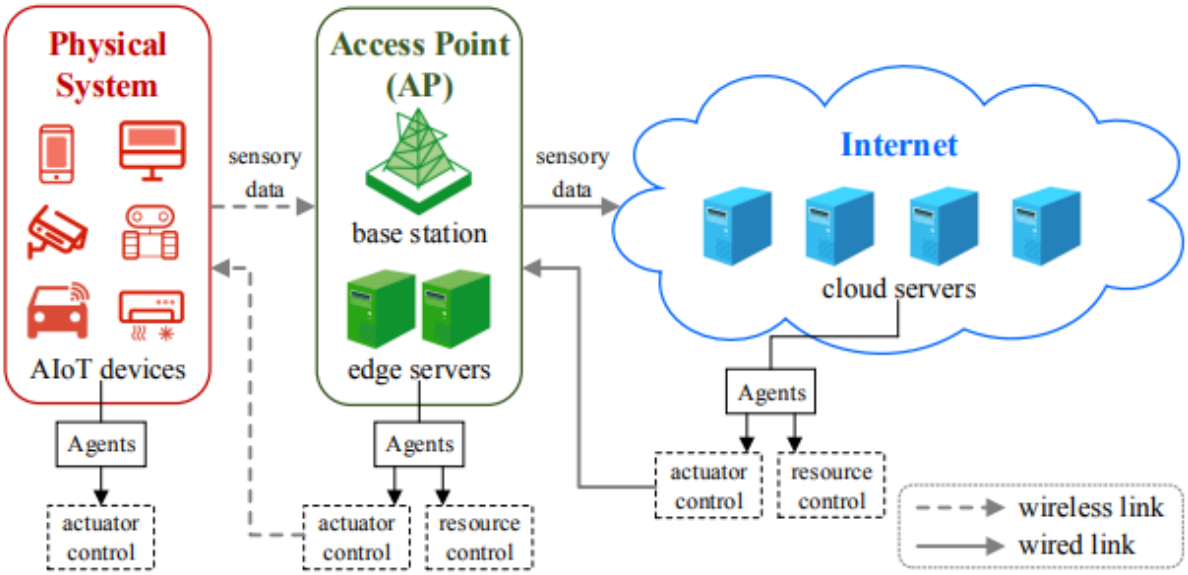


Рисунок 1.6 – Архітектура AIoT

WSAN (Wireless Sensor and Actuator Network) у складі AIoT включає сенсори для моніторингу середовища та виконавчі пристрої, які реагують на аналіз цих даних через бездротові мережі. Застосування алгоритмів глибокого навчання дозволяє автономним агентам WSAN обробляти інформацію та керувати виконавчими пристроями, адаптуючи або змінюючи стан середовища відповідно до змін, що створює динамічну і реактивну систему.

В архітектурі AIoT користувачі мають справу з комплексною системою, яка об'єднує збір, аналіз, обробку та виконання даних, перевищуючи традиційні можливості WSAN. AIoT інтегрує передові комунікаційні технології та обчислювальні можливості для комплексної обробки інформації, включаючи різноманітні сервіси [9].

Структура AIoT включає три ключові рівні:

– шар сприйняття, який є первинним інтерфейсом між фізичним світом та цифровою системою, включаючи IoT пристрої з сенсорами для збору даних та актуаторами для взаємодії з оточенням;

– мережевий шар, що забезпечує зв'язок між IoT пристроями та обчислювальними центрами. Цей шар включає різні точки доступу, такі як мобільні базові станції чи Wi-Fi роутери, необхідні для передачі даних;

– шар додатків, який охоплює сервери та обчислювальні платформи, де відбувається аналіз та обробка даних і де за допомогою алгоритмів штучного інтелекту реалізується автоматизація процесів [10].

1.3 Поняття поведінкових паттернів у контексті розумних домівок

У контексті розумних домівок, поняття «поведінкових паттернів» описує систематичні звички та рутини, які мешканці демонструють у своєму повсякденному житті. Ці звички формуються на основі повторюваних дій і активностей, які здійснюються в певний час та у певних умовах. Розумні домівки використовують цю інформацію для автоматизації процесів, оптимізації ресурсів та підвищення комфорту мешканців.

У рамках розумних домівок, поведінка визначається як послідовність дій або активностей, які виконуються користувачами у домашньому середовищі. Ці дії можуть бути відстежені, зареєстровані та аналізовані через різні сенсори та інтелектуальні системи вбудовані у домашній екосистемі. Прикладами таких дій можуть бути:

– використання побутових приладів. Час і тривалість використання кухонних приладів, як-от кавомашини або мікрохвильової печі, можуть бути залежні від режиму дня мешканців. Розумна домівка може відстежувати і запам'ятовувати ці звички, автоматично включаючи або вимикаючи прилади в заздалегідь визначені часи;

– пересування по дому. Системи розумного дому можуть фіксувати, в яких кімнатах і в які часи доби мешканці перебувають найчастіше. Наприклад, якщо кожного ранку мешканці переходять із спальні в кухню, система може автоматично включати світло та опалення на цьому маршруті;

– зміни у поведінці відповідно до зовнішніх умов. Наявність сенсорів, що фіксують погодні умови, дозволяє системі адаптувати внутрішні умови, як-от

температуру або освітленість, відповідно до зовнішніх змін, забезпечуючи таким чином найкращий рівень комфорту [11].

Інтелектуальні системи, які вивчають звички та повсякденні моделі поведінки мешканців, можуть виявляти незвичайні або аномальні активності, які можуть свідчити про потенційні загрози або несанкціоновані дії. Наприклад, якщо система помічає відкриття вхідних дверей або активацію систем у нехарактерний час, коли мешканці зазвичай відсутні або сплять, вона може автоматично сповістити власників або навіть звернутися до служби охорони.

Також, за допомогою інтеграції камер спостереження, сенсорів руху та інших детекторів, розумна домівка може аналізувати поведінку відвідувачів. Якщо поведінка відвідувача не відповідає типовим взаємодіям, які система класифікує як безпечні, наприклад, якщо гість зупиняється біля особливо чутливих або цінних областей будинку на незвично довгий час, система може вжити заходів, попереджаючи власників або змінюючи режими безпеки.

Таке використання поведінкових даних не лише підвищує ефективність систем безпеки через автоматизовану реакцію на потенційні загрози, але й забезпечує спокій та комфорт для мешканців, знаючи, що їхнє житло активно адаптується для забезпечення їхньої безпеки в реальному часі. Таким чином, інтелектуальні домівки стають не просто місцем проживання, але й активним учасником у забезпеченні добробуту та безпеки своїх мешканців [12].

Поведінкові патерни можуть бути інтегровані з іншими системами дому для створення більш зв'язаного та автономного домашнього середовища. Це включає інтеграцію з системами розпізнавання голосу, автоматичним управлінням мультимедійним контентом, а також персоналізованими охоронними системами.

1.4 Аналіз існуючих методів виявлення вторгнень

Системи безпеки сьогодні включають різноманітні інтелектуальні функції, які здатні адаптуватися до повсякденних потреб та звичок мешканців. Штучний

інтелект, особливо машинне навчання та глибоке навчання, розширює можливості для підвищення безпеки, дозволяючи системам не лише відповідати на загрози, але й передбачати їх. Аналіз застосування методів штучного інтелекту у сфері «розумних будинків» допомагає усвідомити, як інтелектуальні системи можуть захищати житло та сприяти створенню більш безпечного та комфортного життєвого простору.

Методи розумного відеоспостереження

Розумне відеоспостереження трансформує системи безпеки в розумних домівках, використовуючи передові алгоритми для аналізу відео. Це дозволяє не просто фіксувати рух, але й розпізнавати людей, автомобільні номери та інші важливі елементи. Завдяки інтеграції з штучним інтелектом, ці системи вчаться на основі попередніх інцидентів, що підвищує їхню здатність прогнозувати та проактивно реагувати на загрози.

З використанням технік глибокого навчання, розумне відеоспостереження ефективно аналізує масивні обсяги відеоданих, виявляючи складні поведінкові шаблони. Це включає в себе відстеження незвичайних дій, таких як несанкціонований доступ або незвичайні поведінкові відхилення, та відправлення автоматичних сповіщень про такі події.

Інтеграція розумного відеоспостереження з домашніми автоматизованими системами створює можливість для сценаріїв, де камери взаємодіють із системами освітлення, замками та аудіосистемами, посилюючи відчуття безпеки та присутності. Така технологія не лише захищає, але й забезпечує зручності, розпізнаючи мешканців та вітаючи їх при вході, або інформуючи про прибуття гостей.

Однією з основних переваг розумного відеоспостереження є його здатність до самонавчання. Системи безпеки, які постійно оптимізують свої реакції на зібрані дані, з часом стають тільки ефективнішими. Вони не тільки виявляють потенційні загрози, але й адаптують свою поведінку для забезпечення максимального захисту домівок та їх мешканців [13].

Системи розумного освітлення

Системи розумного освітлення є ключовою складовою автоматизації сучасного дому, значно підвищуючи рівень безпеки та комфорту мешканців. Вони автоматично адаптують освітлення відповідно до природного освітлення, часу доби та наявності людей у приміщенні. Системи можна налаштовувати так, щоб активувати світло при детекції руху, що не тільки зручно, але й служить як додатковий захід безпеки, відлякуючи потенційних зловмисників.

Інтегрування розумного освітлення з системами безпеки дозволяє освітленню включатися в режимі тривоги або на відповідь до безпекових повідомлень, покращуючи якість відеоспостереження чи відлякуючи непроханих гостей. Це не тільки забезпечує вищий рівень захисту, але й може імітувати присутність людей у домівці під час їхньої відсутності.

Додатково, розумне освітлення можна інтегрувати з голосовими помічниками для більшої зручності, а використання енергоефективних світлодіодних ламп допомагає знизити споживання електроенергії, що сприяє екологічній стійкості дому [14].

Системи інтелектуальних замків безпеки

Інтелектуальні замки безпеки «розумного будинку» надають власникам безключовий доступ та ряд передових функцій для забезпечення безпеки. Вони використовують біометричні технології, такі як сканування відбитків пальців, розпізнавання обличчя або сітківки ока, для точної ідентифікації осіб, що забезпечує захист від несанкціонованого доступу. Ці замки також можуть інтегруватися з домашніми автоматизаційними системами та управлятися через мобільні додатки, дозволяючи власникам контролювати доступ до їхнього дому з будь-якого місця.

Мобільні додатки, які пов'язані з інтелектуальними замками, відкривають широкі можливості для керування доступом. Власники можуть створювати тимчасові або постійні електронні ключі для гостей або обслуговуючого персоналу, які легко скасовувати або змінювати, що додає гнучкості та контролю. Інтелектуальні замки також забезпечують функцію журналу доступу, фіксуючи

кожну спробу входу або виходу, що дозволяє власникам відстежувати активність біля їхньої власності.

Інтеграція з іншими системами безпеки, такими як відеокамери або системи розумного освітлення, дозволяє створювати комплексні сценарії безпеки, наприклад, імітацію присутності людей в будинку під час їхньої відсутності. Це підвищує рівень захищеності та комфорту для мешканців.

Технології шифрування захищають бездротове з'єднання між замком і мобільними пристроями, забезпечуючи конфіденційність переданих даних. Додатково, інтелектуальні замки можуть підтримувати двофакторну автентифікацію, що вимагає додаткового підтвердження для доступу, наприклад, введення коду доступу або підтвердження через мобільний пристрій.

Інтелектуальні замки, які підтримують Wi-Fi або Bluetooth, інтегруються з домашньою мережею, надаючи власникам повідомлення в реальному часі та дозволяючи їм моніторити та керувати замками з будь-якої точки світу. Ця автоматизація та персоналізація налаштувань надає власникам додатковий рівень зручності та безпеки, роблячи інтелектуальні замки невід'ємною частиною сучасної системи безпеки «розумного будинку» [15].

Методи інтеграції із персональними асистентами

Інтеграція систем безпеки «розумного будинку» з голосовими помічниками є важливим кроком у сфері домашньої автоматизації, який відкриває нові можливості для керування захисними системами через голосові команди. Ця технологія дозволяє користувачам управляти такими системами, як сигналізації, відеонагляд і контроль доступу, використовуючи прості голосові команди, що спрощує налаштування режимів охорони, моніторинг статусу захисних систем та отримання сповіщень про безпекові інциденти.

Завдяки можливості голосових помічників інтегруватися з різноманітними компонентами систем безпеки, такими як дверні дзвінки з камерами, сенсори руху і димові детектори, можна створювати складні сценарії реакції на події. Наприклад, система може автоматично включати освітлення, коли в нічний час фіксується рух, або надсилати сповіщення, коли виявляється незвична активність.

Персональні асистенти можуть бути налаштовані на взаємодію з кількома користувачами, розпізнаючи їхні голоси і надаючи індивідуальні рівні доступу і контролю. Це дозволяє не тільки персоналізувати досвід використання, але й забезпечує додатковий рівень безпеки. Системи також використовують штучний інтелект для аналізу звичок користувачів та автоматичного адаптування налаштувань безпеки, підвищуючи ефективність роботи.

Загалом, використання персональних асистентів для управління системами безпеки значно спрощує повсякденне управління домом, дозволяючи користувачам зосередитись на своїх справах, поки інтелектуальна система безпеки активно працює на захист їх дому [16].

Методи виявлення поведінкових патернів

Методи виявлення поведінкових патернів у розумних домівках є фундаментальними для забезпечення безпеки, комфорту та енергоефективності. Ці методи базуються на зборі та аналізі даних з різноманітних датчиків і пристроїв, що інтегровані у систему «розумного будинку». Вони дозволяють ідентифікувати та аналізувати повторювані дії та звички мешканців, такі як часи активності в певних зонах будинку, використання побутових приладів та систем опалення чи кондиціонування.

Сучасні технології, особливо машинне навчання та штучний інтелект, відіграють ключову роль у процесі виявлення та адаптації до поведінкових патернів. Штучний інтелект може використовуватися для створення моделей нормальної поведінки та виявлення відхилень, які можуть вказувати на нестандартні події або потенційні загрози. Наприклад, якщо система помічає активність у будинку в нетиповий час, вона може сповістити власників або активувати додаткові заходи безпеки.

Також, аналіз поведінкових патернів використовується для оптимізації ресурсів. Наприклад, система може автоматично вимикати освітлення або регулювати клімат-контроль в зонах, де немає активності, що сприяє зниженню споживання енергії. Розумні домівки можуть використовувати аналіз поведінкових патернів для персоналізації досвіду мешканців. Наприклад, система

може автоматично налаштовувати освітлення та температуру відповідно до особистих переваг мешканців, які визначаються на основі їх попередніх дій.

Загалом, методи виявлення поведінкових патернів у «розумних будинках» створюють основу для більш інтелектуальної, адаптивної та безпечної домашньої автоматизації, розширюючи можливості традиційних систем безпеки та управління.

Методи домашньої автоматизації надають централізоване управління безпекою дому через мобільні додатки, включаючи розумне відеоспостереження, яке застосовує аналітичні алгоритми для ідентифікації осіб і відстеження відхилень у поведінці. Системи розумного освітлення автоматично адаптують освітлення, покращуючи безпеку та комфорт. Застосування машинного навчання в безпекових системах дозволяє ефективно моніторити та реагувати на незвичну активність. Інтелектуальні замки забезпечують безключовий доступ і дозволяють віддалене управління через мобільні додатки, забезпечуючи зручність і підвищений рівень захисту.

Крім фізичного вторгнення, значною загрозою для систем «Розумний дім» є кіберзлочини, пов'язані з витоками інформації. Ці системи обробляють велику кількість даних про поведінку та переваги користувачів, що робить їх вразливими до атак. Несанкціонований доступ до цих даних може призвести до порушення конфіденційності та використання інформації для маніпуляцій або шахрайства. Захист інформації включає шифрування даних, забезпечення мережевого захисту і регулярне оновлення безпекових протоколів, що вимагає інтегрованого підходу до кібербезпеки, об'єднання апаратних та програмних рішень для забезпечення належного захисту [17].

1.5 Вивчення методів для аналізу поведінкових даних

У розумних домівках аналіз поведінкових даних стає вирішальним для адаптації систем до індивідуальних потреб мешканців, підвищення рівня безпеки

та оптимізації загальної ефективності. Ці дані, зібрані з різноманітних вбудованих датчиків і пристроїв, відкривають детальний огляд повсякденних звичок і уподобань мешканців. Застосування розширених методів аналізу дозволяє ідентифікувати шаблони поведінки, що повторюються, та налаштовувати системи дому відповідно до змінюваних вимог і ситуацій, що сприяє створенню більш комфортного та раціонального житлового простору.

На рис. 1.7 представлено діаграму основних методів, які використовуються для аналізу поведінкових даних у системі «Розумний дім».

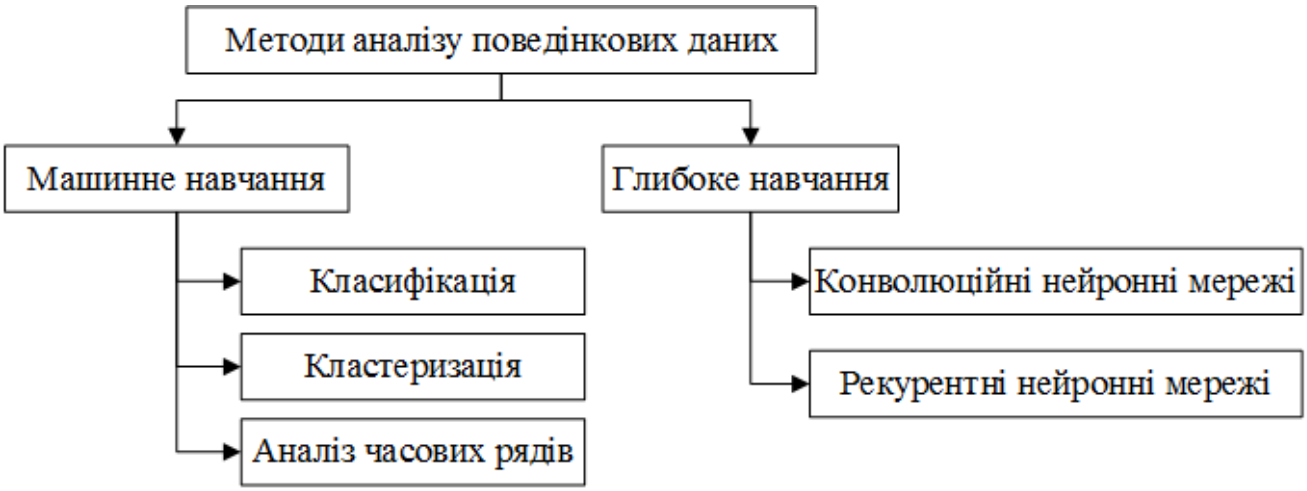


Рисунок 1.7 – Методи аналізу поведінкових даних

Один з основних інструментів для аналізу поведінкових даних – це машинне навчання (МН). Воно дозволяє системам ідентифікувати та прогнозувати поведінкові тенденції, адаптуючи реакції системи відповідно до звичних патернів життя мешканців. До методів МН належать:

– класифікація. Цей метод використовується для розпізнавання видів активностей у домі, наприклад, чи вдома хтось присутній чи ні. Застосування алгоритмів, таких як рандомні ліси або опорні векторні машини, може допомогти в ідентифікації різних типів активностей на основі даних з датчиків; [18]

– кластеризація. Метод кластеризації, такий як k-means, використовується для групування подібних подій або поведінкових трендів, що може допомогти у визначенні типових шаблонів поведінки або їх змін;

– аналіз часових рядів. Техніки аналізу часових рядів, такі як ARIMA чи LSTM (Long Short-Term Memory networks), використовуються для прогнозування

майбутніх поведінкових тенденцій на основі історичних даних, наприклад, для прогнозування змін у використанні енергії [19].

Методи глибокого навчання у свою чергу також забезпечують високу точність у виявленні складних поведінкових патернів через свою здатність виявляти нюанси великих наборів даних. До цих методів можна віднести:

– конволюційні нейронні мережі (CNN). Ефективні у розпізнаванні візуальних патернів з камер відеоспостереження, CNN можуть ідентифікувати не лише присутність людей, але й конкретні дії, такі як приготування їжі чи перегляд телевізора; [20]

– рекурентні нейронні мережі (RNN). Ідеально підходять для моделювання послідовних даних та прогнозування наступних дій на основі попередніх [21].

Методи для аналізу поведінкових даних у розумних домівках дозволяють не тільки збільшити комфорт і ефективність, але й значно підвищити рівень безпеки, роблячи повсякденне життя більш безпечним і зручним.

1.6 Оцінка ризиків і потенційних загроз для розумних домівок

Розумні домівки, хоча і надають значні переваги з точки зору зручності та енергоефективності, також створюють нові виклики та ризики з точки зору безпеки та приватності. Аналізуючи потенційні загрози для системи «Розумний дім», важливо розглянути кілька ключових категорій:

Фізичні вразливості

Фізичні вразливості в системах безпеки розумних домівок представляють собою слабкі місця, які можуть дозволити зловмисникам втручатися в роботу або фізично змінити обладнання. Сюди входять:

– незахищені точки доступу. Вікна, двері та інші входи без належного захисту, такого як датчики відкриття чи руху, становлять ризик для безпеки;

– фізичний доступ до критичної інфраструктури. Доступ до мережевих кабелів, маршрутизаторів чи серверів може дозволити зловмисникам здійснити атаку на систему;

– незахищене обладнання. Безпекові камери та системи управління доступом, доступні для маніпуляції, можуть бути легко саботовані або зламані;

– відсутність або пошкодження замків та бар'єрів. Пошкоджені замки та відсутність бар'єрних засобів можуть сприяти несанкціонованому доступу;

– фізичний доступ до інтерфейсів управління. Незахищені панелі управління, до яких можна дістатися без авторизації, відкривають потенціал для неавторизованих дій;

– недостатнє освітлення та відсутність спостереження. Недостатньо освітлені зони можуть ускладнити виявлення спроб несанкціонованого доступу, давши зловмисникам перевагу;

– недостатній фізичний захист важливих компонентів. Системи безпеки, які не обладнані для витримування атак або несанкціонованого доступу, можуть бути вразливими до знищення.

– відсутність або слабка сигналізація. Неадекватні або неефективні сигналізаційні системи можуть не здатні належно попередити про спроби проникнення або злом.

Мережеві вразливості

Мережеві вразливості представляють собою критичні слабкості в інфраструктурі, які можуть надавати можливість зловмисникам отримати несанкціонований доступ або провести атаку на систему. Ось декілька типових вразливостей:

– слабкі паролі. Прості або стандартні паролі легко можуть бути взламані через їх прогнозованість або доступність в публічних базах даних. Використання складних паролів зменшує ризик несанкціонованого доступу;

– незахищені Wi-Fi мережі. Відкриті або слабо захищені бездротові мережі можуть стати вразливими до атак «man-in-the-middle», де зловмисники можуть перехоплювати і маніпулювати мережевим трафіком;

– вразливі або застарілі мережеві протоколи. Використання старих протоколів, таких як WEP для Wi-Fi, що мають відомі вразливості, підвищує ймовірність несанкціонованого доступу;

– відсутність шифрування. Незашифровані мережеві з'єднання дозволяють легко перехоплювати і аналізувати дані, передані між пристроями;

– недостатні мережеві захисні механізми. Відсутність або недостатня реалізація мережевих брандмауерів або систем виявлення та запобігання вторгненням може не виявити або не заблокувати шкідливий трафік;

– вразливості у програмному забезпеченні маршрутизатора: Несвоєчасні оновлення або слабкості в прошивці маршрутизаторів можуть відкрити двері для атак.

Програмні вразливості

Програмні вразливості визначаються як слабкі місця або дефекти в програмному забезпеченні, що можуть бути використані для здійснення несанкціонованого доступу чи втручання в операції системи. Ось детальний опис основних типів таких вразливостей:

– неправильна обробка даних. Програмне забезпечення, що містить помилки у коді, може дозволити зловмисникам вводити шкідливі дані, що може призвести до збоїв у системі чи виконання неавторизованих команд;

– вразливості ін'єкції. Можливість «ін'єктувати» шкідливий код через вразливі точки введення, як-от веб-форми або API інтерфейси, відкриває шлях для різноманітних атак;

– застаріле програмне забезпечення. Програми, що не отримують регулярних оновлень, можуть залишатися вразливими до відомих атак через відсутність оновлень безпеки;

– несанкціоноване виконання коду. Програмне забезпечення, що не має належних обмежень на виконання коду, може дозволити зловмисникам запускати довільні команди, ставлячи під загрозу контроль над системою;

– неправильне управління сесіями. Вразливості у сесійному управлінні можуть дозволити зловмисникам копіювати або перехоплювати сесії користувачів, забезпечуючи несанкціонований доступ;

– конфігураційні вразливості. Неправильні або незахищені налаштування можуть залишити систему відкритою для атак або доступу до чутливих даних;

– відсутність шифрування. Якщо дані, що передаються або зберігаються, не захищені шифруванням, це може дозволити зловмисникам легко читати чутливу інформацію;

– неправильна обробка помилок. Програмне забезпечення, яке не приховує деталі помилок, може ненавмисно надати зловмисникам інформацію, яка може бути використана для розробки подальших атак;

– відсутність принципу найменших привілеїв. Системи, що надають користувачам або процесам більше прав, ніж необхідно, значно підвищують ризик вразливостей, якщо ці права скомпрометовані.

Вразливості обладнання

Вразливості обладнання представляють собою критичні слабкі місця у фізичних компонентах систем, таких як застарілі або незахищені пристрої, які можуть бути легко скомпрометовані. Ось ключові аспекти цих вразливостей:

– фізичний захист обладнання. Багато пристроїв Інтернету речей (IoT) не призначені витримувати фізичні атаки, що робить їх вразливими до зловмисників, які можуть фізично втручатися в їхню роботу або функціональність;

– застаріле обладнання. Пристрої зі старим апаратним забезпеченням часто містять відомі вразливості і рідко отримують оновлення, роблячи їх легкою мішенню для атак;

– відсутність оновлень безпеки. Навіть новітні пристрої можуть стати вразливими, якщо регулярні оновлення безпеки відсутні або не встановлені своєчасно;

– інтегроване ПЗ. Прошивка та інше вбудоване програмне забезпечення можуть містити помилки або вразливості, що зловмисники можуть використовувати для контролю над пристроєм;

– незахищені порти та інтерфейси. Відкриті USB-порти та інші фізичні інтерфейси можуть дозволити встановлення шкідливого програмного забезпечення або несанкціонований доступ;

– побічні канали витоку інформації. Атаки через побічні канали можуть використовувати електромагнітне випромінювання або аналіз споживаної потужності для витоку інформації з зашифрованих пристроїв.

Розумні домівки, при всіх їх перевагах, мають ряд вразливостей, які стосуються фізичного захисту, мережевої інфраструктури, програмного забезпечення та обладнання. Ці слабкі місця можуть викликати несанкціонований доступ та втручання в систему, що підкреслює необхідність комплексних заходів безпеки для забезпечення надійності та конфіденційності в розумних домах. Регулярні оновлення, посилення захисту мережі та підвищення фізичної безпеки є ключовими для зменшення цих ризиків.

1.7 Формулювання задачі дослідження

Основна мета полягає у створенні алгоритму машинного навчання, який за допомогою аналізу поведінкових патернів мешканців дозволить виявляти вторгнення в розумні домівки. Ключовим аспектом розробки цього методу є його інтеграція з інфраструктурою розумного будинку, що включає не тільки здатність виявляти та реагувати на кіберзагрози, а й взаємодію з іншими системними компонентами, такими як датчики руху. Розроблений алгоритм має бути сумісним із різноманітними пристроями та платформами, використовуваними у розумних

домах, що забезпечить гнучкість та ефективність у вирішенні завдань безпеки, аналізуючи дані для ідентифікації потенційно підозрілої активності, яка може вказувати на несанкціонований доступ.

Функціональні вимоги системи включають наступні ключові аспекти:

– виявлення та ідентифікація загроз. Система повинна автоматично розпізнавати та класифікувати потенційні кіберзагрози та аномальні поведінкові моделі, які можуть вказувати на спроби несанкціонованого доступу;

– прогнозування загроз. Використовуючи розроблений метод, система повинна аналізувати обсяги даних та поведінкові шаблони для передбачення потенційних спроб проникнення;

– реагування на інциденти. Система повинна не тільки оповіщати про виявлені загрози, але й активувати превентивні заходи для запобігання несанкціонованого доступу;

– оновлення захисних механізмів. Забезпечення можливості постійного оновлення захисних алгоритмів для адаптації до нових і еволюціонуючих загроз;

– аудит та звітність. Реалізація можливостей для детального аудиту безпекових подій та формування звітів про інциденти та заходи реагування, що сприятиме посиленню загальної безпекової політики.

Нефункціональні вимоги до системи забезпечують її ефективність та якість роботи в умовах реального часу і включають наступні аспекти:

– продуктивність. Система повинна ефективно обробляти вхідні дані та швидко реагувати на загрози, забезпечуючи мінімальні затримки у відповідях;

– масштабованість. Архітектура системи має дозволяти легке масштабування для задоволення збільшення обсягів даних та вимог користувачів, гарантуючи стабільну роботу при розширенні;

– надійність. Система має демонструвати високий рівень доступності та забезпечувати стабільність роботи з мінімальною кількістю помилок;

– безпека. Важливо забезпечити захист оброблюваних даних від несанкціонованого доступу та витоків, використовуючи сучасні методи шифрування та аутентифікації;

– інтегрованість. Система має бути сумісною з іншими компонентами розумного дому, забезпечуючи легку інтеграцію та синхронізацію функціоналу;

– зручність використання. Інтерфейс системи повинен бути інтуїтивно зрозумілим, що дозволяє користувачам легко управляти системою без спеціальних навичок.

Для досягнення стратегічної мети критично важливим є не тільки розробка технічно продуманої системи, але й забезпечення її зручності для користувачів. Успіх впровадження такої системи залежить від того, наскільки легким та інтуїтивно зрозумілим є її інтерфейс. Одночасно, система має бути надійною, здатною безперебійно працювати в різноманітних умовах та ефективно ідентифікувати загрози, мінімізуючи ризик помилок або збоїв у її роботі.

Висновок до розділу 1

У рамках даного розділу було проведено теоретичний огляд і формулювання проблеми, що лягла в основу дослідження розумних домівок. Аналіз концепцій та архітектур, зокрема централізованих та децентралізованих систем, дозволив визначити ключові компоненти інтелектуальних систем, такі як IoT пристрої, мережеві протоколи, обчислювальні ресурси та аналітичні інструменти. Значна увага приділялася вивченню поведінкових паттернів, які відображають звички мешканців, та аналізу існуючих методів виявлення вторгнень, зокрема через відеоспостереження, розумне освітлення та інтелектуальні замки. Досліджено методи для аналізу поведінкових даних, з особливим акцентом на машинне навчання та глибоке навчання, що включає класифікацію, кластеризацію та аналіз часових рядів. Оцінка ризиків та загроз показала, як фізичні, мережеві, програмні та обладнання вразливості можуть

вплинути на безпеку розумних домівок. Нарешті, формулювання задачі дослідження узагальнило функціональні та нефункціональні вимоги, необхідні для розробки відповідної системи. Зібрані дані покладено в основу наступного етапу дослідження, спрямованого на розробку математичної моделі та проєктування алгоритмів для виявлення аномалій поведінки жителів у системах розумного дому.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА МЕТОДУ ДЛЯ ВИЯВЛЕННЯ ВТОРГНЕНЬ

2.1 Вибір методу машинного навчання

Вибір методу машинного навчання для виявлення вторгнень у розумних будинках базується на аналізі поведінкових патернів, що дозволяє досягти високої точності та надійності в ідентифікації аномальних дій. Серед них виділяються три основні методи: метод Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ), метод опорних векторів (МОВ) та метод випадкового лісу (МВЛ).

Метод БФГШ є одним із найбільш відомих та ефективних алгоритмів для оптимізації нелінійних функцій. Цей метод належить до класу квазі-ньютонівських методів, які використовуються для знаходження мінімуму або максимуму функцій без необхідності обчислення другої похідної, що значно знижує обчислювальну складність. На рис. 2.1 показано алгоритм функціонування даного методу.

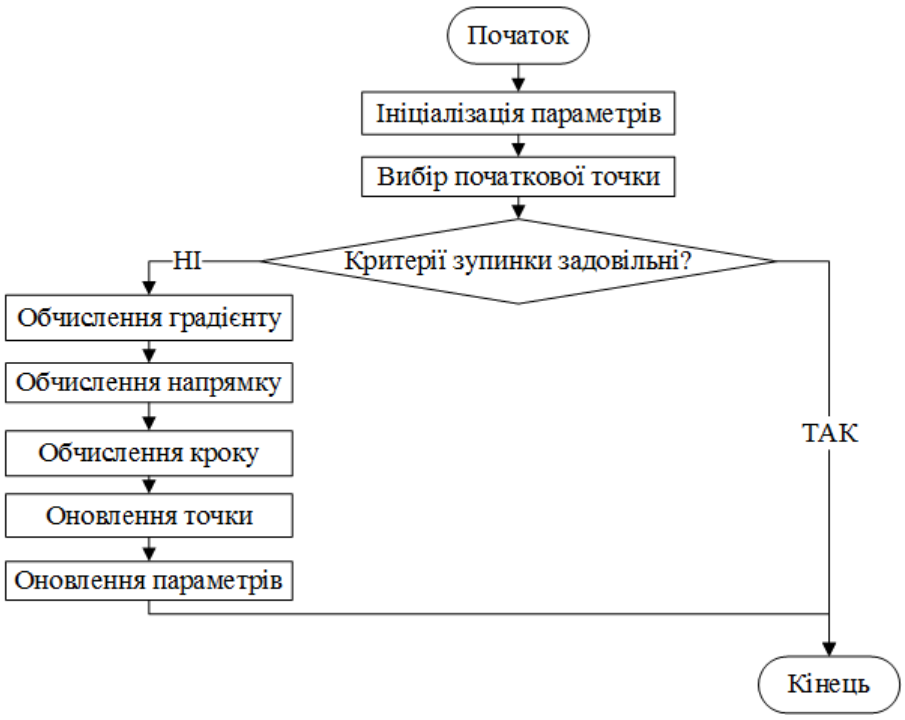


Рисунок 2.1 – Блок-схема алгоритму БФГШ

Алгоритм методу БФГШ розпочинається з ініціалізації параметрів, де визначають початкову точку x_0 та матрицю наближення до оберненої матриці Гессе H_0 [22]. Після цього вибирається початкова точка, з якої почнеться процес оптимізації. Далі, перевіряються критерії зупинки. Якщо вони задовольняються, алгоритм завершується. Якщо ні, виконується обчислення градієнту функції у поточній точці $\nabla f(x_k)$.

На основі градієнту обчислюється напрямок пошуку $pk=-H_k\nabla f(x_k)$. Наступним кроком є обчислення кроку α_k, який визначає, наскільки далеко потрібно рухатися в напрямку p_k. Крок визначається за допомогою методів лінійного пошуку. Після цього відбувається оновлення точки $x_{k+1}=x_k+\alpha_k p_k$. Оновлення параметрів включає в себе обчислення нових значень для H_{k+1}, яке здійснюється за допомогою специфічної формули оновлення, яка забезпечує наближення до оберненої матриці Гессе. Процес повторюється знову з перевірки критеріїв зупинки, і якщо вони задовольняються, алгоритм завершується, в іншому випадку повертається до обчислення градієнту для нової точки. Алгоритм триває доти, доки не будуть задоволені критерії зупинки.

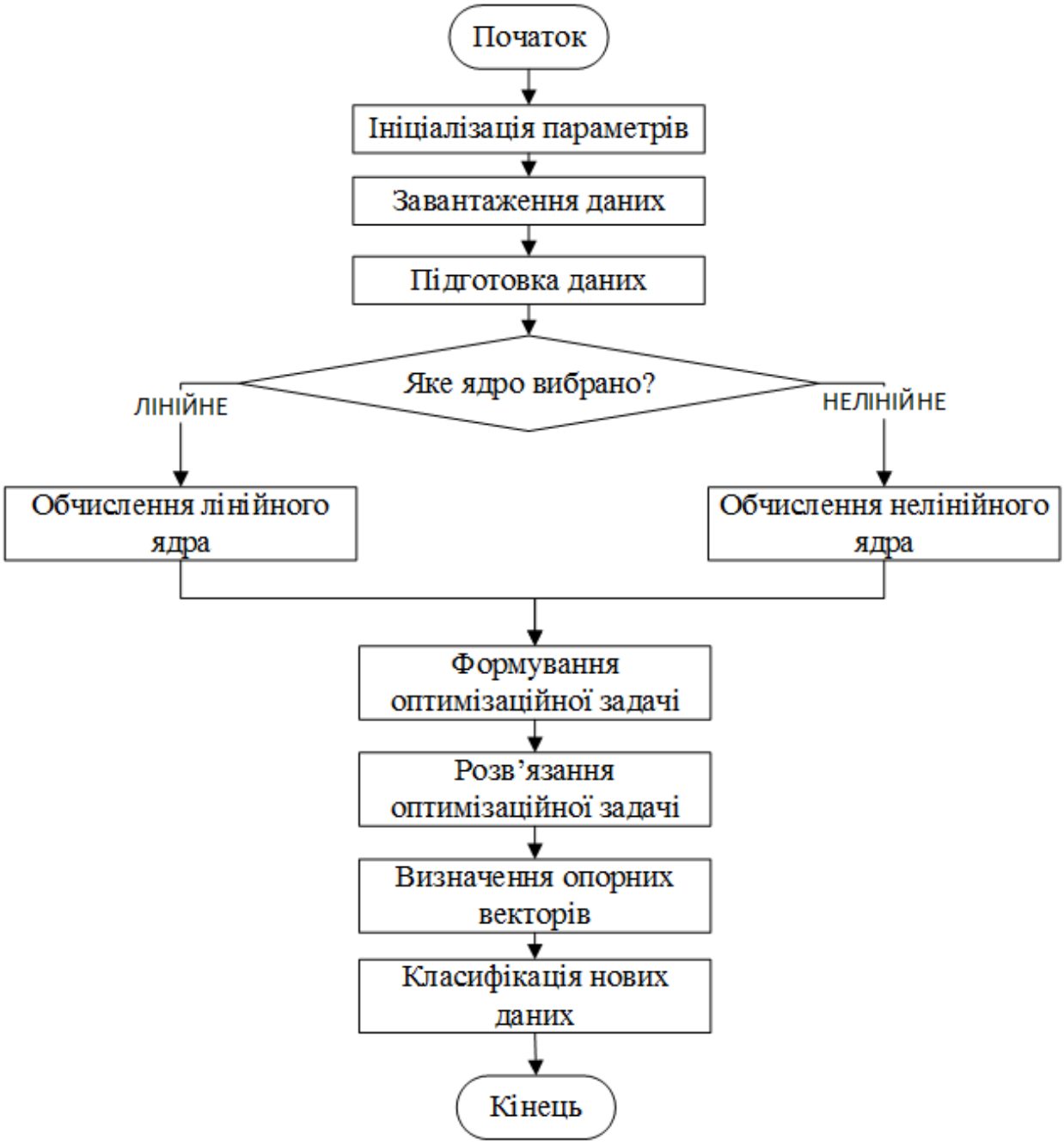
Для виявлення вторгнень у розумні будинки на основі аналізу поведінкових патернів метод БФГШ можна використовувати як ефективний інструмент оптимізації моделі машинного навчання. У цьому контексті, алгоритм БФГШ може застосовуватися для налаштування параметрів моделей, які аналізують поведінкові патерни мешканців та виявляють аномалії, що можуть свідчити про вторгнення.

Основна ідея полягає у тому, щоб зібрати дані про звичайну поведінку мешканців будинку та створити модель, яка може розпізнавати відхилення від цієї норми. БФГШ може бути використаний для оптимізації функції втрат такої моделі, забезпечуючи швидку та точну ідентифікацію аномальних дій. Зокрема, цей метод може допомогти налаштувати ваги нейронних мереж або інших алгоритмів, щоб мінімізувати похибки виявлення вторгнень [23].

Таким чином, метод Бройдена-Флетчера-Гольдфарба-Шанно забезпечує ефективну оптимізацію моделей машинного навчання, що є критично важливим

для своєчасного та точного виявлення вторгнень у розумних будинках, підвищуючи загальний рівень безпеки та надійності цих систем.

Метод опорних векторів є одним із найбільш потужних і широко використовуваних алгоритмів машинного навчання, призначених для розв'язання задач класифікації та регресії [24]. Основна ідея цього методу полягає у знаходженні гіперплощини, яка максимально розділяє класи в багатовимірному просторі, таким чином максимізуючи відстань до найближчих точок обох класів (опорних векторів). На рис. 2.2 представлений алгоритм роботи МОВ.



Рису нок 2.2 – Блок-схема алгоритму МОВ

МОВ розпочинається з ініціалізації параметрів та завантаження даних, які потім піддаються підготовці [25]. Підготовка даних включає їх нормалізацію та трансформацію у відповідний формат. Далі слідує вибір ядра. Якщо вибрано лінійне ядро, обчислюється лінійне ядро. У випадку вибору нелінійного ядра, обчислюється нелінійне ядро.

Після обчислення ядра, формується оптимізаційна задача, яка має на меті знайти оптимальні параметри моделі. Потім вирішується ця оптимізаційна задача за допомогою методів квадратичного програмування або інших оптимізаційних алгоритмів. Визначаються опорні вектори - це підмножина навчальних точок, які визначають гіперплощину розділення для класифікації даних.

На завершення, модель використовується для класифікації нових даних, базуючись на знайдених опорних векторах та обчисленій гіперплощині. Алгоритм завершується після успішної класифікації нових даних.

Для виявлення вторгнень у розумні будинки на основі аналізу поведінкових патернів метод опорних векторів може бути надзвичайно ефективним інструментом. Основна ідея полягає у створенні моделі, яка навчається на звичайній поведінці мешканців будинку і може виявляти відхилення від цієї норми. Збираються дані про нормальну поведінку мешканців з різних сенсорів розумного будинку (рухові сенсори, сенсори відкриття дверей, температурні сенсори тощо). Зібрані дані обробляються для видалення шумів і заповнення відсутніх значень, а також виконується нормалізація та масштабування даних для підвищення ефективності роботи алгоритму. Дані розділяються на тренувальну і тестову вибірки, після чого модель МОВ навчається на тренувальній вибірці, де кожен приклад позначається як нормальна або аномальна поведінка. Вибір ядрової функції (наприклад, лінійна, поліноміальна, радіально-базисна функція) дозволяє перетворити дані у вищий простір для ефективного розділення. Після навчання модель використовується для класифікації нових даних, де кожна нова подія аналізується для визначення, чи є вона аномальною. Модель ідентифікує аномалії як потенційні вторгнення, сповіщаючи систему безпеки про необхідність реагування. Використання методу опорних векторів у розумних будинках

дозволяє створити надійну систему безпеки, яка постійно аналізує поведінкові патерни мешканців і швидко виявляє аномальні дії, забезпечуючи додатковий рівень захисту та підвищуючи загальну безпеку житлових приміщень.

Метод випадкового лісу є одним з найпотужніших алгоритмів машинного навчання, який використовується для задач класифікації, регресії та інших видів аналізу даних. Основна ідея методу полягає у створенні ансамблю великої кількості рішень (дерев рішень), де кожне дерево навчається на випадковій вибірці даних та використовує випадкову підмножину характеристик. Цей підхід дозволяє зменшити ризик перенавчання моделі та підвищити її стійкість і точність, оскільки об'єднання результатів багатьох дерев дозволяє усереднити похибки окремих дерев. Метод випадкового лісу широко використовується в різних галузях, зокрема фінансовову, охорону здоров'я, біоінформатику, маркетинг, виявлення шахрайства та інші сфери, де необхідна висока точність та надійність прогнозів [26].

Для виявлення вторгнень у розумні будинки на основі аналізу поведінкових патернів метод випадкового лісу може бути особливо корисним. Основна ідея полягає у зборі даних про нормальну поведінку мешканців будинку за допомогою різноманітних сенсорів (рухові сенсори, сенсори відкриття дверей, температурні сенсори тощо). Ці дані використовуються для навчання моделі, де кожне дерево в лісі створюється на основі випадкової підвибірки цих даних і характеристик. Після навчання модель випадкового лісу здатна аналізувати нові дані та визначати, чи є вони нормальними або аномальними. Використовуючи ансамбль дерев, модель здатна ефективно виявляти аномалії та відхилення від нормальної поведінки, що може свідчити про потенційні вторгнення.

Алгоритм роботи МВЛ представлено на рис. 2.3.

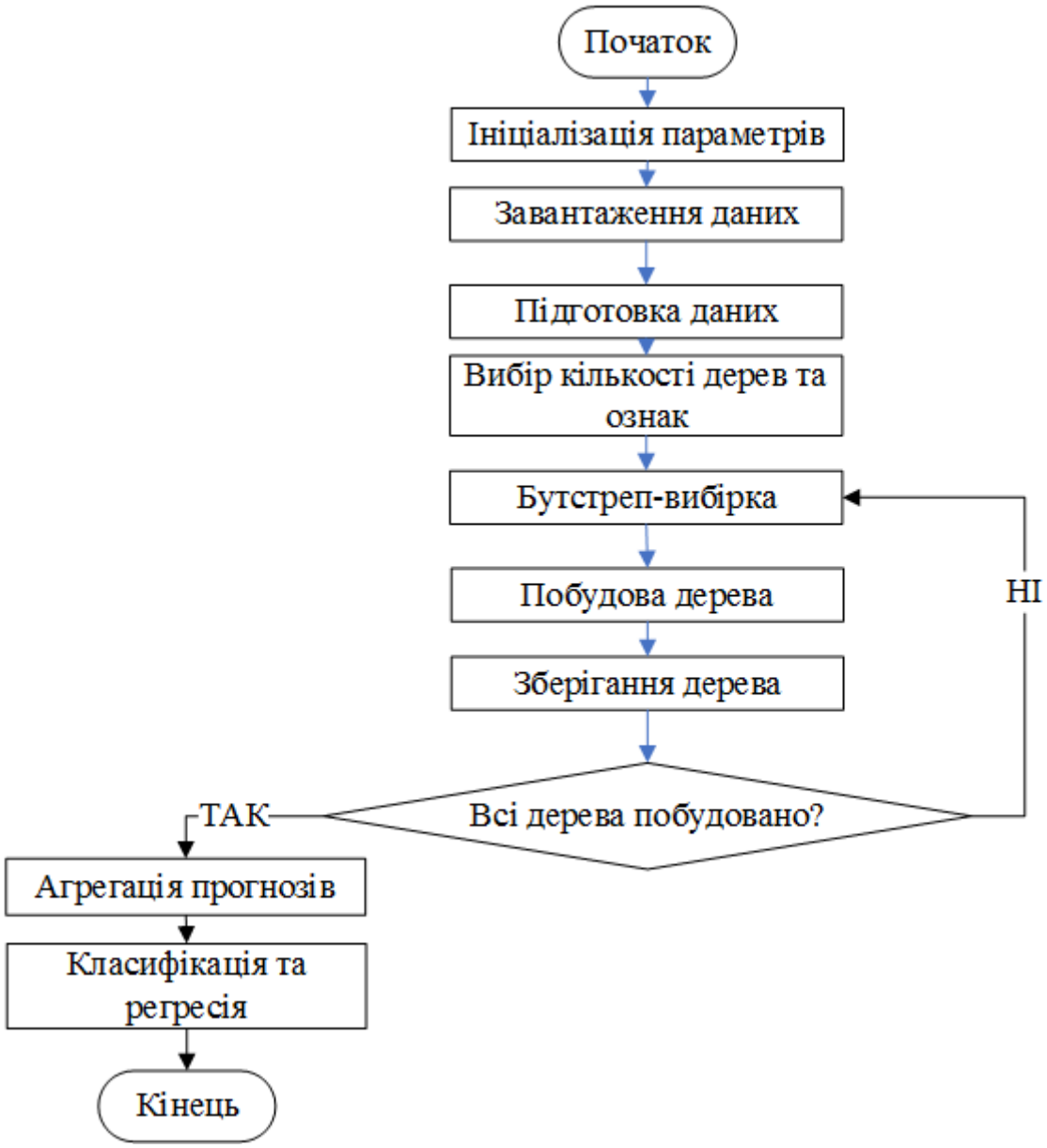


Рисунок 2.3 – Блок-схема алгоритму МВЛ

Метод випадкового лісу починається з ініціалізації параметрів, включаючи кількість дерев та ознак, які будуть використовуватися. Після цього дані завантажуються і готуються шляхом очищення, нормалізації та перетворення у відповідний формат.

Наступним кроком є вибір кількості дерев та ознак для кожного дерева, після чого проводиться бустреп-вибірка, що означає випадкове вибирання з повторенням з навчальної вибірки. Далі для кожного дерева здійснюється побудова дерева на основі бустреп-вибірки. Після побудови кожне дерево зберігається. Цей процес повторюється до тих пір, поки всі дерева не будуть побудовані.

Після того, як всі дерева побудовані, виконується агрегація прогнозів від усіх дерев. Цей крок включає обчислення середнього значення (для регресії) або голосування більшістю (для класифікації) на основі результатів окремих дерев. На завершення, агреговані прогнози використовуються для класифікації або регресії нових даних. Алгоритм завершується після виконання цієї класифікації або регресії [27].

Таким чином, метод випадкового лісу забезпечує надійний і стійкий інструмент для забезпечення безпеки розумних будинків, дозволяючи швидко і точно ідентифікувати можливі загрози.

Кожен з цих методів має свої переваги та особливості, які роблять їх придатними для різних аспектів аналізу поведінкових патернів і виявлення вторгнень. У табл. 2.1 наведено порівняння цих методів дозволяє оцінити їх переваги та вибрати найбільш підходящий метод для конкретних завдань.

Таблиця 2.1 – Порівняльний аналіз методів машинного навчання

Характеристика	БФГШ	МОВ	МВЛ
Область застосування	Оптимізація неперервних функцій	Класифікація та регресія	Класифікація та регресія
Складність обчислень	Висока, ефективна для великих задач	Висока при великій кількості ознак	Середня
Вимоги до даних	Менш чутливий до шуму у даних	Чутливий до шуму та масштабування	Менш чутливий до шуму і масштабування
Швидкість збіжності	Швидка збіжність у випадку гладких функцій	Залежить від вибору ядра та параметрів	Залежить від кількості дерев і глибини
Універсальність	Ефективний для широкого спектру оптимізаційних задач	Обмежений класифікацією та регресією	Обмежений класифікацією та регресією
Використання даних	Використовує градієнти для оптимізації	Потребує вибірку даних для тренування	Потребує вибірку даних для тренування
Здатність до паралелізації	Обмежена можливість паралелізації	Обмежена паралелізацією	Висока паралелізація

Метод БФГШ є оптимальним вибором для виявлення вторгнень у розумні домівки завдяки його здатності ефективно працювати з великими задачами та меншій чутливості до шуму у даних, що є важливим у динамічному середовищі розумних домівок. Його швидка збіжність у випадку гладких функцій дозволяє оперативно виявляти аномалії в поведінкових патернах мешканців. Крім того, метод БФГШ є універсальним інструментом, який можна застосовувати для широкого спектру оптимізаційних задач, що підвищує його ефективність у різноманітних умовах експлуатації.

2.2 Математична модель виявлення вторгнень

Математична модель виявлення вторгнень у розумних будинках базується на аналізі поведінкових патернів мешканців. Основна мета такої моделі полягає у створенні алгоритму, здатного розпізнавати аномальні дії, які можуть свідчити про потенційне вторгнення. Це досягається шляхом аналізу великих обсягів даних, що збираються різними сенсорами розумного будинку, та застосування методів машинного навчання для класифікації поведінки як нормальної або аномальної.

Задача полягає у виявленні аномалій у поведінкових патернах мешканців, що можуть свідчити про можливе вторгнення. Нехай X є матрицею ознак, де $X \in R^{n \times m}$ є набором даних з n прикладів і m ознаками. У нашому випадку $m = 6$, що відповідає шести ознакам класу *SmartHomeData*: *StartTime, Location, TimeOfDay, Object, Posture, Duration*. *Вектор* $Y \in \{0,1\}^n$ є вектором міток, де 1 означає «вторгнення», а 0 - «нормальна поведінка».

Цільова функція для оптимізації представлена як:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right], \quad (2.1)$$

де, $h_\theta(x)$ є сигмоїдною функцією, а θ – вектор параметрів моделі.

Матриця *X* формується з екземплярів класу *SmartHomeData*. Кожен рядок матриці *X* відображає один екземпляр *SmartHomeData*, а кожен стовпець відповідає одній з ознак:

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix} \quad (2.2)$$

Вектор *Y* формується з поля *Label* кожного екземпляра *SmartHomeData*:

$$Y = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{m1} & x_{m2} & \dots & x_{mn} \end{pmatrix} \quad (2.3)$$

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

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

$$\hat{Y} = h_\theta(x) = \frac{1}{1+\exp(-X\theta)} \quad (2.4)$$

Ці прогнози зберігаються в екземплярах класу *SmartHomePrediction* у полі *PredictedActivity*. Логіка виявлення потенційних вторгнень реалізується у методі *IsPotentialIntrusion*, який аналізує прогнозовані активності та час виконання дій.

2.3 Проєктування алгоритмів системи виявлення аномалій у поведінці жителів

Проєктування алгоритмів для виявлення аномалій у поведінці жителів розумних будинків є важливим завданням, яке вимагає ретельного аналізу та моделювання поведінкових патернів. Основна мета таких алгоритмів полягає в ідентифікації відхилень від нормальної поведінки, що може свідчити про потенційне вторгнення або інші небезпеки. Важливим аспектом є інтеграція сенсорних даних з різних джерел для створення цілісної картини поведінки, що забезпечує високу точність і надійність виявлення аномалій.

На рис. 2.4 показано блок-схему, яка демонструє процес навчання та зберігання моделі у системі.

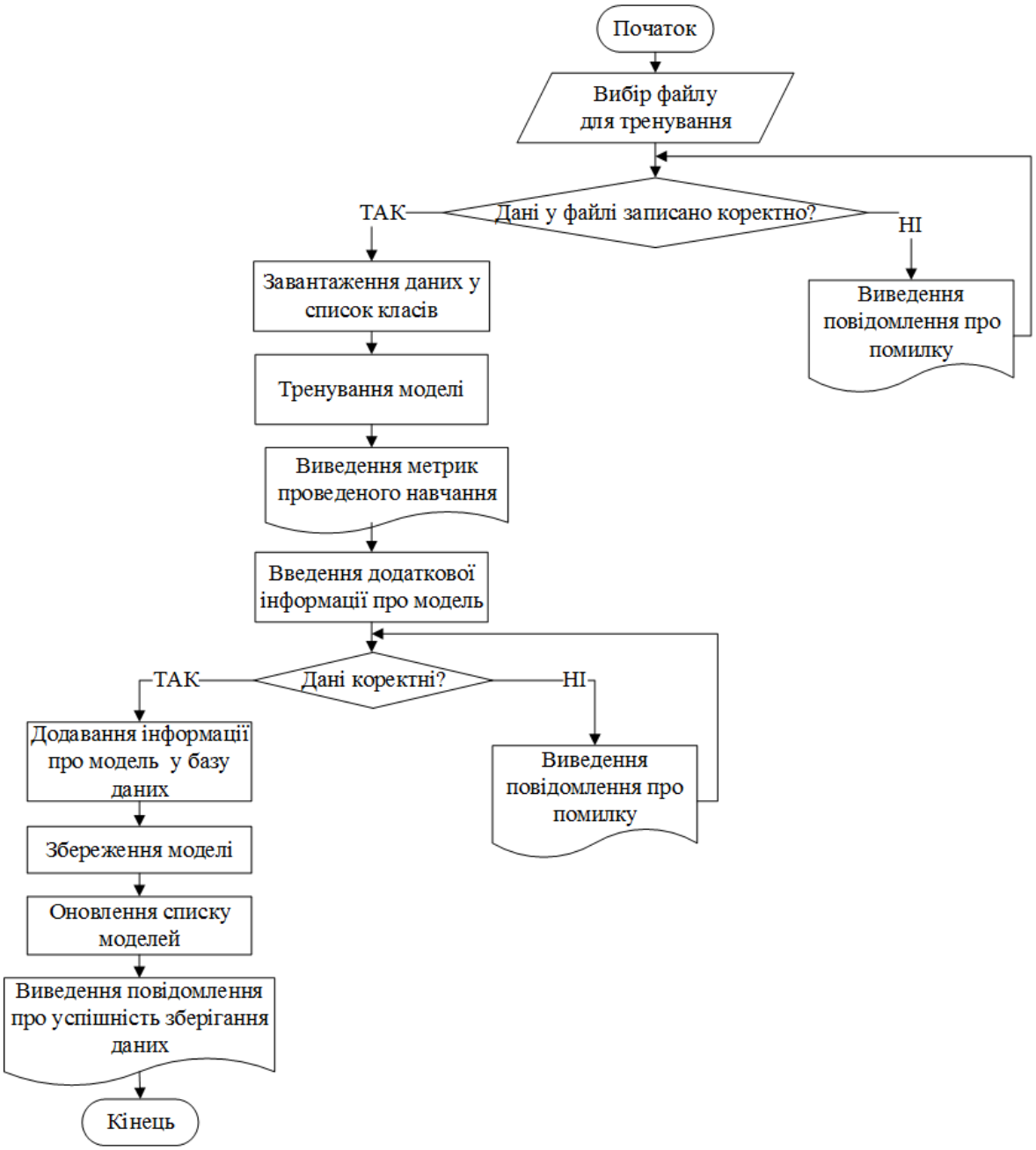


Рисунок 2.4 – Блок-схема алгоритму навчання та зберігання даних моделі

Алгоритм навчання та зберігання даних моделі починається з вибору файлу для тренування моделі. Після вибору файлу перевіряється, чи дані у файлі записано коректно. Якщо дані коректні, то відбувається завантаження даних у список класів, після чого розпочинається тренування моделі. Після завершення тренування моделі виводяться метрики проведеного навчання. Далі користувач вводить додаткову інформацію про модель. Після цього перевіряються введені користувачем дані. Якщо дані коректні, то інформація про модель додається у базу даних, модель зберігається, оновлюється список моделей, і виводиться

повідомлення про успішність зберігання даних. Якщо дані некоректні, виводиться повідомлення про помилку, і процес завершується. У випадку некоректності даних у файлі, також виводиться повідомлення про помилку і процес завершується.

Рис. 2.5 відображає блок-схему алгоритму роботи навченої моделі у реальному часі.

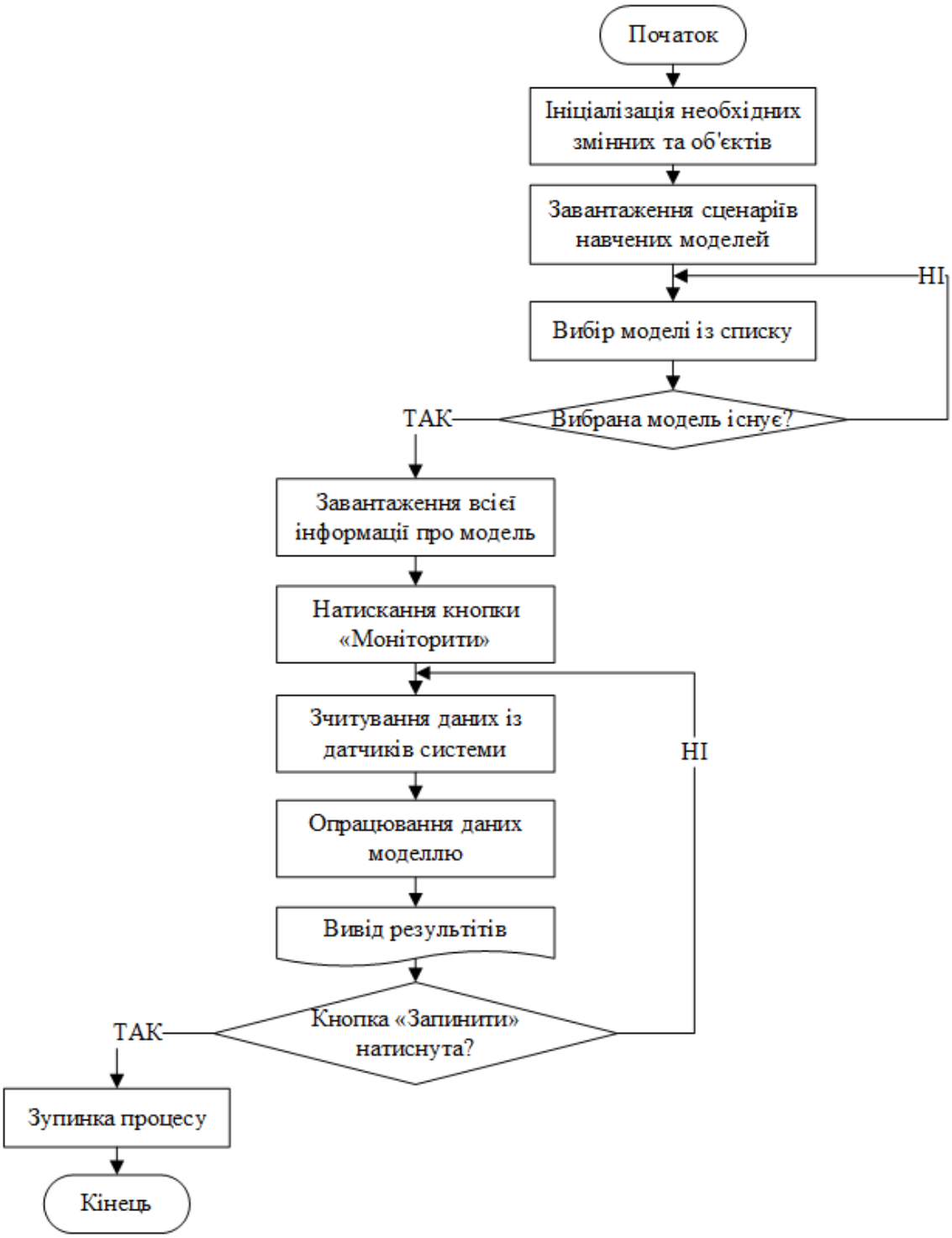


Рисунок 2.5 – Блок-схема алгоритму роботи моделі

Алгоритм роботи моделі починається з ініціалізації необхідних змінних та об'єктів. Далі завантажуються сценарії навчених моделей і виконується вибір моделі зі списку. Якщо вибрана модель існує, відбувається завантаження всієї інформації про модель. Користувач натискає кнопку «Моніторити», після чого зчитуються дані із датчиків системи. Дані обробляються моделлю, і результати виводяться на екран. Процес повторюється, поки не буде натиснута кнопка «Зупинити». Якщо кнопка «Зупинити» натиснута, відбувається зупинка процесу і алгоритм завершується. Якщо вибрана модель не існує, або дані із датчиків не можуть бути зчитані, виводиться повідомлення про помилку і процес завершується.

2.4 Вибір технологічних рішень для реалізації методу

Вибір технологічних рішень та інструментів для реалізації методу виявлення вторгнень у розумних будинках є критично важливим етапом, що визначає ефективність і надійність розробленої системи. Враховуючи складність задачі аналізу поведінкових патернів та необхідність обробки великих обсягів даних у реальному часі, необхідно обирати інструменти, які забезпечують високу продуктивність, масштабованість та зручність інтеграції.

Python є високорівневою мовою програмування, яка відзначається своєю простотою та читабельністю, що сприяє швидкій розробці та прототипуванню. Вона пропонує багатий набір бібліотек для машинного навчання та аналізу даних, таких як pandas, scikit-learn та TensorFlow, що дозволяє ефективно працювати з поведінковими патернами для виявлення аномалій. Завдяки своїй універсальності та підтримці інтеграції з різноманітними сенсорами і системами, Python є відмінним вибором для створення інтелектуальних рішень у розумних будинках [28].

Java є об'єктно-орієнтованою мовою програмування, яка забезпечує високу портативність завдяки принципу "пиши один раз, запускай де завгодно". Вона

відома своєю стабільністю та безпекою, що робить її придатною для розробки великих систем і обробки великих обсягів даних. Завдяки фреймворкам, таким як Apache Kafka та Spring, Java дозволяє створювати надійні та масштабовані рішення для моніторингу та аналізу поведінкових даних у реальному часі [29].

C# (C-Sharp) є сучасною об'єктно-орієнтованою мовою програмування, яка інтегрована у платформу .NET, що забезпечує високу продуктивність і зручність розробки. Вона ідеально підходить для створення настільних і веб-додатків, які потребують інтеграції з IoT-пристроями та сенсорами. Використання ML.NET дозволяє C# ефективно обробляти та аналізувати поведінкові дані, забезпечуючи розробку інтелектуальних систем для виявлення аномалій та підвищення безпеки розумних будинків [30].

У табл. 2.2 проведено порівняльний аналіз цих мов програмування за основними характеристиками, що впливають на реалізацію системи.

Таблиця 2.2 – Порівнялння мов програмування

Характеристика	Python	Java	C#
Швидкість виконання	Помірна, інтерпретована мова	Висока, компільована мова	Висока, компільована мова
Можливості для машинного навчання	Широкий набір бібліотек (TensorFlow, scikit-learn)	Обмежений набір бібліотек, частіше використовується з іншими мовами	Висока, завдяки ML.NET
Інтеграція з IoT-пристроями	Підтримка через бібліотеки	Підтримка через бібліотеки	Відмінна інтеграція з IoT-пристроями через .NET і Azure IoT
Підтримка багатопоточності	Обмежена	Висока, через вбудовані механізми	Висока, через вбудовані механізми
Масштабованість	Добра для прототипування, обмежена для великих проектів	Висока, підходить для великих проектів	Висока, підходить для великих проектів
Інструменти для розробки	Потужні інструменти для аналізу даних	Широкий вибір середовищ розробки	Потужні інтегровані середовища розробки

Вибір мови програмування C# для розробки системи виявлення вторгнень у розумні будинки є оптимальним завдяки її відмінній інтеграції з IoT-пристроями, що забезпечує безперебійну передачу та обробку даних у реальному часі. Завдяки ML.NET, C# надає потужні інструменти для розробки та впровадження моделей, що суттєво спрощує процес аналізу поведінкових патернів та виявлення аномалій. Висока підтримка багатопоточності в C# дозволяє ефективно обробляти великі обсяги даних та забезпечувати швидку реакцію системи на події, що є критично важливим для своєчасного виявлення загроз. Потужні інструменти для розробки, такі як Visual Studio, забезпечують високу продуктивність розробки та тестування додатків, що сприяє швидкому та якісному створенню системи.

2.5 Розробка прототипу системи

Розробка прототипу системи виявлення вторгнень у розумні будинки є важливим етапом, що дозволяє перевірити концепцію та визначити основні технічні вимоги. На цьому етапі реалізується базовий функціонал системи, який забезпечує збір та аналіз даних з різних сенсорів, а також виявлення аномалій у поведінці мешканців. Архітектура системи базується на трьохрівневій моделі, яка включає рівень даних, рівень логіки додатка та рівень представлення. Такий підхід забезпечує гнучкість, масштабованість та зручність у розробці та підтримці системи.

На рівні даних здійснюється зберігання та управління інформацією, отриманою від сенсорів і інших джерел. Рівень логіки додатка відповідає за обробку даних, виконання алгоритмів машинного навчання та прийняття рішень щодо виявлення аномалій. Рівень представлення забезпечує взаємодію користувачів із системою через інтерфейс, що дозволяє моніторити стан розумного будинку та отримувати сповіщення про потенційні загрози.

На рис. 2.6 зображено діаграму класів рівня даних системи, яка ілюструє структуру та взаємозв'язки основних компонентів на цьому рівні.

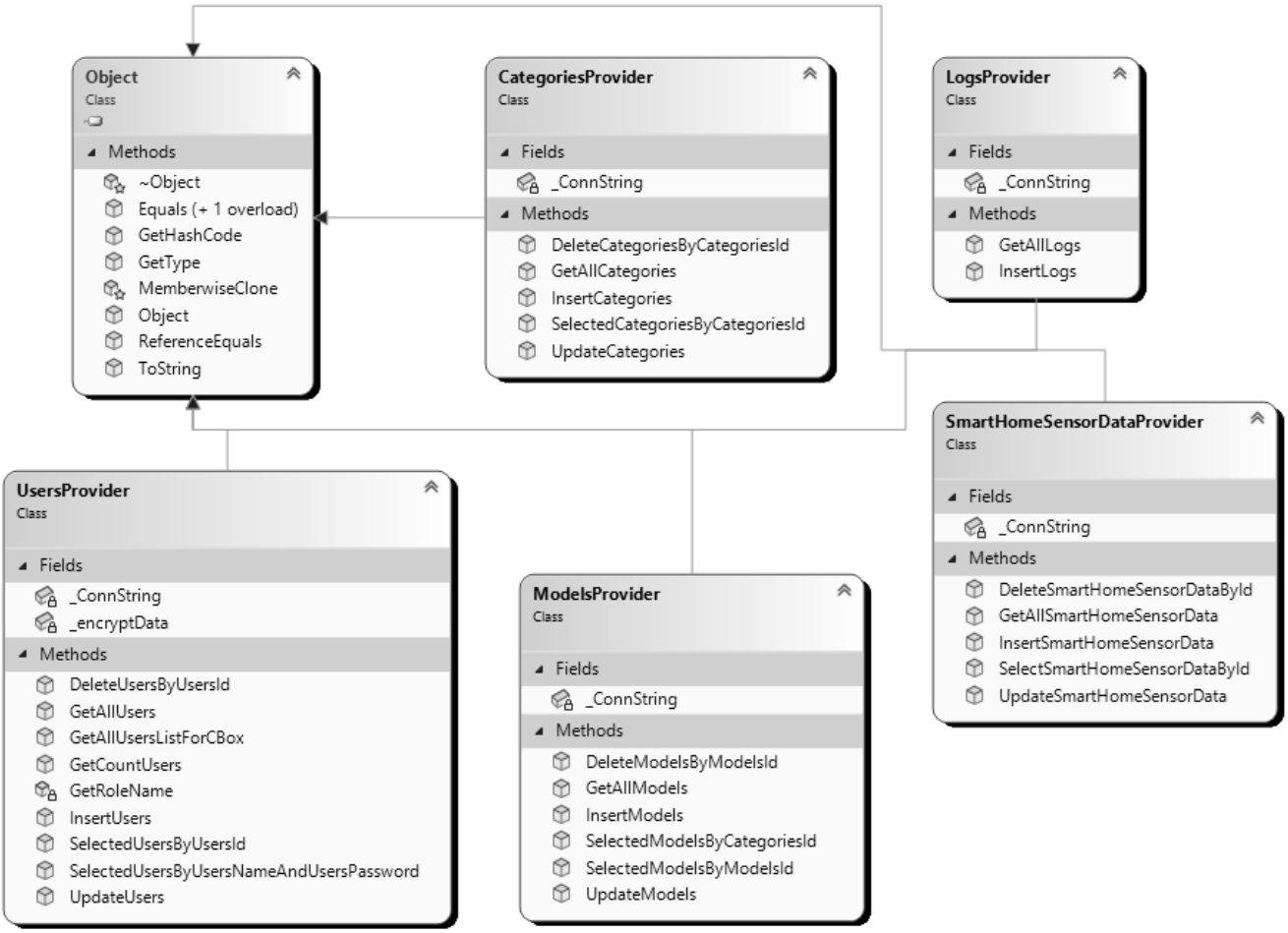


Рисунок 2.6 – Діаграма класів для рівня даних

Діаграма рівня даних складається із наступних класів:

– клас CategoriesProvider відповідає за керування категоріями моделей, які використовуються у системі. Цей клас забезпечує можливість створення, читання, оновлення та видалення категорій моделей. CategoriesProvider дозволяє зберігати інформацію про категорії моделей, їх опис та призначення, що допомагає організувати моделі машинного навчання у структурований та логічний спосіб;

– клас LogsProvider відповідає за керування подіями системи, забезпечуючи можливість відстеження та аналізу різноманітних дій та подій у системі. LogsProvider підтримує операції запису нових подій, зчитування історії подій, оновлення записів про події та їх видалення. Це дозволяє системі зберігати детальну інформацію про активність користувачів та системні процеси, що є критично важливим для моніторингу та безпеки;

– клас ModelsProvider відповідає за керування моделями машинного навчання, які використовуються для аналізу поведінкових патернів та виявлення аномалій. Цей клас забезпечує завантаження, навчання, зберігання та оновлення моделей, а також операції додавання нових моделей і видалення існуючих. ModelsProvider дозволяє ефективно організовувати роботу з моделями, включаючи їх оцінку та вдосконалення продуктивності для забезпечення точного виявлення вторгнень;

– клас SmartHomeSensorDataProvider відповідає за керування даними, що надходять від сенсорів розумного будинку, таких як температура, рух, відкриття дверей та вікон, рівень шуму, освітленість. Цей клас підтримує операції зберігання нових даних, зчитування існуючих, оновлення та видалення записів. SmartHomeSensorDataProvider дозволяє зберігати та обробляти великий обсяг сенсорних даних, необхідних для аналізу поведінкових патернів і виявлення аномалій;

– клас UsersProvider відповідає за керування користувачами системи, забезпечуючи можливість реєстрації нових користувачів, зчитування інформації про існуючих користувачів, оновлення їхніх даних та видалення користувачів із системи. UsersProvider підтримує операції з керування користувачами, що дозволяє ефективно контролювати доступ до системи та налаштовувати права користувачів для забезпечення безпеки та функціональності системи.

Рівень інтерфейсу користувача забезпечує взаємодію користувачів із системою, надаючи зручний та інтуїтивно зрозумілий доступ до функціоналу. На цьому рівні реалізуються інтерфейси для моніторингу стану розумного будинку, керування сенсорами, перегляду звітів та отримання сповіщень про потенційні загрози. Основна мета цього рівня – забезпечити ефективну взаємодію користувачів із системою, підвищуючи їхню обізнаність та контроль над безпекою будинку. Діаграма класів рівня користувацького інтерфейсу представлена на рис. 2.7.

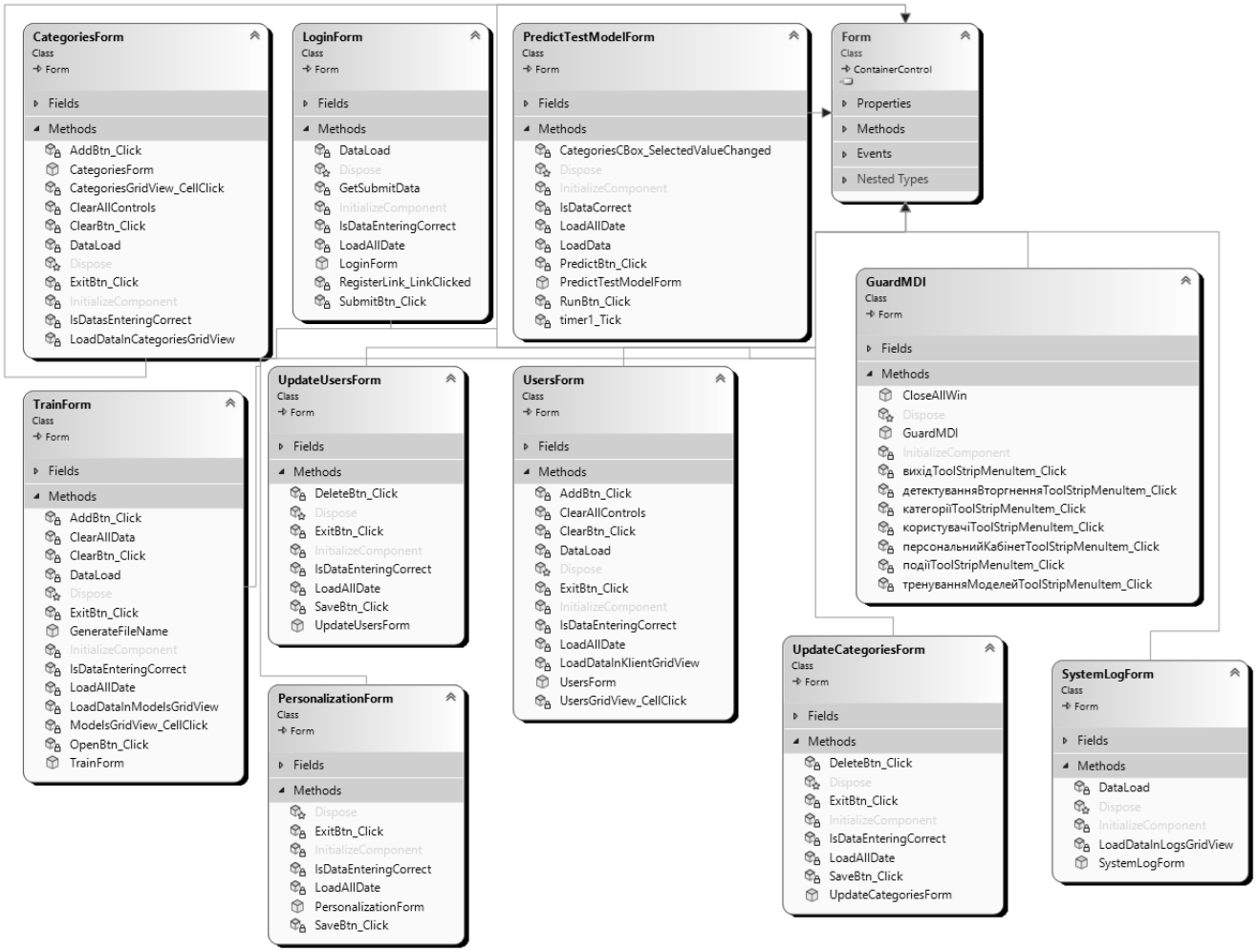


Рисунок 2.7 – Діаграма класів для рівня інтерфейсу користувача

Діаграма класів рівня користувацького інтерфейсу складається із:

– клас GuardMDI представляє головне вікно системи, яке забезпечує доступ до основного функціоналу та навігацію між різними модулями. Це вікно містить меню та панель інструментів для швидкого доступу до основних функцій системи, таких як моніторинг сенсорів, управління користувачами та налаштування системи;

– клас LoginForm відповідає за авторизацію користувачів у системі. Це вікно надає інтерфейс для введення облікових даних користувача та перевірки їхньої коректності, забезпечуючи безпечний доступ до функціоналу системи;

– клас PersonalizationForm дозволяє користувачам налаштовувати свої особисті параметри та уподобання у системі. Це вікно надає інтерфейс для зміни особистих даних, налаштувань відображення та інших персональних параметрів;

– клас SystemLogForm призначений для перегляду подій системи. Це вікно відображає журнал подій, що включає дії користувачів, системні повідомлення та інші важливі події, що відбуваються у системі;

– клас UpdateUsersForm забезпечує інтерфейс для редагування та видалення інформації про користувачів системи. Це вікно дозволяє адміністраторам змінювати дані користувачів, їхні права доступу та видаляти користувачів з системи;

– клас UsersForm відповідає за додавання нових користувачів та перегляд існуючих користувачів системи. Це вікно надає зручний інтерфейс для реєстрації нових користувачів та перегляду їхніх даних;

– клас PredictTestModelForm використовується для тестування моделей машинного навчання. Це вікно дозволяє завантажувати тестові дані, запускати моделі та переглядати результати їхнього прогнозування;

– клас CategoriesForm забезпечує інтерфейс для додавання та перегляду категорій моделей. Це вікно дозволяє адміністраторам створювати нові категорії, переглядати існуючі та організовувати моделі у структурований спосіб;

– клас TrainForm призначений для навчання моделей машинного навчання. Це вікно надає інтерфейс для завантаження тренувальних даних, налаштування параметрів навчання та запуску процесу навчання моделей;

– клас UpdateCategoriesForm забезпечує інтерфейс для редагування та видалення категорій моделей. Це вікно дозволяє адміністраторам змінювати інформацію про категорії та видаляти непотрібні категорії з системи.

Рівень бізнес-логіки відповідає за реалізацію основних алгоритмів і процесів, які визначають функціональність системи. На цьому рівні здійснюється обробка даних, виконання алгоритмів машинного навчання, прийняття рішень щодо виявлення аномалій та управління подіями системи. Основна мета цього рівня – забезпечити логічну обробку даних та взаємодію між рівнем даних і рівнем користувацького інтерфейсу, гарантуючи коректність та ефективність роботи системи. Діаграма класів рівня бізнес-логіки представлена на рис. 2.8.

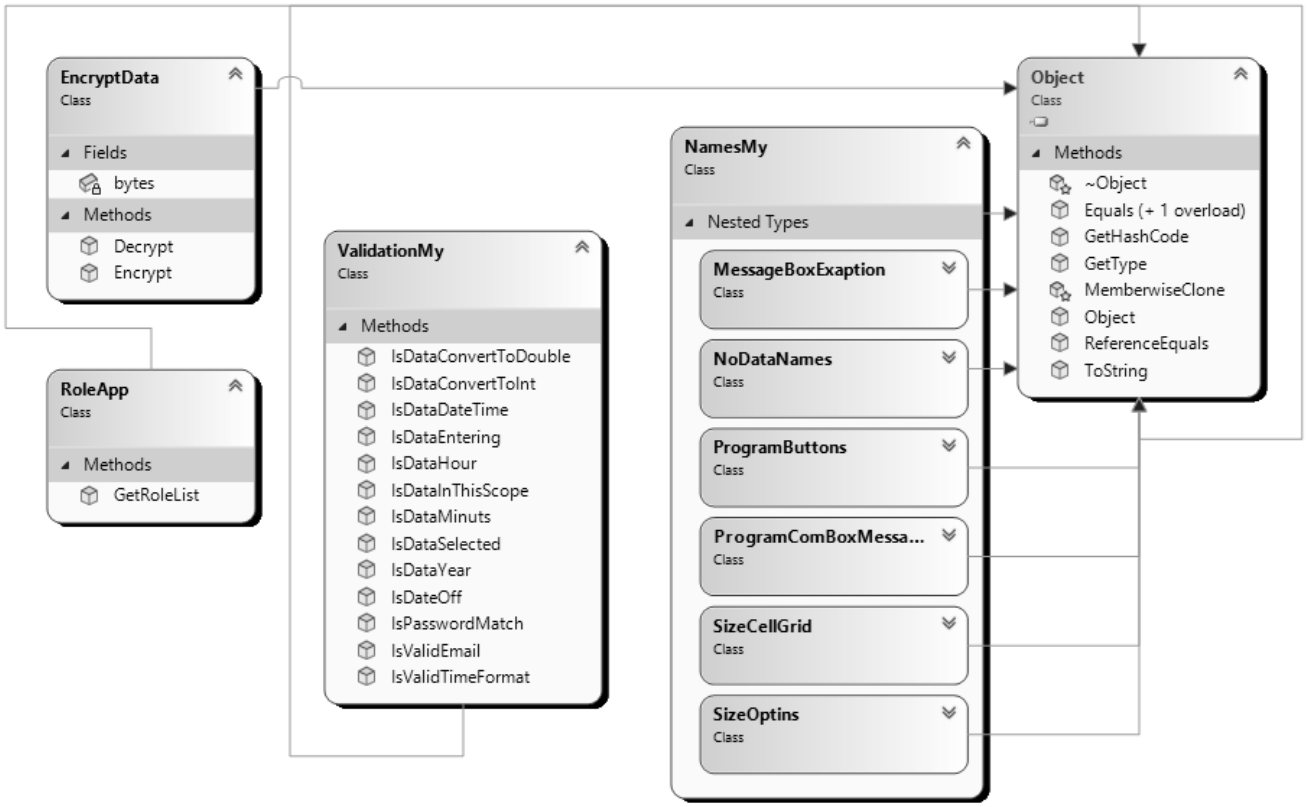


Рисунок 2.8 – Діаграма класів рівня бізнес-логіки

Основу даного рівня становлять:

– клас ValidationMy відповідає за перевірку коректності даних, які вводяться користувачами або обробляються системою. Він включає методи для валідації форматів, діапазонів значень та інших умов, необхідних для забезпечення цілісності даних. Це дозволяє запобігти введенню некоректних даних і забезпечити правильну роботу інших компонентів системи;

– клас NamesMy містить логіку для обробки імен та назв, які використовуються в системі. Він забезпечує стандартизацію, нормалізацію та перевірку імен, що дозволяє уникнути дублювань та забезпечити уніфікованість даних. Цей клас також може включати методи для генерації унікальних ідентифікаторів або псевдонімів для різних об'єктів системи;

– клас EncryptData відповідає за шифрування та дешифрування даних у системі. Він включає методи для забезпечення конфіденційності та безпеки чутливих даних, таких як особисті дані користувачів, інформація про моделі машинного навчання та інші критичні дані. Використання цього класу допомагає

захистити дані від несанкціонованого доступу та забезпечити безпечне зберігання і передачу інформації;

– клас RoleApp забезпечує управління ролями та правами доступу користувачів у системі. Він включає логіку для призначення ролей, перевірки прав доступу та управління дозволами для різних функцій системи. Це дозволяє забезпечити належний рівень безпеки, контролювати доступ до чутливих даних та функцій, а також підтримувати гнучку систему управління користувачами.

Кожен з цих класів виконує свої функціональні обов'язки на відповідному рівні архітектури системи та сприяє її загальній безпеці, цілісності та ефективності. Вони забезпечують взаємодію між даними, бізнес-логікою та користувацьким інтерфейсом, що є критично важливим для стабільної роботи системи виявлення вторгнень у розумні будинки. Такий структурований підхід гарантує надійність та масштабованість системи.

Висновки до розділу 2

У рамках даного розділу проведено детальне проектування та розробку методу для виявлення вторгнень у розумні будинки. Було проаналізовано потенційні методи машинного навчання, включаючи методи Бройдена-Флетчера-Гольдфарба-Шанно, опорних векторів та випадкового лісу. На основі порівняльного аналізу обрано метод БФГШ для реалізації. Описано математичну модель виявлення вторгнень, яка включає аналіз поведінкових патернів мешканців. Проведено проєктування алгоритмів для навчання та зберігання даних моделі, а також алгоритм роботи моделі. Проаналізовано технологічні рішення для реалізації методу, зокрема Python, Java та C#, з яких обрано C# як найоптимальніше для даної задачі. Реалізовано трьохрівневу модель системи, детально описано кожен рівень – даних, бізнес-логіки та користувацького інтерфейсу, а також побудовано діаграми класів системи.

3 ВИБІР КОМПОНЕНТІВ ТА РОЗРОБКА ЗАГАЛЬНОЇ АРХІТЕКТУРИ СИСТЕМИ

3.1 Аналіз вимог до апаратного забезпечення для системи виявлення вторгнень

Для забезпечення ефективної роботи системи виявлення вторгнень у розумні будинки необхідне належне апаратне забезпечення, яке включає кілька ключових компонентів. Основні елементи апаратної системи складаються із:

1. Контролер

Контролер є центральним компонентом, який здійснює управління всією системою. Він приймає дані від сенсорів, виконує попередню обробку та передає інформацію на сервер для подальшого аналізу. Контролер повинен мати достатню обчислювальну потужність для обробки великих обсягів даних у реальному часі, а також можливість інтеграції з різноманітними сенсорами.

2. Сенсори

Система виявлення вторгнень потребує широкого спектру сенсорів для моніторингу різних аспектів середовища. До основних сенсорів належать:

– рухові сенсори, що виявляють переміщення у приміщеннях.

– сенсори відкриття дверей та вікон, що визначають стан дверей та вікон (відкриті або закриті);

– температурні сенсори, що контролюють температуру в приміщенні;

– сенсори рівня шуму, що фіксують звукові аномалії;

– сенсори освітленості, що вимірюють рівень освітлення.

Ці сенсори повинні бути високоточними та надійними, оскільки від їхньої роботи залежить точність виявлення аномалій.

3. Сервер

Сервер виконує основні обчислювальні задачі, включаючи зберігання даних, навчання моделей машинного навчання та аналіз поведінкових патернів.

Він повинен мати високу продуктивність для обробки великих обсягів даних та підтримувати паралельні обчислення для швидкого виконання алгоритмів. Сервер також забезпечує зберігання історичних даних та управління базами даних, що є критично важливим для аналізу та звітування.

4. Мережева інфраструктура

Для забезпечення безперебійної передачі даних між сенсорами, контролерами та серверами необхідна надійна мережева інфраструктура. Вона повинна підтримувати високошвидкісні з'єднання та забезпечувати захист даних під час передачі.

5. Джерело безперебійного живлення (UPS)

Для захисту системи від можливих перебоїв у електропостачанні потрібно використовувати джерело безперебійного живлення (UPS). Це забезпечить автономну роботу системи під час відключень електроенергії, що дозволяє уникнути втрати даних і забезпечити безперервний моніторинг та безпеку.

Отже, ефективна робота системи виявлення вторгнень у розумні будинки залежить від правильного вибору апаратного забезпечення, яке включає потужний контролер, надійні сенсори, високопродуктивний сервер, надійну мережеву інфраструктуру та джерела безперебійного живлення. Ці компоненти забезпечують точне збирання, обробку та аналіз даних, що дозволяє своєчасно виявляти аномалії та забезпечувати безпеку житлових приміщень.

3.2 Огляд та вибір контролера для системи

Контролери в системах «Розумний дім» відіграють ключову роль у координації роботи домашньої автоматизації, керуючи множиною інтелектуальних пристроїв. Ці пристрої надають можливість дистанційного управління такими аспектами дому, як освітлення, клімат, безпека та інші, через мобільні додатки, панелі керування або голосові команди. Додатково, контролери підсилюють безпеку житла, інтегруючи в одну мережу різноманітні компоненти

безпеки — від датчиків руху до систем спостереження та оповіщення. Це забезпечує можливість моніторингу і своєчасної реакції на підозрілі дії або надзвичайні обставини. Користувачі мають змогу отримувати сповіщення на свої мобільні пристрої у випадках виявлення руху чи неавторизованого доступу, що забезпечує додатковий рівень захисту дому.

Arduino UNO виступає як одна з найпопулярніших платформ для створення електронних проектів, охоплюючи від простих до складних розробок. Ця платформа часто використовується в системах "Розумний дім" для різноманітних застосувань, зокрема для контролю та автоматизації домашніх процесів.

Рисунок 3.1 – Вигляд контролера «Arduino UNO»

Завдяки гнучкості та легкості програмування, Arduino UNO є вибором багатьох ентузіастів та фахівців у сфері "Розумний дім". Платформа здатна інтегрувати різноманітні модулі та датчики, перетворюючись на універсальний інструмент для розробки індивідуалізованих рішень. Arduino UNO може використовуватись для моніторингу споживання енергії у будинку, що дозволяє оптимізувати витрати та сприяти сталому споживанню ресурсів. Одним з важливих застосувань є розробка систем безпеки, до складу яких входять датчики руху, системи відеоспостереження та автоматичні замки дверей, що забезпечують високий рівень безпеки для домогосподарств.

Таблиця 3.1 –Технічні характеристики «Arduino UNO»

Характеристика	Опис
Мікроконтролер	ATmega328P
Робоча напруга	5 В
Вхідна напруга	7-12 В
Цифрові входи/виходи	14 (з яких 6 можуть служити як PWM)
Аналогові входи	6
Постійний струм на вході/виході	20 мА
Струм	50 мА
Частота кристала	16 МГц
USB-з'єднання	Так
ICSP заголовок	Так
Кнопка скидання	Так

Arduino забезпечує користувачам можливість безперешкодно інтегрувати та управляти домашньою автоматизацією, включаючи системи управління освітленням, температурою, безпекою та іншими побутовими функціями. Завдяки різноманітності доступних входів і виходів, Arduino UNO служить надійним інструментом для ентузіастів DIY та професіоналів, які прагнуть створити ефективні та економічно вигідні розумні системи для дому [31].

Raspberry Pi Zero вирізняється компактністю та високою продуктивністю, що робить її популярною для створення інноваційних проектів у сфері електроніки, в тому числі для систем "Розумний дім" (рис. 3.2). Ця плата ідеально підходить для автоматизації домашніх процесів завдяки можливості підключення широкого спектра датчиків та пристроїв, що дозволяє контролювати параметри як температура, вологість, освітлення та безпека. Наявність Wi-Fi робить Raspberry Pi Zero зручним варіантом для розробників систем "Розумний дім" через легкий доступ до мережевих ресурсів і можливість віддаленого управління [32].

Рисунок 3.2 – Вигляд контролера «Raspberry Pi Zero»

Таблиця 3.2 –Технічні характеристики «Raspberry Pi Zero»

Характеристика	Опис
Центральний процесор	Одноядерний
Оперативна пам'ять	512 МБ
Підключення	Mini HDMI, USB On-The-Go порт
Підтримка Wi-Fi	Так
Підтримка Bluetooth	Так
GPIO піни	Для підключення датчиків та пристроїв
Мікро SD слот	Для зберігання даних та ОС
USB-з'єднання	Micro USB
Вихідне відео	Mini HDMI
Робоча напруга	3.3 В

Отже, Завдяки своїм компактним розмірам та вбудованим можливостям з'єднання, включаючи Wi-Fi і Bluetooth, Raspberry Pi Zero є прекрасним варіантом для інтеграції у домашні середовища. Одноядерний центральний процесор та 512 МБ оперативної пам'яті забезпечують достатньо обчислювальних ресурсів для управління основними задачами домашньої автоматизації. Гнучкість у підключенні додаткових модулів і датчиків через GPIO піни робить цю плату практичним вибором для розробки та дослідження у сфері автоматизації дому.

STM32 Blue Pill – мікроконтролерна плата, яка активно використовується в електронних проектах, які варіюються від базових до високоспеціалізованих

систем (рис. 3.3). Ця плата є чудовим інструментом для систем "Розумний дім", оскільки забезпечує можливість інтеграції з широким спектром датчиків та пристроїв, що дозволяє ефективно контролювати та автоматизувати домашні процеси. STM32 Blue Pill допомагає реалізувати індивідуальні сценарії управління важливими аспектами домашнього життя, включно з освітленням, клімат-контролем та системами безпеки, пропонуючи високу гнучкість і ефективність в управлінні домашнім середовищем.

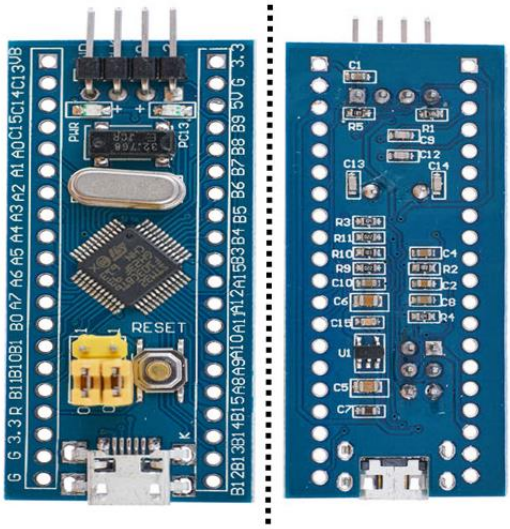


Рисунок 3.3 – Вигляд контролера «STM32 Blue Pill»

Таблиця 3.3 – Технічні характеристики «STM32 Blue Pill»

Характеристика	Опис
Мікроконтролер	STM32F103C8T6 (ARM Cortex-M3)
Робоча напруга	3.3 В
Вхідна напруга	7-12 В (через регулятор напруги)
Цифрові входи/виходи	37 (всі GPIO піни)
Аналогові входи	10
Постійний струм на вході/виході	20 мА (на пін)
Частота кристала	72 МГц
USB-з'єднання	Micro USB
Пам'ять	64 КБ Flash, 20 КБ SRAM
Додаткові функції	CAN, I2C, USART, SPI, USB, ADC

STM32 Blue Pill вважається потужним ресурсом для розумних домашніх систем завдяки своїй високій потужності, адаптивності та доступності. Побудована на базі ARM Cortex-M3, ця мікроконтролерна плата має значні обчислювальні ресурси, просунуті можливості підключення та численні входи та виходи, що робить її відмінним вибором для створення складних і глибоко інтегрованих рішень у сфері домашньої автоматизації. Ці характеристики відкривають широкі перспективи для керування освітленням, кліматом, безпекою та іншими інтелектуальними функціями [33].

Отже, Arduino UNO є відмінним вибором для інтеграції у системи виявлення вторгнень у розумні будинки, особливо з огляду на її високу популярність та доступність у спільноті розробників. Використання цієї платформи дозволяє швидко втілювати та тестувати різноманітні компоненти системи безпеки, як-от датчики руху та автоматичні замки, завдяки широкому спектру доступних модулів та датчиків. Гнучкість та легкість програмування Arduino UNO забезпечують зручність налаштування та інтеграції із мобільними додатками для моніторингу і контролю, що є критично важливим для сучасних систем "Розумний дім". Водночас, розширені можливості з'єднання і адаптації до споживчих потреб роблять Arduino UNO надійним вибором для створення ефективних рішень в області домашньої автоматизації. Все це робить Arduino UNO ідеальним інструментом для розробки інтелектуальних, адаптивних та безпечних систем виявлення вторгнень.

3.3 Вибір комплектуючих для системи

При розробці системи безпеки для інтелектуального дому, вирішальне значення має правильний вибір апаратних компонентів. Цей вибір має базуватися на принципах використання компонентів високої якості та точності, які здатні забезпечити надійну функціональність усієї системи безпеки. Невід'ємні складові

системи, такі як датчики руху, датчики температури, мікрофонні модулі та інші, є критичними для збору даних та реагування на зміни у середовищі.

Кожен елемент у цій колекції вибирається з огляду на його специфікації та внесок у цілісність системи безпеки. Датчики руху служать для детектування будь-якого несанкціонованого доступу, тоді як температурні датчики мають здатність виявляти значні коливання температури, що можуть сигналізувати про ризик пожежі чи інші небезпеки. Мікрофонні модулі, в свою чергу, призначені для ідентифікації підозрілих звуків, що є життєво важливим для повноцінної системи безпеки.

Отже, кожен вибраний компонент має свою чітко визначену функцію в структурі розумної системи безпеки. Вибір цих елементів ґрунтується на їхній точності, надійності та сумісності з іншими компонентами системи, що забезпечує не тільки високий рівень захисту, але й забезпечує стабільність та ефективність роботи системи "Розумний дім".

Датчик pir-d203s

Датчики руху є ключовими елементами в системах безпеки та автоматизації, дозволяючи відстежувати несанкціоновану активність або автоматично активувати певні системи при виявленні руху. Нижче представлено порівняльний аналіз трьох популярних моделей датчиків руху в табл. 3.4.

Таблиця 3.4 – Порівняльний аналіз датчиків руху

Характеристика	PIR-D203S	HC-SR501	AM312
Дальність детекції (метрів)	5	7	3
Кути виявлення	120°	110°	100°
Напруга живлення	3-6V	4.5-20V	2.7-12V
Споживана потужність	низька	висока	помірна
Чутливість	висока	середня	висока
Час затримки	налаштовується	налаштовується	2с

Вибір датчика PIR-D203S для систем "Розумний дім" надає значні переваги завдяки його широкому куту детекції, 120°, що дозволяє покривати більшу площу без необхідності інсталяції додаткових датчиків. Його дальність детекції до 5

метрів є достатньою для більшості застосувань у житлових та комерційних об'єктах. Низьке споживання енергії та можливість налаштування часу затримки роблять цей датчик енергоефективним та адаптивним до різних умов експлуатації, що є ключовим для "Розумних домів".

Датчик PIR (пасивний інфрачервоний) моделі D203S виявляє присутність людей за допомогою інфрачервоного випромінювання (рис. 3.4). Ці датчики є широко вживаними в системах безпеки та автоматизації домів, оскільки вони чутливі до змін інфрачервоного випромінювання, що відбувається, коли люди або тварини рухаються в полі детекції. Внутрішня структура D203S складається з двох елементів, які реєструють зміни в інфрачервоному випромінюванні, викликаючи електричний сигнал, який може бути оброблений для активації системних відповідей, таких як включення освітлення, спрацювання сигналізації, або запис відео.

Рисунок 3.4– Зовнішній вигляд датчика руху D203S

У контексті системи безпеки інтелектуального дому, датчик PIR D203S використовується як важливий збиральний пристрій даних, які аналізуються для ідентифікації звичних поведінкових патернів та виявлення незвичайних активностей. Це означає, що неочікуване спрацьовування датчика в нетиповий час може бути розглянуте як потенційна загроза, викликаючи відповідні заходи реакції від системи.

Датчик температури DHT22

Температурні датчики відіграють критичну роль у системах контролю та автоматизації, особливо в проектах "Розумний дім" та інших застосуваннях, пов'язаних з Інтернетом речей (IoT). Висока точність і надійність цих датчиків критично важливі для забезпечення стабільної і ефективної роботи систем. Один

із широко відомих датчиків, DHT22, виокремлюється завдяки своїй точності та довговічності. У табл. 3.5 представлено порівняльний аналіз датчика DHT22 із іншими моделями, такими як DHT11 та DS18B20, демонструючи його переваги для певних застосувань.

Таблиця 3.5 – Порівняльний аналіз датчиків температури

Характеристика	DHT22	DHT11	DS18B20
Діапазон вимірювання температури	-40°C до +80°C	0°C до 55°C	-55°C до +85°C
Точність вимірювання температури	±0.5°C	±2°C	±0.5°C
Діапазон вимірювання вологості	0% до 100%	5 - 95% RH	Не застосовується
Точність вимірювання вологості	±2% RH	±5% RH	Не застосовується
Частота оновлення даних	2 секунди	1 секунда	750 мс

DHT22 виділяється серед інших датчиків завдяки ширшому діапазону вимірювань температури та вологості та більшій точності цих вимірювань, що робить його більш адаптивним до різних умов експлуатації. Відмінно від DS18B20, який вимірює лише температуру, DHT22 також здатний фіксувати рівні вологості, що робить його більш придатним для застосувань, де важливо одночасно вимірювати обидва ці параметри. Ці характеристики підкреслюють його універсальність і високу практичність у широкому спектрі систем автоматизації.

Температурний датчик DHT22 інтегровано складається з ємнісного датчика вологості та термістора. Він також оснащений вбудованим аналого-цифровим перетворювачем (АЦП), який перетворює аналогові дані вологості та температури в цифрові сигнали (рис. 3.5).

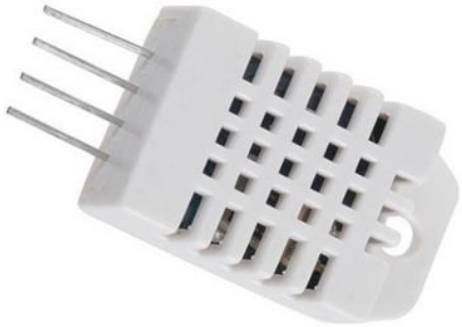


Рисунок 3.5 –Вигляд температурного датчика DHT22

Цей датчик визначає температуру в діапазоні від -40 до +80 градусів за Цельсієм з точністю до +/- 0.5°C та відносну вологість від 0 до 100% з точністю до +/- 2%. DHT22 має цифровий вихід, що дозволяє безпосередньо підключати його до мікроконтролерів, таких як Arduino, без потреби в зовнішніх аналого-цифрових перетворювачах, спрощуючи інтеграцію у системи.

У контексті інтелектуального дому, особливо в системах, які використовують машинне навчання для аналізу та забезпечення безпеки, DHT22 відіграє роль ключового датчика для моніторингу кліматичних умов. Це дозволяє системі вчасно реагувати на нестандартні зміни в умовах проживання, забезпечуючи додатковий рівень комфорту та безпеки.

Температурний магнітний датчик Reed module KY-025

Магнітні датчики мають широкий спектр застосувань, від елементарних домашніх задач до більш складних індустріальних систем. У табл. 3.6 викладено порівняльний аналіз цих датчиків, що підходять для використання у створенні розумних домашніх систем.

Таблиця 3.6 – Порівняльний аналіз магнітних датчиків

Характеристика	Reed Module KY-025	Hall Effect Sensor (A3144)	Magnetoresistive (HMC5883L)
Напруга живлення	3.3V - 5.5V	4.5V - 24V	2.16V - 3.6V
Розміри плати	1.5cm x 3.6cm	Залежить від моделі	Залежить від моделі
Вихідний сигнал	Цифровий	Цифровий	Аналоговий
Чутливість до сильних магнітних полів	Висока	Середня	Середня

Зокрема, Reed Module KY-025 виділяється серед інших магнітних датчиків. Його особливістю є спрощена інтеграція завдяки вбудованому компаратору LM393, а також висока чутливість до сильних магнітних полів, що робить його прекрасним вибором для проектів, де важливі надійність і легкість у використанні.

Магнітний датчик Reed Module KY-025 реагує на магнітне поле, змінюючи свій стан між відкритим і закритим, коли магніт наближається до датчика (рис. 3.6). Цей датчик включає скляний корпус, в якому розміщено два металеві контакти, що замикаються або розмикаються під впливом магніту. Така конструкція забезпечує його ефективну роботу в різних умовах, особливо коли потрібна точність у виявленні магнітних полів у домашніх та промислових системах автоматизації.

Рисунок 3.6 – Вигляд датчика Reed module KY-025

В системі захисту інтелектуального дому Reed Module KY-025 використовується для створення ефективної та простої системи детекції відкриття чи закриття дверей та вікон. Цей датчик реагує на наближення магніту, який знаходиться на рухомій частині двері чи вікна, до контактів датчика, встановленого на стаціонарній рамі. Закриття дверей або вікна призводить до замикання контактів датчика, сигналізуючи про закритий стан. Відкриття дверей або вікна змушує магніт віддалитися, що розмикає контакти і реєструється системою як відкритий стан.

Інтеграція Reed Module KY-025 у систему безпеки дозволяє автоматично відстежувати та реагувати на несанкціоновані спроби входу, наприклад, активуванням тривожного сигналу або відправленням сповіщень власникам.

Фоторезистор KY-018

Фоторезистори є неодмінними компонентами у світі електронних проектів, зокрема через їх здатність точно реєструвати зміни в рівнях освітлення. Ця характеристика робить їх незамінними у широкому спектрі застосувань. Наприклад, вони використовуються у простих освітлювальних системах, де їх головна задача — контроль за включенням і вимкненням світла в залежності від зовнішнього освітлення, тим самим економлячи енергію та покращуючи комфорт проживання. Для зручності порівняння, табл. 3.7 надає аналіз найбільш відповідних фоторезисторів для інтеграції у проекти.

Таблиця 3.7 – Порівняльний аналіз фоторезисторів

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

Один із варіантів, фоторезистор KY-018, виділяється серед інших за декількома важливими характеристиками, які представлено у таблиці 2.4. Ключовою перевагою KY-018 є його робочий діапазон напруги від 3.3V до 5V, що є ідеальним для більшості плат Arduino, забезпечуючи сумісність без необхідності застосування зовнішніх регуляторів напруги.

Фоторезистор KY-018 має опір у темряві 500 кОм, що забезпечує гарний компроміс чутливості для більшості застосувань. Це робить його більш адаптованим для загального використання порівняно з моделями GM5528 з тіньовим опором 1 МОм, що може бути надто чутливим для деяких проектів. Компактний розмір KY-018 робить його ідеальним для використання у просторово обмежених умовах, а його висока чутливість до світла дозволяє точно фіксувати рівні освітлення навіть при слабкому світлі. Температурний діапазон роботи KY-018, що коливається від -40°C до +85°C, є більш широким порівняно з моделями GM5528 і GL5516, забезпечуючи його стійкість у екстремальних умовах.

Крім того, KY-018 як готовий модуль для Arduino спрощує інтеграцію і використання, на відміну від моделей GM5528 і GL5516, які потребують додаткових налаштувань. Це робить KY-018 оптимальним вибором для користувачів різного рівня досвіду, особливо для тих, хто цінує простоту в налаштуванні і використанні.

Фоторезистор KY-018 (рис. 3.7) відрізняється своєю високою чутливістю до освітлення, що дозволяє йому змінювати електричний опір залежно від інтенсивності світла, що на нього падає. Ці зміни в опорі можуть бути використані для керування електронними схемами, активуючи або деактивуючи їх у відповідь на зміни в освітленості навколишнього середовища. У системах безпеки розумних будинків фоторезистори часто застосовуються для моніторингу світла, що може слугувати індикатором присутності людей або контролювати системи автоматичного освітлення.

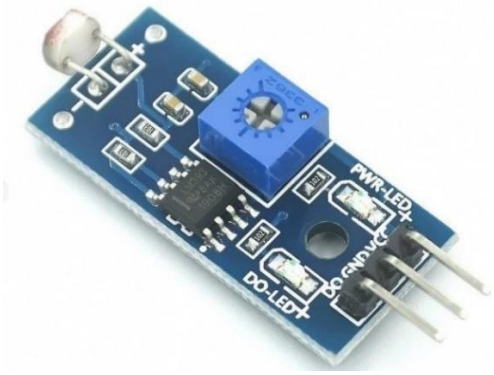


Рисунок 3.7 – Вигляд фоторезистора KY-018

У контексті аналізу поведінкових патернів, інформація отримана з фоторезистора KY-018 може бути використана для тренування машинно-навчальних моделей, щоб розпізнавати типові та атипові патерни освітленості. Це дозволяє системі більш точно виявляти потенційні загрози або необхідність корективних дій, забезпечуючи ефективніше та адаптивніше управління безпекою домашнього середовища.

Мікрофонний модуль MAX9814

Мікрофонні модулі є ключовими складовими для успішного реалізації цього проекту. Вибір відповідного мікрофонного модуля критично впливає на якість звуку та ефективність загальної роботи системи. У табл. 3.8 представлено аналіз трьох широко використовуваних мікрофонних модулів: MAX9814, LM4881 та MAX4466, що дозволяє виділити унікальні особливості кожного з них та визначити найкращий варіант для специфічних потреб.

Таблиця 3.8 – Порівняльний аналіз мікрофонних модулів

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

Серед розглянутих варіантів, модуль MAX9814 було вибрано через його здатність до автоматичного регулювання посилення (AGC), яке допомагає підтримувати постійний рівень аудіосигналу незалежно від зміни відстані до джерела звуку. Цей модуль також пропонує три можливості налаштування посилення (40dB, 50dB, 60dB), що забезпечує високу гнучкість у різноманітних аудіозастосуваннях. Додатково, низький рівень шуму цього модуля важливий для забезпечення чистоти звукового запису.

Мікрофонний модуль MAX9814 (рис. 3.8) представляє собою високоякісний аудіопідсилювач з функцією автоматичного контролю виграшу, ідеально підходящий для широкого спектру звукових систем. Цей модуль включає в себе електретний мікрофон, який ефективно захоплює звукові хвилі, та інтегрований підсилювач, який збільшує амплітуду звукового сигналу, що робить його незамінним інструментом у розробці звукових інтерфейсів.

Рисунок 3.8 – Вигляд модуля MAX9814

Основною властивістю мікрофонного модуля MAX9814 є його функція автоматичного регулювання виграшу (AGC), яка дозволяє адаптуватися до різноманітних акустичних умов середовища без потреби ручного втручання. У контексті систем безпеки для розумних будинків, MAX9814 ефективно використовується для ідентифікації звукових подій, що можуть свідчити про потенційні загрози чи екстрені ситуації. Завдяки своїй високій чутливості та здатності захоплювати якісний аудіосигнал, модуль здатний виявляти особливі звуки, як наприклад, звук розбитого скла, крики тривоги чи будь-яку несподівану активність у незвичний час.

Для інтеграції кожного такого модуля з платою Arduino UNO необхідне підключення до відповідних портів вводу/виводу, і це може потребувати додаткових компонентів, таких як резистори, транзистори або реле для управління різними навантаженнями. Важливим є також розробка спеціалізованого програмного забезпечення, що забезпечує коректне зчитування даних від датчиків і їх подальшу обробку для забезпечення адекватної реакції системи на зміни в середовищі.

3.4 Розробка скетчу та алгоритму управління системою

Розробка скетчу та алгоритму управління системою виявлення вторгнень у розумні будинки є важливим етапом реалізації методики аналізу поведінкових патернів. Основна мета полягає у створенні програмного забезпечення для контролера, який збиратиме дані з різноманітних сенсорів, оброблятиме їх та передаватиме на сервер для подальшого аналізу. Використання сучасних мікроконтролерів, таких як Arduino UNO, дозволяє ефективно інтегрувати різні типи сенсорів та забезпечити надійну роботу системи.

Для реалізації цього завдання було обрано датчики для зчитування основних параметрів середовища, таких як температура, вологість, рух, відкриття дверей та вікон, рівень освітленості та звуку. Дані, отримані від цих сенсорів, будуть використовуватися для визначення поведінкових патернів та виявлення аномалій, які можуть свідчити про потенційне вторгнення. Щоб забезпечити стабільну роботу та надійну передачу даних, було вирішено використати Ethernet-модуль для Arduino UNO. Це дозволяє здійснювати безперервний моніторинг і швидко передавати зібрані дані на сервер для обробки та зберігання. На рис. 3.9 наведено код для підключення бібліотек для роботи з датчиком температури та вологості, SPI та Ethernet для зв’язку з мережею.

```
#include <DHT.h>
#include <SPI.h>
#include <Ethernet.h>
```

Рисунок 3.9 – Підключення бібліотек

Бібліотека DHT.h використовується для роботи з датчиками температури і вологості DHT22, дозволяючи легко зчитувати ці дані. Бібліотека SPI.h відповідає за SPI-зв'язок, який необхідний для взаємодії Arduino з Ethernet-модулем. Бібліотека Ethernet.h забезпечує можливість підключення Arduino до мережі Інтернет через Ethernet-модуль, що дозволяє відправляти зібрані дані з датчиків на сервер для подальшої обробки та зберігання.

У фрагменті коду на рис. 3.10 налаштовуються піни для підключення різних датчиків до Arduino.

```
// Налаштування датчиків
#define DHTPIN 2          // Пін для DHT22
#define DHTTYPE DHT22     // Тип DHT22
DHT dht(DHTPIN, DHTTYPE);

#define PIRPIN 3          // Пін для PIR датчика
#define REEDPIN 4         // Пін для Reed модуля
#define LDRPIN A0         // Пін для фоторезистора
#define MICPIN A1         // Пін для мікрофонного модуля

// Ethernet налаштування
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };  // MAC-адреса Ethernet-модуля
IPAddress server(192, 168, 1, 100);                   // IP-адреса сервера бази даних
EthernetClient client;
```

Рисунок 3.10 – Налаштування пінів

Для датчика DHT22, який вимірює температуру та вологість, використовується пін 2, а також задається тип датчика. PIR-сенсор підключений до піна 3 для виявлення руху, а Reed модуль підключений до піна 4 для фіксації стану дверей або вікон. Фоторезистор підключений до аналогового піна A0 для вимірювання рівня освітленості, а мікрофонний модуль підключений до аналогового піна A1 для вимірювання рівня звуку. Налаштовується також Ethernet-з'єднання, де вказується MAC-адреса Ethernet-модуля та IP-адреса сервера бази даних, до якого буде здійснюватися передача даних. Використовується об'єкт EthernetClient для створення клієнта мережевого з'єднання.

На рис. 3.11 представлено функцію «setup», що виконує початкову ініціалізацію системи.

```
void setup() {
  Serial.begin(9600);
  dht.begin();
    // Ініціалізація Ethernet
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    while (true);
  }
    // Підключення до сервера
  if (client.connect(server, 80)) {
    Serial.println("Connected to server");
  } else {
    Serial.println("Connection to server failed");
  }
  pinMode(PIRPIN, INPUT);
  pinMode(REEDPIN, INPUT);
}
```

Рисунок 3.11 – Код функції «setup»

Спочатку налаштовується серійний зв'язок із швидкістю 9600 бод для відлагодження. Потім ініціалізується датчик DHT22 для зчитування температури та вологості. Відбувається ініціалізація Ethernet-з'єднання, і якщо не вдається отримати IP-адресу через DHCP, виводиться повідомлення про помилку та система переходить у нескінченний цикл. Після успішної ініціалізації Ethernet-модуля здійснюється спроба підключення до сервера на вказаній IP-адресі та порту 80, і залежно від результату виводяться відповідні повідомлення. У кінці, пін-коди для PIR-сенсора та Reed модуля налаштовуються як входи.

Рис. 3.12 відображає функцію «loop», виконує основний цикл роботи програми, в якому відбувається зчитування даних з усіх підключених датчиків і відправка цих даних на сервер.

```
void loop() {
  float humidity = readHumidity();
  float temperature = readTemperature();
  int pirState = readPIR();
  int reedState = readReed();
  int lightLevel = readLight();
  int soundLevel = readSound();
  sendDataToServer(temperature, humidity, pirState, reedState, lightLevel, soundLevel);
  delay(10000);
}
```

Рисунок 3.12 – Код функції «loop»

На початку зчитуються значення вологості та температури за допомогою відповідних функцій readHumidity() та readTemperature(). Потім зчитується стан PIR-сенсора, стан Reed модуля, рівень освітленості з фоторезистора та рівень звуку з мікрофонного модуля за допомогою функцій readPIR(), readReed(), readLight() та readSound(). Після збору всіх необхідних даних вони передаються на сервер за допомогою функції sendDataToServer(), яка відправляє всі зібрані значення. В кінці циклу виконується затримка на 10 секунд перед наступним зчитуванням даних.

Функція «sendDataToServer» відповідає за відправку зібраних даних на сервер (рис. 3.13).

```
// Функція відправки даних на сервер
void sendDataToServer(float temperature, float humidity, int pir,
int reed, int light, int sound) {
  String data = "GET /insert_data.php?";
  data += "temperature=" + String(temperature);
  data += "&humidity=" + String(humidity);
  data += "&pir=" + String(pir);
  data += "&reed=" + String(reed);
  data += "&light=" + String(light);
  data += "&sound=" + String(sound);
  data += " HTTP/1.1\r\n";
  data += "Host: 192.168.1.100\r\n";
  data += "Connection: close\r\n\r\n";

  if (client.connect(server, 80)) {
    client.print(data);
    Serial.println("Data sent to server");
  } else {
    Serial.println("Connection to server failed");
  }
  client.stop();
}
```

Рисунок 3.13 – Код функції «sendDataToServer»

Вона формує HTTP GET-запит, який включає параметри з даними про температуру, вологість, стан PIR-сенсора, стан Reed модуля, рівень освітленості та рівень звуку. Запит формується у вигляді рядка, який включає адресу PHP-скрипту на сервері, що приймає ці дані, та відповідні параметри з їх значеннями. Після формування запиту функція намагається встановити з'єднання з сервером через порт 80. Якщо з'єднання встановлено успішно, дані надсилаються на сервер, і виводиться повідомлення про успішну відправку. Якщо з'єднання не вдалося встановити, виводиться повідомлення про помилку. Після відправки даних або невдалої спроби з'єднання функція закриває з'єднання.

Загальний код скетчу приведено у додатку А.

3.5 Розробка загальної архітектури системи розумного будинку

Архітектура системи повинна враховувати взаємодію різноманітних компонентів, включаючи сенсори, контролери, сервери, мережеву інфраструктуру та користувацький інтерфейс. Важливою метою є створення гнучкої та модульної

системи, яка легко адаптується до змінних вимог і дозволяє додавати нові компоненти та функціональні можливості без значних зусиль.

На початковому етапі проектування було визначено основні компоненти системи, їх функціональні обов'язки та взаємодію між ними. Рис. 3.14 відображає загальну схему роботи системи, яка детально ілюструє взаємодію між усіма компонентами.

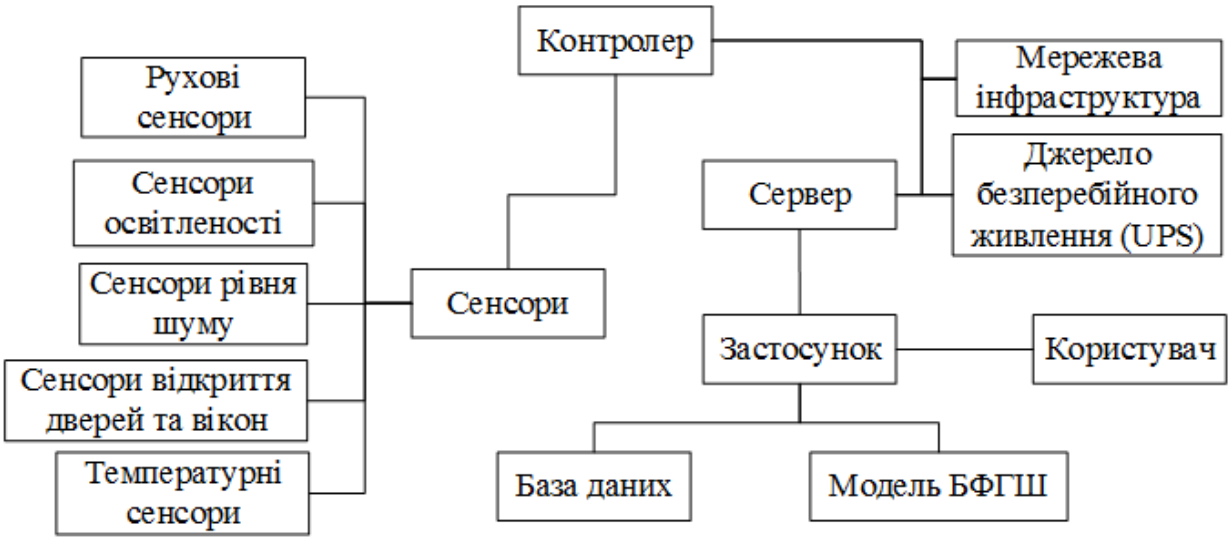


Рисунок 3.14– Схема взаємодії компонентів системи

Центральним елементом даної системи є контролер, який призначений для отримання даних від різноманітних сенсорів, обробляє їх та передає на сервер для подальшого аналізу. Сенсори, такі як рухові сенсори, сенсори освітленості, сенсори рівня шуму, сенсори відкриття дверей та вікон, а також температурні сенсори, зчитують відповідні параметри середовища та передають ці дані на контролер. Контролер координує роботу всіх сенсорів та виконує початкову обробку зібраних даних.

Сервер обробляє та аналізує отримані дані від контролера. Застосунок на сервері виконує основну логіку обробки даних, використовуючи модель Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ) для аналізу поведінкових патернів та виявлення аномалій. База даних на сервері зберігає всі зібрані дані, результати аналізу та історичні записи для подальшого аналізу та звітності. Мережева інфраструктура забезпечує надійне та швидке з'єднання між контролером та сервером, що дозволяє безперебійно передавати дані.

Джерело безперебійного живлення (UPS) забезпечує безперервну роботу системи у випадку перебоїв з електропостачанням, що дозволяє уникнути втрати даних та забезпечити безпеку системи під час неочікуваних відключень електроенергії. Користувач взаємодіє із системою через застосунок, який надає інтерфейс для моніторингу стану розумного будинку, отримання сповіщень про потенційні загрози та перегляду історичних даних. Застосунок дозволяє користувачам контролювати та налаштовувати систему відповідно до їхніх потреб.

Кожен модуль і датчик системи не лише виконують свою ізольовану функцію, але й стають частиною більшої інтегрованої системи, де дані та аналітика об'єднуються для створення безпечного, інтелектуального та реактивного домашнього середовища.

Висновки до розділу 3

У рамках даного розділу було здійснено аналіз вимог до апаратного забезпечення системи виявлення вторгнень, що охоплює необхідність використання контролера, сенсорів, сервера, мережевої інфраструктури та джерела безперебійного живлення. В ході дослідження проведено огляд потенційних контролерів, в результаті якого для інтеграції у систему було обрано Arduino UNO, завдяки його гнучкості та доступності для розробників.

Також визначено та обґрунтовано вибір основних компонентів системи, включаючи датчик руху PIR-D203S, датчик температури DHT22, температурний магнітний датчик Reed Module KY-025, фоторезистор KY-018, та мікрофонний модуль MAX9814 для ефективної реалізації виявлення різноманітних параметрів у середовищі розумного будинку.

Було розроблено скетч та алгоритм управління системою, який дозволяє коректно обробляти сигнали з різних сенсорів та відповідно реагувати на зміни в умовах середовища. Це забезпечує базу для майбутньої оптимізації та інтеграції

додаткових функцій. Завершальний етап розділу включав розробку загальної архітектури системи розумного будинку, де було побудовано схему взаємодії компонентів, забезпечуючи чітке уявлення про структуру та механізми взаємодії всіх елементів системи.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ МЕТОДУ

4.1 Налаштування експериментального середовища

Використання Arduino UNO вимагає інсталяції Arduino Integrated Development Environment (IDE), яка є основним інструментом для програмування мікроконтролерів Arduino. Ініціація процесу включає завантаження та установку Arduino IDE. Потім Arduino UNO підключається до комп'ютера через USB-кабель, що забезпечує активацію плати, про що сигналізують увімкнений світлодіод «ON» та миготіння світлодіода «L», демонструючи виконання завантаженої програми Blink.

Для продовження роботи важливо визначити номер COM порту, який комп'ютер присвоїв Arduino UNO. Це можна зробити, переглянувши «Порти (COM і LPT)» в Диспетчері пристроїв Windows, де вказаний потрібний порт для Arduino UNO. Встановлення зв'язку з платою в Arduino IDE відбувається з використанням цього номера порту COM.

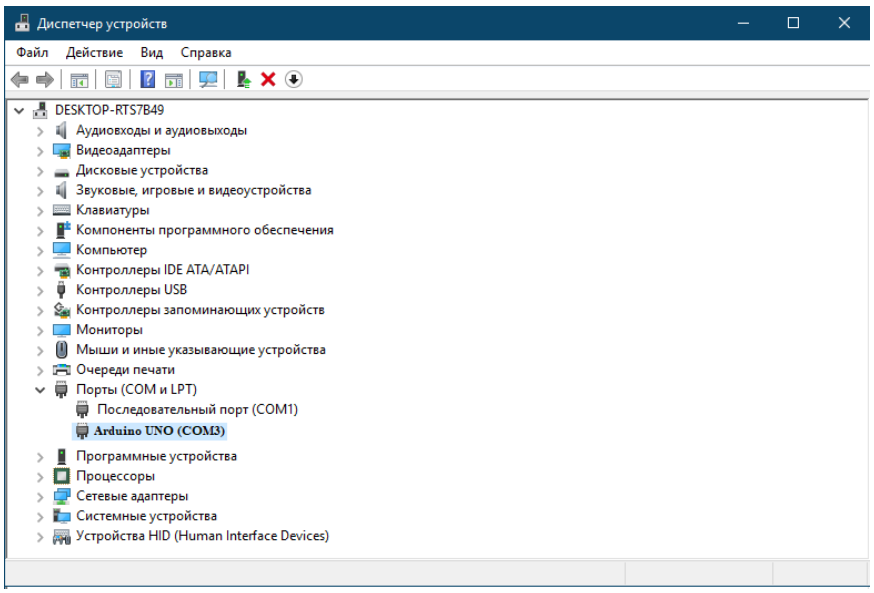


Рисунок 4.1 – Визначення порту Arduino UNO в диспетчері пристроїв Windows

При першому підключенні плати Arduino UNO до комп'ютера, операційна система автоматично розпізнає новий пристрій і проводить процес ідентифікації.

У цьому випадку, система успішно встановлює Arduino UNO як COM-порт, автоматично вибирає і встановлює потрібний драйвер, присвоюючи порту певний номер, наприклад, «COM3». Важливо врахувати, що кожне нове підключення плати Arduino до комп'ютера може вимагати присвоєння нового унікального номера COM-порту. У випадку використання декількох плат Arduino одночасно, кожна з них отримує свій індивідуальний номер порту.

Далі, для коректної взаємодії з платою Arduino UNO, яка підключена до порту «COM3», необхідно налаштувати Arduino IDE. В цьому програмному середовищі користувачу слід відкрити меню «Сервіс», обрати пункт «Послідовний порт» і встановити вказаний порт «COM3» (рис. 4.2). Після цього налаштування Arduino IDE матиме точне знання про те, що контролер підключений до порту «COM3», що дозволяє правильно налаштовувати середовище для завантаження коду на контролер та моніторингу даного порту для отримання інформації з контролера.

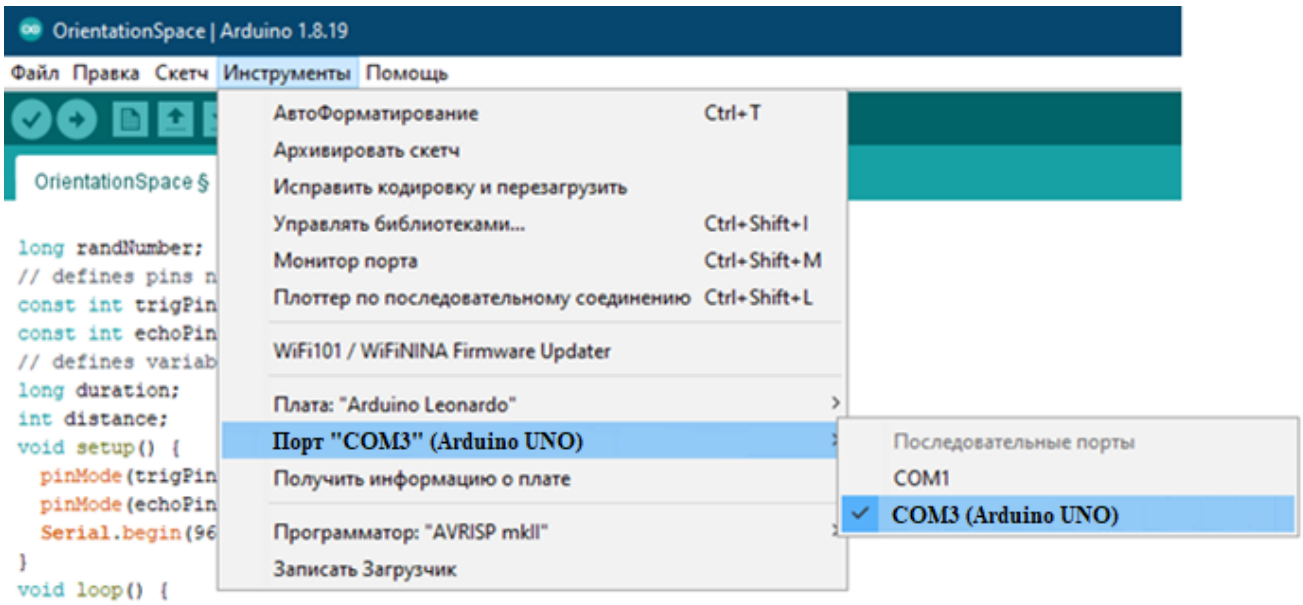


Рисунок 4.2 – Вибір послідовного порту для з'єднання з Arduino UNO

Цей етап має вирішальне значення для ефективного взаємодіяння програмного коду з апаратурою, забезпечуючи синхронну роботу написаного користувачем коду з фізичними діями плати Arduino в реальному часі. Закінчивши налаштування номера COM-порту для плати Arduino UNO, наступний крок полягає у вказівці конкретного типу контролера для

використання. Це важливо для забезпечення відповідності між програмним середовищем Arduino IDE та апаратними специфікаціями плати, що є критичним для коректної компіляції та завантаження програми. В Arduino IDE в меню «Сервіс» потрібно обрати пункт «Плата» та вибрати «Arduino UNO» зі списку доступних опцій, щоб середовище належним чином налаштувало всі необхідні параметри для цієї моделі плати.

Після встановлення параметрів, настає час для написання та перевірки програми. Програмування в Arduino IDE ведеться на вбудованій мові, заснованій на C/C++. Коли програма буде написана, її потрібно скомпілювати, щоб переконатися у відсутності помилок, використавши комбінацію клавіш «Ctrl+R». Цей процес виявляє синтаксичні та логічні помилки. Якщо помилок не знайдено, на дні вікна Arduino IDE з’являється повідомлення «Компіляція завершена», що підтверджує готовність програми до завантаження на плату для її виконання.

Завершення компіляції без помилок є ключовим моментом у процесі розробки, оскільки це свідчить про те, що програма написана без помилок і може безперешкодно функціонувати на платі. Це також гарантує, що всі використані команди та функції підтримуються апаратними можливостями Arduino UNO, що забезпечує точну та ефективну роботу програми (рис. 4.3).

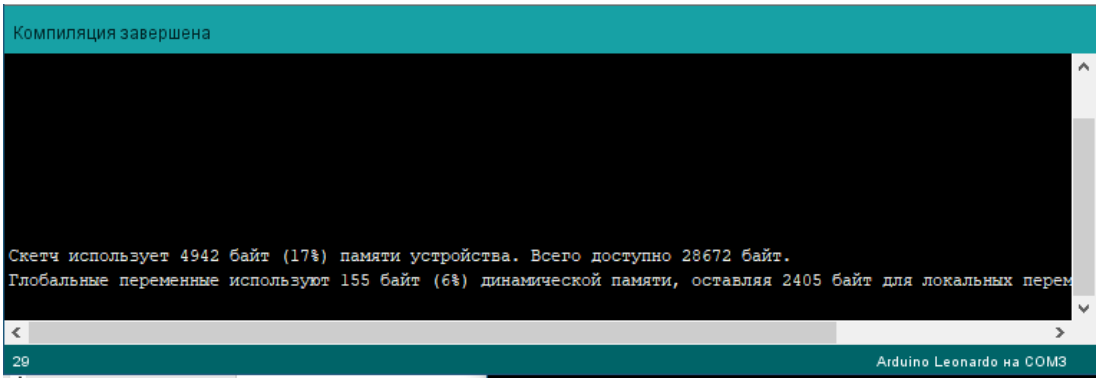


Рисунок 4.3 – Візуалізація процесу компіляції програми в Arduino IDE

Після успішного завершення компіляції програми без виявлених помилок, наступним етапом є її завантаження на мікроконтролер Arduino. Цей процес відбувається через інтерфейс Arduino IDE, де користувач має виконати кілька послідовних дій. Насамперед, у програмному меню слід обрати вкладку «Скетч», під якою розміщено опцію «Завантаження». Вибір цієї функції ініціює передачу

скомпільованої програми з комп'ютера на плату Arduino, що становить останній крок перед її виконанням у реальному часі (рис. 4.4).

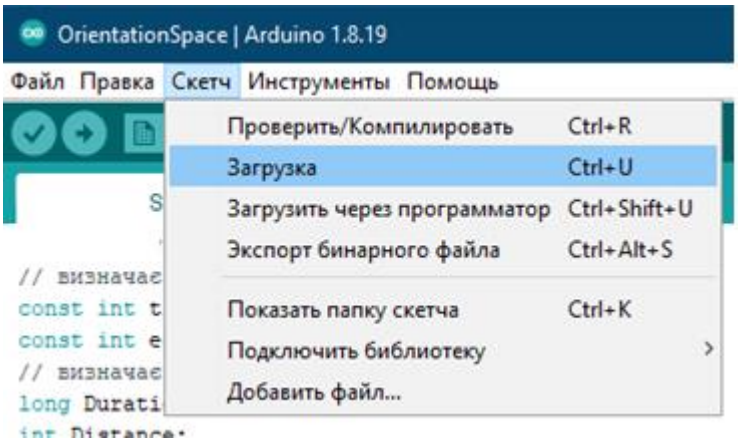


Рисунок 4.4 – Процес завантаження програми на мікроконтролер

Коли передача програми завершується, в інтерфейсі Arduino IDE з'являється інформаційне повідомлення у нижній частині вікна, яке підтверджує успішне завантаження. Це повідомлення слугує важливим індикатором того, що програма була не тільки правильно скомпільована, але й ефективно інтегрована у апаратуру контролера (рис. 4.5).

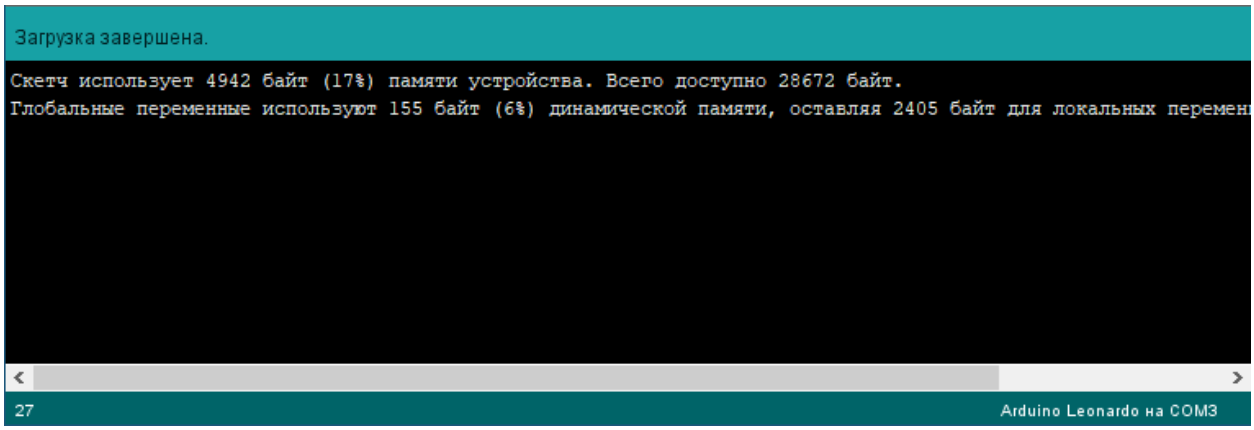


Рисунок 4.5 – Підтвердження завантаження програми на мікроконтролер

4.2 Збір даних для оцінки ефективності методу

Для реалізації системи «розумного будинку» було обрано набір даних з платформи Kaggle, відомої ресурсу для дослідників та розробників у сфері машинного навчання. Цей набір даних, доступний за вказаним посиланням [34],

забезпечує можливість аналізувати поведінкові моделі в контексті «розумного будинку». Він включає дані, зібрані з різних сенсорів, встановлених у такому будинку, такі як датчики руху, температури, вологості, звукового тиску, а також датчики, які реєструють зміни у системах освітлення та світлового оточення. Ці дані є незамінними для розробки розуміння поведінкових шаблонів та підвищення безпеки в «розумних» системах.

Процес підготовки даних перед використанням включав нормалізацію, щоб забезпечити високу якість та однорідність необхідних для навчання моделей даних. Початковий аналіз допоміг виявити і виправити аномалії та відхилення. Нормалізація вирівнювала різні масштаби даних, забезпечуючи консистентність, яка сприяє ефективному навчанню. Також було проведено фільтрацію, під час якої відбулося видалення шумів та випадкових помилок, що могли б вплинути на точність прогнозів моделі. На рис. 4.6 представлено знімок екрану з даними, підготовленими до використання у тренуванні моделі машинного навчання.

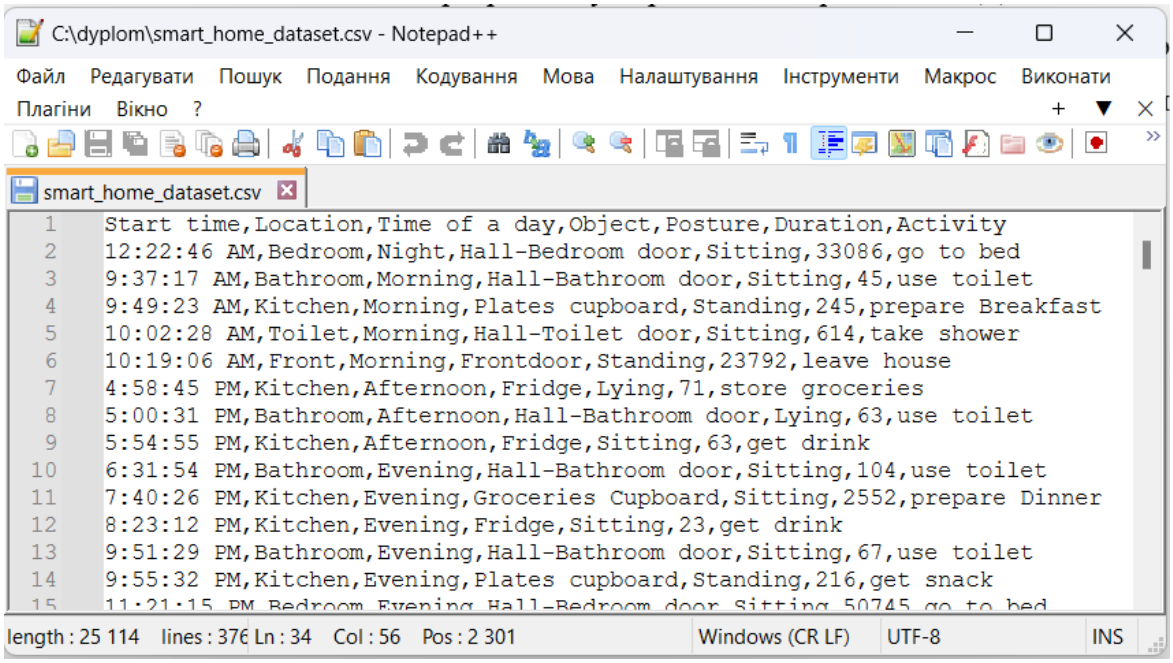


Рисунок 4.6 – Знімок екрану з даними, готовими до тренування моделі

Для забезпечення ефективного тренування моделі виявлення вторгнень у застосунку була створена окрема категорія під назвою «Виявлення поведінкових патернів», що зображено на рис. 4.7.

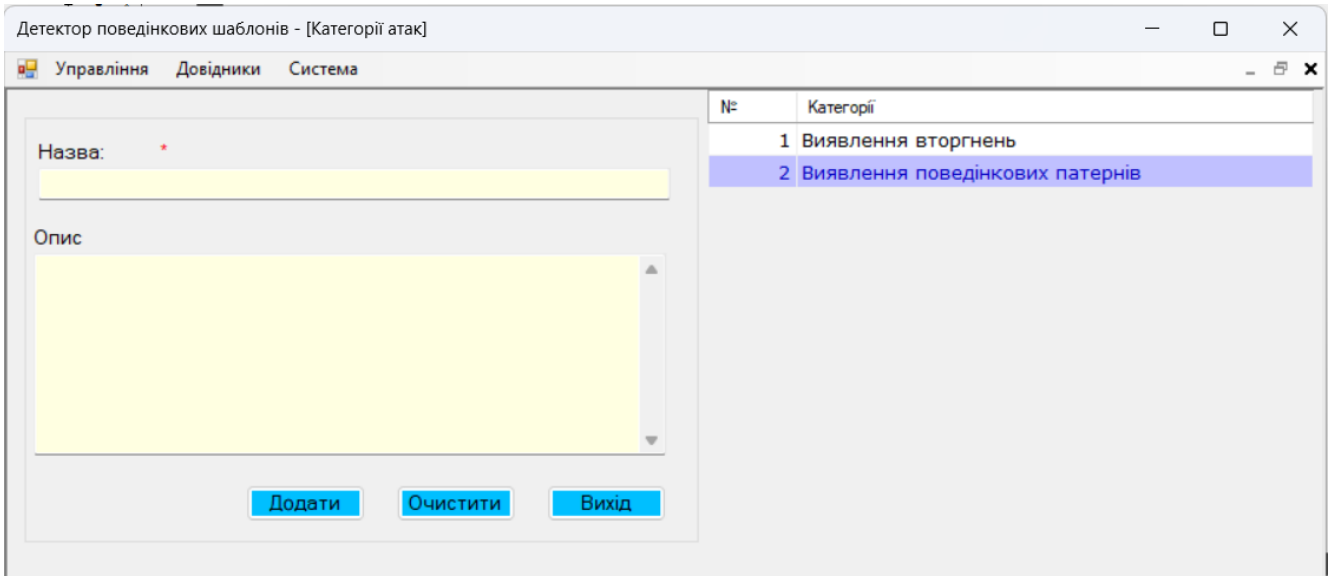


Рисунок 4.7 – Визначення категорії для навчання моделі

Для реалізації процесу тренування цієї моделі використовувався спеціально підготовлений датасет. У програмному середовищі через меню «Довідники» → «Навчання моделей» було вибрано зазначену категорію. Надалі, після вибору підготовленого датасету в діалоговому вікні, тренування моделі стартувало автоматично. Процес і результати навчання моделі можна побачити на рис. 4.8.

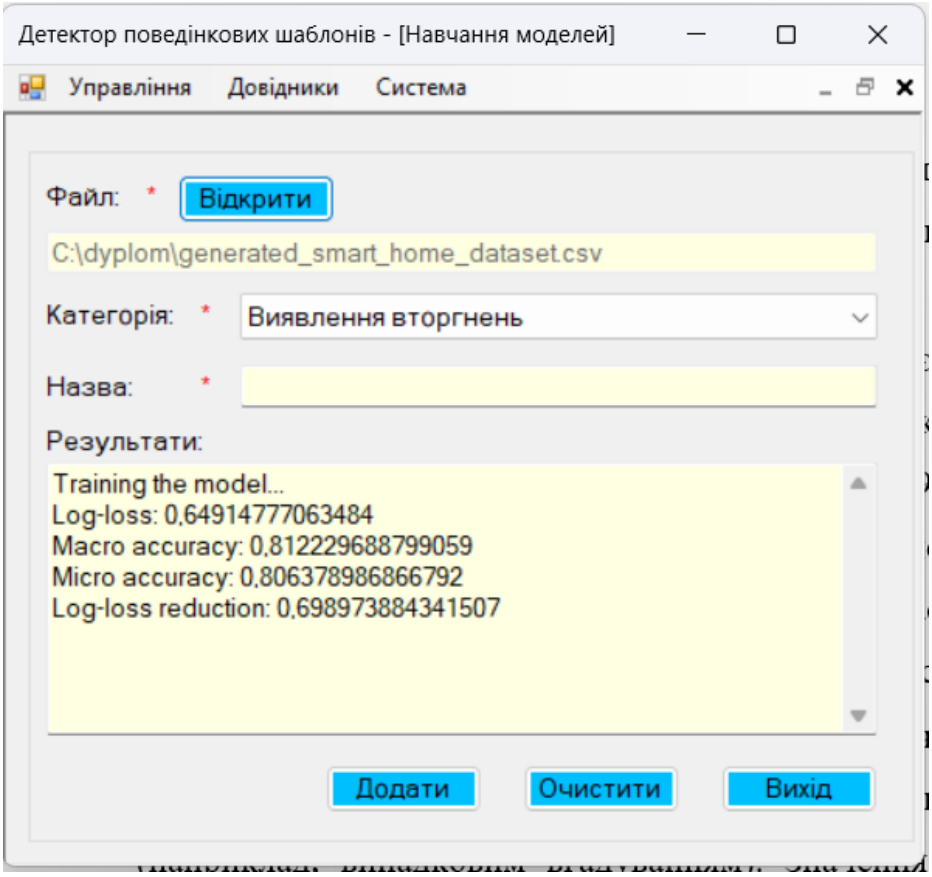


Рисунок 4.8 – Відображення результату тренування моделі

Аналіз метрик моделі дозволяє оцінити її ефективність та точність у виконанні завдань класифікації. Нижче представлено детальний аналіз отриманих метрик:

– Log-loss (логарифмічна втрата) є метрикою, яка вимірює ефективність класифікаційної моделі, враховуючи ймовірність правильних передбачень. Значення Log-loss для нашої моделі становить 0,64914777063484. Нижче значення Log-loss вказує на кращу модель, оскільки ця метрика штрафує модель за невірні прогнози з високою впевненістю. Значення 0,649 є досить прийнятним, що вказує на те, що модель робить відносно точні прогнози, але ще має простір для поліпшення;

– Macro accuracy (макро точність) розраховується як середнє значення точностей для кожного класу, незалежно від кількості прикладів у кожному класі. Значення Macro accuracy для нашої моделі становить 0,812229688799059. Це означає, що в середньому по всіх класах модель класифікує 81,2% прикладів правильно. Це хороше значення, яке свідчить про те, що модель демонструє стабільну продуктивність по всіх класах, включаючи ті, що можуть бути менш представлені в даних;

– Micro accuracy (мікро точність) враховує загальну кількість правильно класифікованих прикладів відносно загальної кількості прикладів. Значення Micro accuracy для нашої моделі становить 0,806378986866792. Це означає, що 80,6% всіх передбачень моделі є правильними. Це значення є дещо нижчим за Macro accuracy, що може вказувати на те, що деякі класи можуть мати більше ваги в даних, але модель все ще демонструє високу точність загалом;

– Log-loss reduction (редукція логарифмічних втрат) є метрикою, яка показує, наскільки модель покращилася порівняно з початковою моделлю (наприклад, випадковим вгадуванням). Значення Log-loss reduction для нашої моделі становить 0,698973884341507. Це означає, що модель значно покращилася порівняно з базовою лінією, знижуючи логарифмічну втрату на майже 70%. Це свідчить про те, що модель добре навчається і робить прогнози набагато краще, ніж випадкове вгадування.

Загальний аналіз метрик показує, що модель демонструє високу точність і ефективність у виконанні завдань класифікації. Log-loss свідчить про прийнятну якість передбачень, Macro і Micro accuracy показують високу точність як по окремих класах, так і загалом, а Log-loss reduction вказує на значне покращення моделі порівняно з базовою лінією. Це дозволяє зробити висновок, що модель є досить надійною для використання в реальних умовах, але все ще може бути вдосконалена для досягнення ще кращих результатів.

Після цього навчену модель було збережено у системі, для проведення подальшого експериментального дослідження (рис. 4.9).

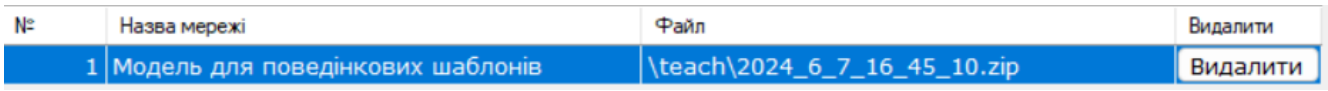


Рисунок 4.9 –Збереження моделі

4.3 Верифікація методу через різні сценарії виявлення вторгнень

Для тестування збереженої моделі виявлення вторгнення в систему «розумний дім» користувачам слід перейти через меню програми «Управління» до розділу «Детектування вторгнення». У відкритому вікні можна провести симуляцію можливих сценаріїв вторгнення, використовуючи випадаючий список для вибору категорії, за якою було здійснено навчання моделі з використанням методу БФГШ.

Процес прогнозування потенційних загроз у системі «Розумний дім» складається з наступних кроків:

– генерація даних. Розроблено спеціалізований клас (SmartHomeDataGenerator), що відповідає за створення даних, імітуючих повсякденні події у домі та потенційні вторгнення. Для цього використовуються параметри, такі як локація (наприклад, спальня чи ванна кімната), час доби, об'єкти (наприклад, двері, вікна), пози (сидячи, стоячи) та різноманітні дії (наприклад, «йти спати», «віджати двері»);

– прогнозування за допомогою моделі. Після генерації, дані передаються у модель, яка аналізує кожну подію на предмет можливості вторгнення.

– аналіз результатів прогнозування. Результати, отримані від моделі, ретельно аналізуються, щоб визначити, чи відбулося потенційне вторгнення. Оцінюється відповідність діяльності до контексту та можливість її загрози;

– візуалізація та звітування. Результати симуляції відображаються в спеціальному текстовому полі програми, з інформацією про кожен аспект згенерованих даних, включаючи час, місцеположення, час доби, діяльність, а також оцінку потенційного вторгнення.

Основна мета цієї симуляції полягає в тому, щоб перевірити здатність розробленої нейронної мережі ефективно розрізняти звичайні події домашнього життя від потенційних загроз. Це не тільки демонструє ефективність розробленої системи, але й дає змогу виявити та усунути можливі недоліки в моделі перед її реальним застосуванням.

Для оцінки точності моделі, навченої за допомогою методу БФГШ, були створені різні потенційні сценарії вторгнення за допомогою спеціально розробленого генератора сценаріїв. Кожен із сценаріїв піддався ретельному аналізу. Рис. 4.10 демонструє перший із серії згенерованих сценаріїв.

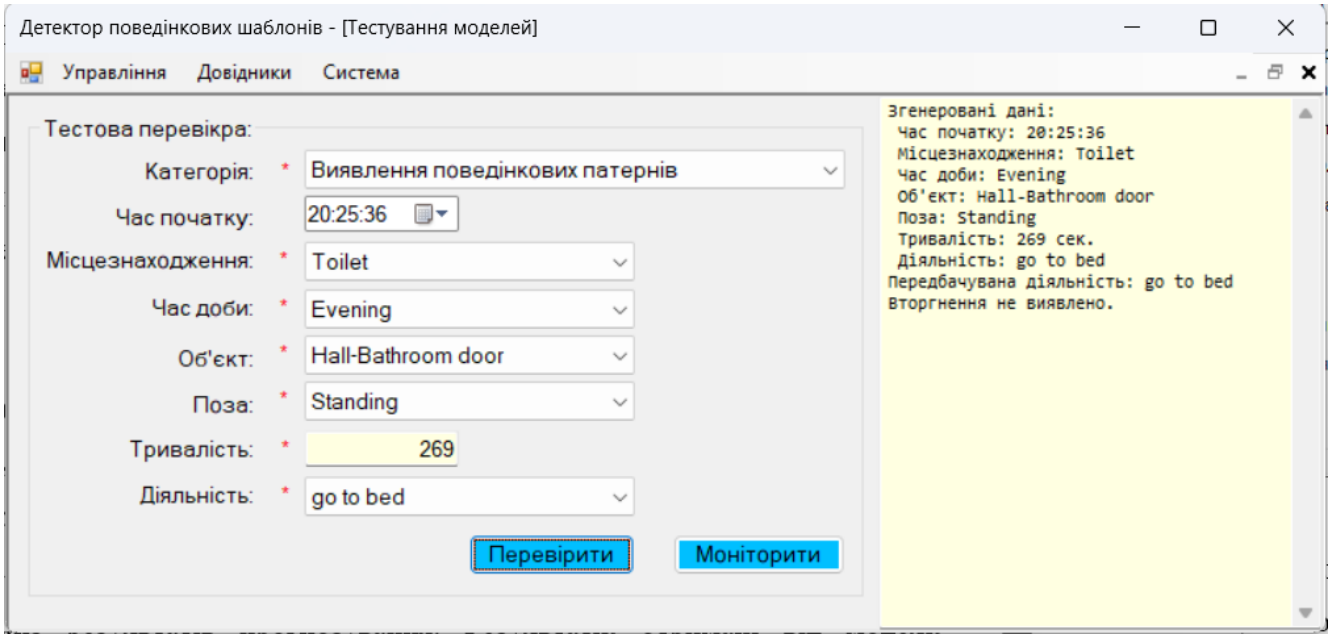


Рисунок 4.10 – Результат генерації 1-го сценарію

У даному сценарії були згенеровані наступні дані: час початку – 20:25:36, місцезнаходження – туалет, час доби – вечір, об'єкт – двері між холом та ванною кімнатою, поза – стоячи, тривалість – 269 секунд, діяльність – "йти спати". Модель прогнозування визначила передбачувану діяльність як "йти спати", що відповідає згенерованій діяльності. На основі аналізу результатів, модель не виявила вторгнення.

Результати даного сценарію показують, що модель коректно визначила діяльність користувача, співставивши її із згенерованими даними. Система не виявила аномальної поведінки або потенційного вторгнення, оскільки діяльність "йти спати" є типовою для вечірнього часу та відповідного контексту (місцезнаходження туалет, стоячи біля дверей між холом та ванною кімнатою). Тривалість у 269 секунд також не викликає підозр, оскільки це є типовою тривалістю для підготовки до сну.

На рис. 4.11 зображено скріншот результат виконання 2-го сценарію.

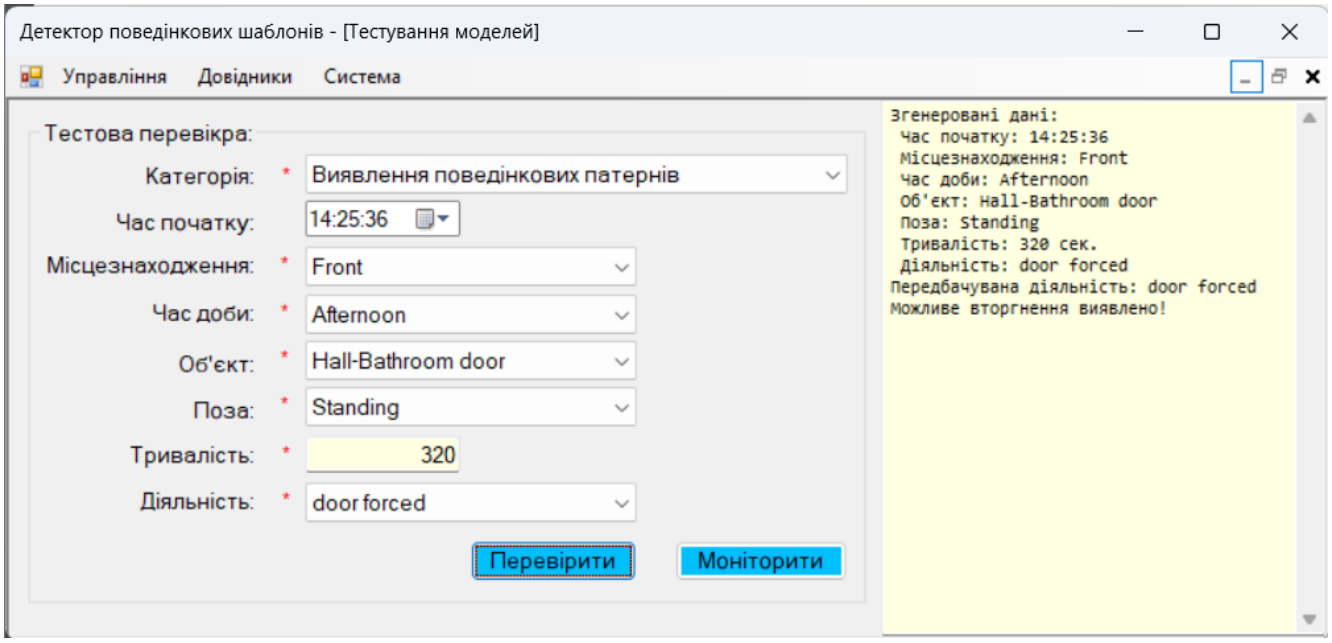


Рисунок 4.11 – Результат генерації 2-го сценарію

У даному сценарії були згенеровані наступні дані: час початку – 14:46:44, місцезнаходження – передня частина будинку, час доби – післяобідній, об'єкт – двері між холом та туалетом, поза – стоячи, тривалість – 320 секунд, діяльність – "віджати двері". Модель прогнозування визначила передбачувану діяльність як

"віджати двері", що відповідає згенерованій діяльності. На основі аналізу результатів, модель виявила можливе вторгнення.

Результати цього сценарію показують, що модель коректно визначила підозрілу діяльність користувача. Діяльність "віджати двері" є нетиповою для нормальної поведінки в будинку, особливо враховуючи місцезнаходження в передній частині будинку та час доби – післяобідній. Тривалість у 320 секунд також підкреслює незвичність цієї діяльності, оскільки вона перевищує середній час, необхідний для звичайних дій біля дверей.

Модель успішно ідентифікувала підозрілу діяльність і правильно визначила її як можливе вторгнення. Це підтверджує ефективність моделі у виявленні аномальної поведінки, що може свідчити про загрозу безпеці. Такий результат підкреслює здатність моделі забезпечувати своєчасне виявлення потенційних загроз і підвищує рівень безпеки в системі розумного будинку.

4.4 Перспективи подальших досліджень у цій області

Розроблена захисна система на основі методу БФГШ відкриває нові перспективи для покращення захисту та комфорту у повсякденному житті. Однак, для повної реалізації потенціалу цієї системи важливо зосередитись на її подальшому вдосконаленні та розширенні. У цьому контексті були окреслені декілька ключових напрямів для розвитку, які можуть суттєво підвищити ефективність та адаптивність системи:

– інтеграція з додатковими датчиками та інформаційними джерелами. Включення додаткових датчиків та інших джерел даних може зміцнити здатність системи ідентифікувати аномалії та прогнозувати потенційні загрози. Наприклад, інтеграція з системами відеонагляду, аудіосенсорами, чи датчиками якості повітря може збагатити аналітичні моделі новим контекстом;

– розробка гібридних моделей штучного інтелекту. Комбінація методу БФГШ з іншими алгоритмами штучного інтелекту, наприклад, згортковими

нейронними мережами (CNN) для обробки візуальних даних, може покращити точність та ефективність системи;

– адаптивне навчання та самонавчання. Розробка механізмів, які дозволяють системі постійно адаптуватися та навчатися на основі нових даних та змін у поведінці користувачів, зробить систему більш гнучкою та ефективною у виявленні нових видів загроз;

– забезпечення приватності та безпеки даних. Оскільки системи розумного будинку збирають значні обсяги особистих даних, надзвичайно важливо вдосконалити заходи захисту цих даних. Впровадження передових технологій шифрування та анонімізації може значно підвищити рівень довіри та безпеки системи;

– інтеграція з екосистемами розумного будинку. Інтеграція цієї системи з іншими компонентами розумного будинку, такими як системи управління освітленням та опаленням, може створити більш інтегровану та автоматизовану систему безпеки;

– використання хмарних обчислень та великих даних. Впровадження хмарних платформ може значно підвищити швидкість і ефективність системи, дозволяючи виконувати більш складні алгоритми штучного інтелекту, що потребують значних обчислювальних ресурсів;

– подальші дослідження у галузі кібербезпеки. Продовження досліджень у цій сфері, зокрема з акцентом на використання штучного інтелекту, може виявити нові можливості та методи захисту від кіберзагроз.

Висновки до розділу 4

У рамках даного розділу було проведено комплексне експериментальне дослідження та оцінку ефективності розробленого методу виявлення вторгнень у розумні будинки. На початковому етапі було налаштовано експериментальне середовище, включаючи визначення порту Arduino UNO в диспетчері пристроїв

Windows, вибір послідовного порту для з'єднання, компіляцію програми в Arduino IDE та завантаження її на мікроконтролер. Для оцінки ефективності системи було використано набір даних з платформи Kaggle, після чого модель пройшла навчання і було отримано наступні метрики: логарифмічні втрати – 0,65, макро точність – 0,81, мікро точність – 0,81, редукція логарифмічних втрат – 0,7.

Верифікація методу відбулася через аналіз різних сценаріїв виявлення вторгнень, де було згенеровано та проаналізовано два сценарії. Модель успішно ідентифікувала підозрілу діяльність і правильно визначила її як можливе вторгнення в одному із сценаріїв, що підтверджує її здатність виявляти аномалії у поведінкових патернах.

Перспективи подальших досліджень у цій області включають інтеграцію з додатковими датчиками та інформаційними джерелами, розробку гібридних моделей штучного інтелекту, адаптивне навчання та самонавчання, забезпечення приватності та безпеки даних, інтеграцію з екосистемами розумного будинку, використання хмарних обчислень та великих даних, створення співтовариства та обмін даними, а також подальші дослідження у галузі кібербезпеки. Проведені експерименти та отримані результати дозволяють впевнено стверджувати, що розроблена система є ефективною для виявлення вторгнень у розумні будинки і має великий потенціал для подальшого розвитку та вдосконалення.

ВИСНОВКИ

У рамках даної роботи було проведено комплексне дослідження та розробку системи виявлення вторгнень у розумні будинки на основі аналізу поведінкових патернів. В процесі теоретичного огляду було розглянуто концепції розумних будинків, їх централізовану та децентралізовану архітектуру, а також ключові компоненти інтелектуальних систем. Було детально проаналізовано існуючі методи виявлення вторгнень, включаючи розумне відеоспостереження, системи розумного освітлення, інтелектуальні замки безпеки та методи інтеграції із персональними асистентами. Крім того, було вивчено методи аналізу поведінкових даних з використанням машинного та глибокого навчання, а також оцінено ризики та потенційні загрози для розумних домівок.

В процесі проектування та розробки методу виявлення вторгнень було проаналізовано потенційні методи машинного навчання, зокрема Бройдена-Флетчера-Гольдфарба-Шанно (БФГШ), метод опорних векторів (МОВ) та метод випадкового лісу (МВЛ). Після проведення порівняльного аналізу було обрано метод БФГШ для реалізації. Було розроблено математичну модель виявлення вторгнень та алгоритми для виявлення аномалій у поведінці жителів. Вибрано технологічні рішення для реалізації методу, серед яких C# виявився найбільш оптимальним. Реалізовано прототип системи, який включає трьохрівневу модель із детальним описом кожного рівня та побудованими діаграмами класів системи.

У процесі вибору компонентів та розробки загальної архітектури системи було проаналізовано вимоги до апаратного забезпечення, зокрема контролерів, сенсорів, серверів, мережевої інфраструктури та джерел безперебійного живлення. Проведено огляд та вибір контролера для системи, де було обрано Arduino UNO. Вибрано комплектуючі для системи, серед яких датчик pir-d203s, датчик температури DHT22, температурний магнітний датчик Reed module KY-025, фоторезистор KY-018 та мікрофонний модуль MAX9814. Розроблено та

частково описано скетч для роботи системи, а також побудовано схему взаємодії компонентів системи.

Під час експериментального дослідження та оцінки ефективності методу було налаштовано експериментальне середовище, включаючи визначення порту Arduino UNO, вибір послідовного порту, компіляцію програми в Arduino IDE та завантаження її на мікроконтролер. Для оцінки ефективності системи було зібрано дані з платформи Kaggle, проведено навчання моделі та отримано такі метрики: логарифмічні втрати 0,65, макро точність 0,81, мікро точність 0,81, редукція логарифмічних втрат 0,7. Було збережено модель та проведено її верифікацію через аналіз різних сценаріїв виявлення вторгнень. Модель успішно ідентифікувала підозрілу діяльність та правильно визначила її як можливе вторгнення в одному з двох сценаріїв, що підтверджує її здатність виявляти аномалії у поведінкових патернах.

Перспективи подальших досліджень у цій області включають інтеграцію з додатковими датчиками та інформаційними джерелами, розробку гібридних моделей штучного інтелекту, адаптивне навчання та самонавчання, забезпечення приватності та безпеки даних, інтеграцію з екосистемами розумного будинку, використання хмарних обчислень та великих даних, створення співтовариства та обмін даними, а також подальші дослідження у галузі кібербезпеки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Дудинець Дмитро. Інтернет речі та розумний дім. TTSIIT, 2024, 81 с.

2. Kotsiubivska Kateryna; Prisych Vladyslav; Yavorskyi Oleksandr. Упровадження технологій інтернету речей під час створення системи «Розумний дім». Цифрова платформа: інформаційні технології в соціокультурній сфері, 2019. – 143с.

3. Younis Mohammed Issam; Hussein Taif Fadhil. Design and Implementation of a Contactless Smart House Network System. International Journal of Electrical & Computer Engineering , 2018. –68 p.

4. Слюсар В. І.; Слюсар І. І.; Колодій В. В. Система безпеки в концепції «розумний дім». ПолтНТУ. 2018. – 76с.

5. Hayes-Roth Barbara. An architecture for adaptive intelligent systems. Artificial intelligence, 1995. – 365 p.

6. Milan Giulia, et al. RL-IoT: reinforcement learning to interact with IoT devices. In: 2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS). IEEE, 2021. –68 p.

7. Bar-Joseph Ziv, et al. Computational discovery of gene modules and regulatory networks. Nature biotechnology, 2003. – 1342 p.

8. Madakam Somayya; Ramaswamy Ramya; Tripathi Siddharth. Internet of Things (IoT): A literature review. Journal of Computer and Communications, 2015. – 173 p.

9. Xia Feng. QoS challenges and opportunities in wireless sensor/actuator networks. Sensors, 2008. – 1110 p.

10. Aliahmadi Alireza; Nozari Hamed; Ghahremani-Nahr Javid. AIoT-based sustainable smart supply chain framework. International journal of innovation in management, economics and social sciences, 2022. –28 p.

11. Lesani Fatemeh Sadat; Fotouhi Ghazvini Faranak; Amirkhani Hossein. Smart home resident identification based on behavioral patterns using ambient sensors. Personal and Ubiquitous Computing, 2021. – 155 p.

12. Lee Hyun-Soo; Park Sung-Jun; Jung Ji-Yea. A study on developing a smart home service based on the behavior patterns of the elderly. Journal of the Architectural Institute of Korea Planning & Design, 2012. – 159 p.

13. Zhang Junge; Shan Yanhu; Huang Kaiqi. ISEE Smart Home (ISH): Smart video analysis for home security. Neurocomputing, 2015. – 149 p.

14. Barghi Armin, et al. Intelligent lighting control with LEDS for smart home. In: 2014 Smart Grid Conference (SGC). IEEE, 2014. – 58 p.

15. Jose Arun Cyril; Malekian Reza. Improving smart home security: Integrating logical sensing into smart home. IEEE Sensors Journal, 2017 .– 428 p.

16. Edu Jide S.; Such Jose M.; Suarez-Tangil Guillermo. Smart home personal assistants: a security and privacy review. ACM Computing Surveys, 2020. –36 p.

17. Lesani Fatemeh Sadat; Fotouhi Ghazvini Faranak; Amirkhani Hossein. Smart home resident identification based on behavioral patterns using ambient sensors. Personal and Ubiquitous Computing, 2021. – 162 p.

18. Русаков Микита Андрійович. Розробка модернізованої автоматизованої система керування розумним будинком.2023. – 122с.

19. Muneer Amgad, et al. Short term residential load forecasting using long short-term memory recurrent neural network. International Journal of Electrical & Computer Engineering (2088-8708), 2022. – 125 p.

20. Kattenborn Teja, et al. Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. ISPRS journal of photogrammetry and remote sensing, 2021. – 173 p.

21. Sherstinsky Alex. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 2020. – 404p.

22. Gill Philip E.; Runnoe Jeb H. On recent developments in BFGS methods for unconstrained optimization. Center for Computational Mathematics Report CCoM, 2013.p. 228 p.

23. Nawi Nazri Mohd; Ransing Meghana R.; Ransing Rajesh S. An improved learning algorithm based on the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method for back propagation neural networks. In: Sixth International Conference on Intelligent Systems Design and Applications. IEEE, 2006. – 157 p.

24. Tax David Mj; Duin Robert PW. Data domain description using support vectors. In: ESANN. 2019. – 256 p.

25. Schölkopf Bernhard, et al. Support vector method for novelty detection. Advances in neural information processing systems, 2018. – 12 p.

26. Rigatti Steven J. Random forest. Journal of Insurance Medicine, 2017. – 85 p.

27. Reis Itamar; Baron Dalya; Shahaf Sahar. Probabilistic random forest: A machine learning algorithm for noisy data sets. The Astronomical Journal, 2018 – 157.p.

28. Python : веб-сайт. URL : https://www.python.org/ (дата звернення 06.06.2024).

29. Java : веб-сайт. URL:https://dou.ua/lenta/articles/how-to-learn-java/ (дата звернення 06.06.2024).

30. C# : веб-сайт. URL: https://beetroot.academy/blog/courses/what-is-C (дата звернення 06.06.2024).

31. Hadwan H. Hamid; Reddy Y. P. Smart home control by using raspberry pi and arduino uno. Int. J. Adv. Res. Comput. Commun. Eng, 2016. – 288 p.

32. Шварц Марко. Будівництво розумних будинків за допомогою Raspberry Pi Zero . Packt Publishing Ltd, 2016. – 68с.

33. Khoo Y. Z., et al. STM32-based IoT Monitoring System for an Indoor Plant. International Journal of Intelligent Systems and Applications in Engineering, 2022. – 371 p.

34. Dataset for Smart Home. URL: https://www.kaggle.com/datasets/shegguy87/dataset-for-smart-home (дата звернення 06.06.2024).

ДОДАТОК А. Скетч для управління контролером

```
#include <DHT.h>
#include <SPI.h>
#include <Ethernet.h>

// Налаштування датчиків
#define DHTPIN 2          // Пін для DHT22
#define DHTTYPE DHT22     // Тип DHT22
DHT dht(DHTPIN, DHTTYPE);

#define PIRPIN 3          // Пін для PIR датчика
#define REEDPIN 4         // Пін для Reed модуля
#define LDRPIN A0         // Пін для фоторезистора
#define MICPIN A1         // Пін для мікрофонного модуля

// Ethernet налаштування
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };  // MAC-адреса Ethernet-модуля
IPAddress server(192, 168, 1, 100);                   // IP-адреса сервера бази даних
EthernetClient client;

void setup() {
  Serial.begin(9600);
  dht.begin();

  // Ініціалізація Ethernet
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    while (true);
  }

  // Підключення до сервера
  if (client.connect(server, 80)) {
    Serial.println("Connected to server");
  } else {
    Serial.println("Connection to server failed");
  }

  pinMode(PIRPIN, INPUT);
  pinMode(REEDPIN, INPUT);
}

void loop() {
  float humidity = readHumidity();
  float temperature = readTemperature();
  int pirState = readPIR();
  int reedState = readReed();
  int lightLevel = readLight();
  int soundLevel = readSound();
```

```
  sendDataToServer(temperature, humidity, pirState, reedState, lightLevel, soundLevel);

  delay(10000);
}

// Функція зчитування вологості з DHT22
float readHumidity() {
  float h = dht.readHumidity();
  if (isnan(h)) {
    Serial.println("Failed to read humidity from DHT sensor!");
    return -1;
  }
  return h;
}

// Функція зчитування температури з DHT22
float readTemperature() {
  float t = dht.readTemperature();
  if (isnan(t)) {
    Serial.println("Failed to read temperature from DHT sensor!");
    return -1;
  }
  return t;
}

// Функція зчитування стану PIR датчика
int readPIR() {
  return digitalRead(PIRPIN);
}

// Функція зчитування стану Reed модуля
int readReed() {
  return digitalRead(REEDPIN);
}

// Функція зчитування рівня освітленості з фоторезистора
int readLight() {
  return analogRead(LDRPIN);
}

// Функція зчитування рівня звуку з мікрофонного модуля
int readSound() {
  return analogRead(MICPIN);
}

// Функція відправки даних на сервер
void sendDataToServer(float temperature, float humidity, int pir, int reed, int light, int sound) {
  String data = "GET /insert_data.php?";
  data += "temperature=" + String(temperature);
  data += "&humidity=" + String(humidity);
  data += "&pir=" + String(pir);
  data += "&reed=" + String(reed);
```

```
  data += "&light=" + String(light);
  data += "&sound=" + String(sound);
  data += " HTTP/1.1\r\n";
  data += "Host: 192.168.1.100\r\n";
  data += "Connection: close\r\n\r\n";

  if (client.connect(server, 80)) {
    client.print(data);
    Serial.println("Data sent to server");
  } else {
    Serial.println("Connection to server failed");
  }

  client.stop();
}
```

ДОДАТОК Б. Лістинги програми

Лістинг 1. Код класу «SmartHomeSensorDataProvider»

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

namespace GuardApp.Providers {
  internal class SmartHomeSensorDataProvider {
    private string _connectionString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    // Метод для додавання даних сенсорів у базу даних
    public void AddSensorData(HomeSecuritySensorData data) {
      string sqlQuery = "INSERT INTO HomeSecuritySensorData (Timestamp, MotionDetected,
Temperature, DoorOpen, WindowOpen, LightIntensity, NoiseDetected, Humidity, CameraFeed,
AirQuality, RoomName) Values(@Timestamp, @MotionDetected, @Temperature, @DoorOpen,
@WindowOpen, @LightIntensity, @NoiseDetected, @Humidity, @CameraFeed, @AirQuality,
@RoomName)";

      using (SqlConnection connection = new SqlConnection(_connectionString)) {
        using (SqlCommand command = new SqlCommand(sqlQuery, connection)) {
          command.CommandType = CommandType.Text;
          command.Parameters.AddWithValue("@Timestamp", data.Timestamp);
          command.Parameters.AddWithValue("@MotionDetected", data.MotionDetected);
          command.Parameters.AddWithValue("@Temperature", data.Temperature);
          command.Parameters.AddWithValue("@DoorOpen", data.DoorOpen);
          command.Parameters.AddWithValue("@WindowOpen", data.WindowOpen);
          command.Parameters.AddWithValue("@LightIntensity", data.LightIntensity);
          command.Parameters.AddWithValue("@NoiseDetected", data.NoiseDetected);
          command.Parameters.AddWithValue("@Humidity", data.Humidity);
          command.Parameters.AddWithValue("@CameraFeed", data.CameraFeed ??
(object)DBNull.Value); // Handle null values
          command.Parameters.AddWithValue("@AirQuality", data.AirQuality);
          command.Parameters.AddWithValue("@RoomName", data.RoomName);
          connection.Open();
          command.ExecuteNonQuery();
          connection.Close();
        }
      }
    }

    // Метод для отримання всіх даних сенсорів з бази даних
    public List<HomeSecuritySensorData> GetAllSensorData() {
      string sqlQuery = "SELECT * FROM HomeSecuritySensorData ORDER BY Timestamp DESC";
```

```
    List<HomeSecuritySensorData> allSensorDataList = new List<HomeSecuritySensorData>();
    using (SqlConnection connection = new SqlConnection(_connectionString)) {
     using (SqlCommand command = new SqlCommand(sqlQuery, connection)) {
      connection.Open();
      using (SqlDataReader reader = command.ExecuteReader()) {
       while (reader.Read()) {
        HomeSecuritySensorData sensorData = new HomeSecuritySensorData();
        // Припускаючи, що клас HomeSecuritySensorData має відповідний конструктор
        sensorData.Timestamp = reader.GetDateTime(reader.GetOrdinal("Timestamp"));
        sensorData.MotionDetected = reader.GetBoolean(reader.GetOrdinal("MotionDetected"));
        sensorData.Temperature = reader.GetDouble(reader.GetOrdinal("Temperature"));
        sensorData.DoorOpen = reader.GetBoolean(reader.GetOrdinal("DoorOpen"));
        sensorData.WindowOpen = reader.GetBoolean(reader.GetOrdinal("WindowOpen"));
        sensorData.LightIntensity = reader.GetDouble(reader.GetOrdinal("LightIntensity"));
        sensorData.NoiseDetected = reader.GetBoolean(reader.GetOrdinal("NoiseDetected"));
        sensorData.Humidity = reader.GetDouble(reader.GetOrdinal("Humidity"));
        sensorData.CameraFeed = reader.IsDBNull(reader.GetOrdinal("CameraFeed")) ? null :
reader.GetString(reader.GetOrdinal("CameraFeed"));
        sensorData.AirQuality = reader.GetDouble(reader.GetOrdinal("AirQuality"));
        sensorData.RoomName = reader.GetString(reader.GetOrdinal("RoomName"));
        allSensorDataList.Add(sensorData);
       }
      }
      connection.Close();
     }
    }

    return allSensorDataList;
   }

   // Метод для вибору конкретних даних сенсора за ідентифікатором
   public HomeSecuritySensorData GetSensorDataById(int id) {
    string sqlQuery = "SELECT * FROM HomeSecuritySensorData WHERE Id=@Id";

    HomeSecuritySensorData sensorData = null;
    using (SqlConnection connection = new SqlConnection(_connectionString)) {
     using (SqlCommand command = new SqlCommand(sqlQuery, connection)) {
      command.Parameters.AddWithValue("@Id", id);
      connection.Open();
      using (SqlDataReader reader = command.ExecuteReader()) {
       if (reader.Read()) {
        sensorData = new HomeSecuritySensorData {
         // Припускається, що клас HomeSecuritySensorData має відповідний конструктор або
публічні сеттери
         Timestamp = reader.GetDateTime(reader.GetOrdinal("Timestamp")),
         MotionDetected = reader.GetBoolean(reader.GetOrdinal("MotionDetected")),
         Temperature = reader.GetDouble(reader.GetOrdinal("Temperature")),
         DoorOpen = reader.GetBoolean(reader.GetOrdinal("DoorOpen")),
         WindowOpen = reader.GetBoolean(reader.GetOrdinal("WindowOpen")),
         LightIntensity = reader.GetDouble(reader.GetOrdinal("LightIntensity")),
         NoiseDetected = reader.GetBoolean(reader.GetOrdinal("NoiseDetected")),
         Humidity = reader.GetDouble(reader.GetOrdinal("Humidity")),
```

```
            CameraFeed = reader.IsDBNull(reader.GetOrdinal("CameraFeed")) ? null :
reader.GetString(reader.GetOrdinal("CameraFeed")),
            AirQuality = reader.GetDouble(reader.GetOrdinal("AirQuality")),
            RoomName = reader.GetString(reader.GetOrdinal("RoomName"))
          };
        }
       }
       connection.Close();
      }
     }
     return sensorData;
    }

    // Метод для оновлення даних сенсора
    public void UpdateSensorData(HomeSecuritySensorData data) {
     string sqlQuery = @"
            UPDATE HomeSecuritySensorData
            SET
               Timestamp = @Timestamp,
               MotionDetected = @MotionDetected,
               Temperature = @Temperature,
               DoorOpen = @DoorOpen,
               WindowOpen = @WindowOpen,
               LightIntensity = @LightIntensity,
               NoiseDetected = @NoiseDetected,
               Humidity = @Humidity,
               CameraFeed = @CameraFeed,
               AirQuality = @AirQuality,
               RoomName = @RoomName
            WHERE Id = @Id";

     using (SqlConnection connection = new SqlConnection(_connectionString)) {
      using (SqlCommand command = new SqlCommand(sqlQuery, connection)) {
       command.CommandType = CommandType.Text;
       command.Parameters.AddWithValue("@Timestamp", data.Timestamp);
       command.Parameters.AddWithValue("@MotionDetected", data.MotionDetected);
       command.Parameters.AddWithValue("@Temperature", data.Temperature);
       command.Parameters.AddWithValue("@DoorOpen", data.DoorOpen);
       command.Parameters.AddWithValue("@WindowOpen", data.WindowOpen);
       command.Parameters.AddWithValue("@LightIntensity", data.LightIntensity);
       command.Parameters.AddWithValue("@NoiseDetected", data.NoiseDetected);
       command.Parameters.AddWithValue("@Humidity", data.Humidity);
       command.Parameters.AddWithValue("@CameraFeed", (object)data.CameraFeed ??
DBNull.Value); // Use DBNull for null values
       command.Parameters.AddWithValue("@AirQuality", data.AirQuality);
       command.Parameters.AddWithValue("@RoomName", data.RoomName);
       command.Parameters.AddWithValue("@Id", data.Id);

       connection.Open();
       command.ExecuteNonQuery();
       connection.Close();
      }
```

```
    }
   }

   // Метод для видалення даних сенсора за ідентифікатором
   public void DeleteSensorDataById(int id) {
    string sqlQuery = "DELETE FROM HomeSecuritySensorData WHERE Id=@Id";
    using (SqlConnection connection = new SqlConnection(_connectionString)) {
     using (SqlCommand command = new SqlCommand(sqlQuery, connection)) {
      command.Parameters.AddWithValue("@Id", id);
      connection.Open();
      command.ExecuteNonQuery();
      connection.Close();
     }
    }
   }

  }
}

public class HomeSecuritySensorData {
  public int Id { get; set; }
  public DateTime Timestamp { get; set; }
  public bool MotionDetected { get; set; }
  public double Temperature { get; set; }
  public bool DoorOpen { get; set; }
  public bool WindowOpen { get; set; }
  public double LightIntensity { get; set; }
  public bool NoiseDetected { get; set; }
  public double Humidity { get; set; }
  public string CameraFeed { get; set; }
  public double AirQuality { get; set; }
  public string RoomName { get; set; }

  public HomeSecuritySensorData() { }

  // Конструктор класу
  public HomeSecuritySensorData(DateTime timestamp, bool motionDetected, double temperature,
                bool doorOpen, bool windowOpen, double lightIntensity,
                bool noiseDetected, double humidity, string cameraFeed,
                double airQuality, string roomName) {
   Timestamp = timestamp;
   MotionDetected = motionDetected;
   Temperature = temperature;
   DoorOpen = doorOpen;
   WindowOpen = windowOpen;
   LightIntensity = lightIntensity;
   NoiseDetected = noiseDetected;
   Humidity = humidity;
   CameraFeed = cameraFeed;
   AirQuality = airQuality;
   RoomName = roomName;
  }
```

```
}
```

Лістинг 2. Код класу «PredictTestModelForm»

```
using HomeGuard.AppCode;
using HomeGuard.Providers;
using Microsoft.ML;
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 HomeGuard.Forms.Controls {
   public partial class PredictModelForm : Form {
      private DataValidator _dataValidator = new DataValidator();
      private NeuralModel _selectedModel = new NeuralModel();
      private MLContext _mlContext = new MLContext();
      private PredictionEngine<HomeSensorData, SensorPrediction> _predictionEngine;
      private ModelProvider _modelProvider = new ModelProvider();

      private CategoryProvider _categoryProvider = new CategoryProvider();
      private List<Category> _categoryList = new List<Category>();
      private bool _isCategoryLoaded = false;

      private SensorDataGenerator _sensorDataGenerator = new SensorDataGenerator();

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

      private void LoadAllData() {
         LocationComboBox.SelectedIndex = 0;
         TimeOfDayComboBox.SelectedIndex = 0;
         ObjectComboBox.SelectedIndex = 0;
         PostureComboBox.SelectedIndex = 0;
         ActivityComboBox.SelectedIndex = 0;

         _categoryList = _categoryProvider.GetAllCategories();
         CategoryComboBox.DataSource = _categoryList;
         CategoryComboBox.ValueMember = "CategoryId";
         CategoryComboBox.DisplayMember = "CategoryName";
         _isCategoryLoaded = true;
         CategoryComboBox_SelectedValueChanged(CategoryComboBox, EventArgs.Empty);
      }

      private void LoadModelData(string filePath) {
         string modelPath = Application.StartupPath + filePath;
```

```
        DataViewSchema modelSchema;

        // Load the trained model
        ITransformer model = _mlContext.Model.Load(modelPath, out modelSchema);

        // Create a prediction engine
        _predictionEngine = _mlContext.Model.CreatePredictionEngine<HomeSensorData,
SensorPrediction>(model);
    }

    private void MonitorButton_Click(object sender, EventArgs e) {
        if (monitorTimer.Enabled) {
            monitorTimer.Enabled = false;
            MonitorButton.Text = "Start Monitoring";
        } else {
            monitorTimer.Enabled = true;
            MonitorButton.Text = "Stop Monitoring";
        }
    }

    private void CategoryComboBox_SelectedValueChanged(object sender, EventArgs e) {
        if (_isCategoryLoaded) {
            _selectedModel =
_modelProvider.GetModelByCategoryId(Convert.ToInt32(CategoryComboBox.SelectedValue));
            LoadModelData(_selectedModel.ModelFilePath);
        }
    }

    private void monitorTimer_Tick(object sender, EventArgs e) {
        var generatedData = _sensorDataGenerator.GenerateRandomData();

        // Display generated data in the report textbox
        ReportTextBox.Text = $"Generated Data:\r\n Start Time: {generatedData.StartTime}\r\n " +
                             $"Location: {generatedData.Location}\r\n " +
                             $"Time of Day: {generatedData.TimeOfDay}\r\n " +
                             $"Object: {generatedData.Object}\r\n " +
                             $"Posture: {generatedData.Posture}\r\n " +
                             $"Duration: {generatedData.Duration} sec.\r\n " +
                             $"Activity: {generatedData.Activity}\r\n";

        // Make predictions using the model
        var prediction = _predictionEngine.Predict(generatedData);
        bool isIntrusionDetected = prediction.IsIntrusion();

        // Display prediction results
        ReportTextBox.Text += $"Predicted Activity: {prediction.PredictedActivity}\r\n";
        ReportTextBox.Text += isIntrusionDetected ? "Potential Intrusion Detected!" : "No Intrusion
Detected." + "\r\n";

        // Scroll the report textbox to the last entry
        ReportTextBox.SelectionStart = ReportTextBox.Text.Length;
        ReportTextBox.ScrollToCaret();
```

```
        }

        private void PredictButton_Click(object sender, EventArgs e) {
            if (IsInputDataValid()) {
                HomeSensorData sensorData = new HomeSensorData {
                    StartTime = StartTimePicker.Value.ToLongTimeString(),
                    Location = LocationComboBox.Text,
                    TimeOfDay = TimeOfDayComboBox.Text,
                    Object = ObjectComboBox.Text,
                    Posture = PostureComboBox.Text,
                    Duration = Convert.ToInt32(DurationTextBox.Text),
                    Activity = ActivityComboBox.Text
                };

                // Display generated data in the report textbox
                ReportTextBox.Text = $"Generated Data:\r\n Start Time: {sensorData.StartTime}\r\n " +
                                     $"Location: {sensorData.Location}\r\n " +
                                     $"Time of Day: {sensorData.TimeOfDay}\r\n " +
                                     $"Object: {sensorData.Object}\r\n " +
                                     $"Posture: {sensorData.Posture}\r\n " +
                                     $"Duration: {sensorData.Duration} sec.\r\n " +
                                     $"Activity: {sensorData.Activity}\r\n";

                // Make predictions using the model
                var prediction = _predictionEngine.Predict(sensorData);
                bool isIntrusionDetected = prediction.IsIntrusion();

                // Display prediction results
                ReportTextBox.Text += $"Predicted Activity: {prediction.PredictedActivity}\r\n";
                ReportTextBox.Text += isIntrusionDetected ? "Potential Intrusion Detected!" : "No
Intrusion Detected." + "\r\n";
            }
        }

        private bool IsInputDataValid() {
            bool isValid = true;
            if (Convert.ToInt32(LocationComboBox.SelectedValue) > -1) {
                LocationValidationLabel.Text = ButtonLabels.ValidationPassed;
            } else {
                LocationValidationLabel.Text = ButtonLabels.ValidationError;
                isValid = false;
            }
            if (_dataValidator.CanConvertToInt(DurationTextBox.Text)) {
                DurationValidationLabel.Text = ButtonLabels.ValidationPassed;
            } else {
                DurationValidationLabel.Text = ButtonLabels.ValidationError;
                isValid = false;
            }
            return isValid;
        }
    }
}
```

```
class SensorDataGenerator {
    private static Random random = new Random();
    private static List<string> locations = new List<string> { "Living Room", "Bathroom", "Kitchen",
"Garage", "Front Door" };
    private static List<string> timesOfDay = new List<string> { "Morning", "Afternoon", "Evening",
"Night" };
    private static List<string> objects = new List<string> { "Main Door", "Bathroom Door", "Kitchen
Cabinet", "Garage Door", "Front Door" };
    private static List<string> postures = new List<string> { "Sitting", "Standing" };
    private static List<string> activities = new List<string> { "go to sleep", "use bathroom", "make
breakfast", "take a shower", "leave house", "break door", "smash window", "unauthorized access" };
    private static List<string> intrusionActivities = new List<string> { "break door", "smash window",
"unauthorized access" };
    private static List<string> intrusionTimesOfDay = new List<string> { "Night" };

    public HomeSensorData GenerateRandomData() {
        bool isIntrusion = random.NextDouble() < 0.2; // 20% chance to generate intrusion data
        string generatedTime = GenerateRandomTime(isIntrusion);
        return new HomeSensorData {
            StartTime = generatedTime,
            Location = locations[random.Next(locations.Count)],
            TimeOfDay = DetermineTimeOfDay(generatedTime),
            Object = objects[random.Next(objects.Count)],
            Posture = postures[random.Next(postures.Count)],
            Duration = random.Next(30, 600),
            Activity = isIntrusion ? intrusionActivities[random.Next(intrusionActivities.Count)] :
activities[random.Next(activities.Count)]
        };
    }

    private string GenerateRandomTime(bool isIntrusion) {
        int hour = isIntrusion ? random.Next(0, 4) : random.Next(0, 24);
        int minute = random.Next(0, 60);
        int second = random.Next(0, 60);
        return $"{hour:D2}:{minute:D2}:{second:D2}";
    }

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

        // Adjust hour range for 24-hour format
        if (hour < 0 || hour > 23) hour = 0; // default to 0 in case of invalid hour

        if (hour >= 6 && hour < 12) return "Morning";
        if (hour >= 12 && hour < 17) return "Afternoon";
        if (hour >= 17 && hour < 21) return "Evening";
        return "Night";
    }
}
```