Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр
(ступінь вищої освіти)

на тему СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНОЇ ПІДСИСТЕМИ БІОМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ ДЛЯ СИСТЕМИ РОЗУМНИЙ БУДИНОК
CREATING AN INTELLIGENT SUBSYSTEM OF BIOMETRIC IDENTIFICATION FOR THE SMART HOME SYSTEM

Виконав: студент 4 курсу, групи КНТ-212
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

БАЛАБУХА А.Р.
(ПРІЗВИЩЕ та ініціали)
Керівник ПАРХОМЕНКО А.В.
(ПРІЗВИЩЕ та ініціали)
Рецензент ПОЛЯКОВ М.О.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
________________Сергій СУББОТІН
"____" ________________2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БАЛАБУХИ Артема Руслановича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Створення інтелектуальної підсистеми біометричної ідентифікації для системи Розумний будинок. Creating an Intelligent Subsystem of Biometric Identification for the Smart Home System

керівник проєкту (роботи) к.т.н., доцент, ПАРХОМЕНКО Анжеліка Володимирівна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від " 07 " квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 01 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Програмна реалізація інтелектуальної підсистеми біометричної ідентифікації. 4. Експлуатація, тестування та експериментальне дослідження інтелектуальної підсистеми біометричної ідентифікації.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ПАРХОМЕНКО А.В., к.т.н, доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання « 20 » квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	ПЗ, Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент __________ Артем БАЛАБУХА
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) __________ Анжеліка ПАРХОМЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 90 с., 7 табл., 14 рис., 2 дод., 40 джерел.

БІОМЕТРИЧНА ІДЕНТИФІКАЦІЯ, РОЗПІЗНАВАННЯ ОБЛИЧ, КОМП'ЮТЕРНИЙ ЗІР, КАСКАДИ ХААРА, ЗАХИСТ ВІД СПУФІНГУ, ПЕРИФЕРІЙНІ ОБЧИСЛЕННЯ.

Об'єкт дослідження – алгоритми та процеси обчислень та їхнє практичне застосування для створення засобів опрацювання даних для біометричної ідентифікації.

Предмет дослідження – методи та алгоритми комп'ютерного зору для біометричної ідентифікації особи.

Мета роботи – підвищення точності біометричної ідентифікації особи у системі «Розумний будинок» на основі алгоритмів та програмного забезпечення комп'ютерного зору.

Матеріали, методи та технічні засоби: мова програмування C#, методи комп'ютерного зору, бібліотеки OpenCV, FaceRecognitionDotNet, протокол MQTT, формат JSON, платформа домашньої автоматизації OpenHAB.

Результати. Розроблено інтелектуальну підсистему, яка в реальному часі ідентифікує особу розпізнаванням обличчя у відеопотоці з перевіркою автентичності та формує JSON-пакет з ідентифікатором та рівнем збігу.

Висновки. Виконано програмну реалізацію інтелектуальної підсистеми біометричної ідентифікації особи та її інтеграцію у систему «Розумний будинок», що забезпечує автоматичне керування доступом і персоналізованими сценаріями без застосування фізичних ідентифікаторів.

Галузь використання – системи «Розумний будинок», охоронні системи житлових і промислових приміщень, автоматизація сценаріїв, інтеграція з мультимедійними комплексами.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 90 pages, 7 tables, 14 figures, 2 appendixes, 40 sources.

BIOMETRIC IDENTIFICATION, FACIAL RECOGNITION, COMPUTER VISION, HAAR CASCADES, ANTI-SPOOFING, EDGE COMPUTING.

Object of research – algorithms and computing processes and their practical application for creating data processing tools for biometric identification.

Subject of research – computer vision methods and algorithms for biometric person identification.

Purpose of the work – increasing the accuracy of biometric person identification in the "Smart Home" system based on computer vision algorithms and software.

Materials, methods, and technical means: C# programming language, computer vision methods, OpenCV, FaceRecognitionDotNet libraries, MQTT protocol, JSON format, OpenHAB home automation platform.

Results. An intelligent subsystem has been developed that identifies a person in real time by facial recognition in a video stream with authenticity verification and forms a JSON packet with an identifier and match level.

Conclusions. A software implementation of the intelligent subsystem for biometric person identification and its integration into the "Smart Home" system was performed, which provides automatic access control and personalized scenarios without the use of physical identifiers.

Field of use – "Smart Home" systems, security systems for residential and industrial premises, scenario automation, integration with multimedia complexes.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AI	– Artificial Intelligence;
CNN	– Convolutional Neural Network;
DDoS	– Distributed Denial of Service;
DNN	– Deep Neural Network;
FAR	– False Acceptance Rate;
FRR	– False Rejection Rate;
FSM	– Finite State Machine;
GUI	– Graphical User Interface;
ID	– Identifier;
IoT	– Internet of Things;
JSON	– JavaScript Object Notation;
LBPH	– Local Binary Patterns Histograms;
MQTT	– Message Queuing Telemetry Transport;
PCA	– Principal Component Analysis;
QoS	– Quality of Service;
RGB	– Red, Green, Blue;
ROI	– Region of Interest;
ToF	– Time of Flight;
XML	– eXtensible Markup Language;
ПЗ	– програмне забезпечення.

ВСТУП

Сучасний етап розвитку технологій характеризується стрімким зростанням автоматизації й інтеграції інтелектуальних помічників у повсякденне життя. Розвиток штучного інтелекту та нейромереж дав поштовх до створення інструментів, покликаних полегшити побут [1].

Дедалі більше людей встановлюють системи «Розумний будинок», перекладаючи частину рутини на автоматизовані сценарії.

Системи «Розумний будинок» керують приладами та виконують послідовні сценарії, але досі не вміють розпізнавати, хто саме стоїть перед камерою. Сценарій однаково спрацьовує для власника і для гостя, бо система реагує на рух, а не на особу. Відкрити двері може будь-хто з ключем чи RFID-карткою [2]. При цьому камера в більшості таких систем не використовується для ідентифікації, хоча технічно здатна розв’язати цю проблему без додаткового обладнання [3].

Впровадження підсистеми біометричної ідентифікації дозволяє виключити фізичні носії доступу та знизити потребу в голосовій взаємодії, оскільки система автоматично визначатиме особу та її рівень доступу [2]-[4]. Однак цей метод має практичні проблеми. Розпізнавання обличчя в реальних умовах стикається з двома ключовими перешкодами.

По-перше, освітлення постійно змінюється: денне світло з вікна, штучне освітлення різної температури, тіні в кутах кімнат [5].

По-друге, систему можна обманути фотографією – зловмисник показує камері знімок власника, отримуючи несанкціонований доступ [6]. Обидві проблеми вирішуються програмно – без заміни обладнання.

Метою роботи є підвищення точності біометричної ідентифікації особи у системі «Розумний будинок» на основі алгоритмів та програмного забезпечення комп'ютерного зору.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Стрімкий розвиток концепції «Інтернету речей» (IoT) призвів до появи складних систем автоматизації житлового простору, відомих як «Розумний будинок».

У даному розділі проводиться аналіз поточного стану технологій «Розумного будинку», розглядаються переваги та недоліки існуючих методів біометричної ідентифікації.

Особлива увага приділяється вивченню вразливостей IoT-інфраструктури, етичним аспектам використання комп’ютерного зору в приватному просторі та чинникам навколишнього середовища, що перешкоджають стабільній роботі систем розпізнавання облич.

Результатом аналізу є формування чіткої постановки завдань дипломного проєкту

1.1 Огляд архітектури сучасних систем класу «Розумний будинок»

Розвиток концепції «Розумний дім» призвів до формування складної багаторівневої архітектури, де кожний елемент виконує чітко визначену роль у забезпеченні життєдіяльності об'єкта. За своєю суттю, це система об'єднаних пристроїв, що здатні виконувати дії і вирішувати певні повсякденні завдання без безпосередньої участі людини [7].

Зазвичай архітектура такої системи будується за ієрархічним принципом і складається з трьох основних рівнів, що забезпечують циклічність процесів сприйняття, керування та виконання як це продемонстровано на рисунку 1.1 [8]-[9].

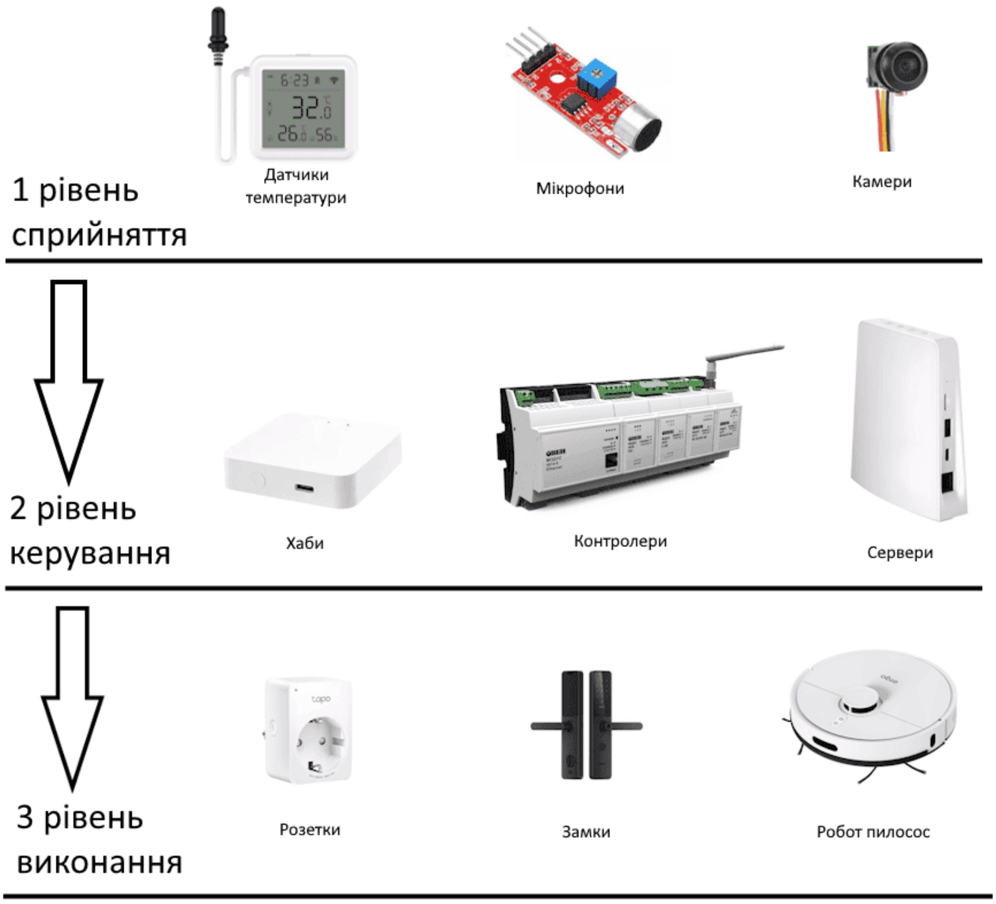

Рисунок 1.1 – Рівні архітектури системи «Розумний будинок» [8]-[9]

Архітектура подібних систем складається зазвичай з наступних рівнів:

– рівень сприйняття: датчики руху, температури, вологості, інтенсивності сонячного світла, камери, мікрофони – збирає дані всередині та зовні приміщення;

– рівень керування: хаби, контролери – центральний вузол приймання, аналізу та перевірки даних, побудови сценарію і надсилання команд виконавчим приладам;

– рівень виконання: реле, розумні розетки, роботи-пилососи, електронні замки, мультимедійні пристрої – виконання надісланих команд.

Важливою характеристикою сучасної архітектури є її здатність до інтеграції. Завдяки розумному дому всі системи в будівлі можуть працювати разом, виконуючи завдання відповідно до заданих сценаріїв або змін у навколишньому середовищі [7].

Як зазначено в [9], централізація управління дозволяє не лише автоматизувати побут, а й створити цілісну систему моніторингу, яка здатна запобігати критичним ситуаціям ще до їх виникнення

Отже, огляд архітектури демонструє, що ключовим елементом системи є саме центральний вузол. Проте для підвищення рівня безпеки та якості персоналізації сучасні системи потребують впровадження методів ідентифікації, які були б невід'ємною частиною цієї архітектури, що й обумовлює необхідність аналізу біометричних технологій.

1.2 Проблематика безпеки та персоналізації в системах домашньої автоматизації

Впровадження біометричної ідентифікації в домашню автоматизацію породжує комплекс проблем, що лежать на перетині технологічних обмежень та соціально-етичних викликів. Проблематика безпеки охоплює як захист самого пристрою від зламу, так і забезпечення цілісності біометричних шаблонів, тоді як персоналізація вимагає розробки інтелектуальних алгоритмів, здатних коректно інтерпретувати контекст перебування особи в будинку [10].

1.2.1 Вразливості IoT-інфраструктури та кіберзагрози

Системи «Розумного будинку» є привабливою мішенню для кіберзлочинців через їхню високу взаємопов'язаність та часто недостатній рівень захисту програмного забезпечення (ПЗ).

Згідно з дослідженнями, близько 70% найбільш поширених IoT-пристроїв мають критичні вразливості, серед яких відсутність шифрування даних при передачі, слабкі механізми автентифікації та наявність недокументованих функцій.

Проблема загострюється гетерогенністю компонентів: пристрої різних виробників використовують власні закриті протоколи, що ускладнює створення єдиного захищеного периметра [10].

Особливе занепокоєння викликають атаки типу «відмова у обслуговуванні» (DDoS), де скомпрометовані розумні пристрої об'єднуються в ботнети, а також несанкціоноване спостереження [11].

Відомі випадки, коли через вразливості в хмарних сервісах або вбудованому ПЗ хакери отримували доступ до відеокамер і термостатів, порушуючи приватність мешканців без їх відома [11].

В контексті біометрії будь-який злам системи керування означає потенційний доступ до цифрових зліпків обличчя або відбитків пальців, які, на відміну від паролів, неможливо змінити у разі компрометації.

1.2.2 Ризики біометричної ідентифікації та спуфінг-атаки

Біометрична ідентифікація, попри свою зручність, не є абсолютно надійним методом захисту. Головною загрозою для систем розпізнавання облич є атаки підміни, або спуфінг. Зловмисник може спробувати обійти систему, представивши сенсору штучний біометричний зразок легітимного користувача [12].

Класифікацію основних видів спуфінг-атак за рівнями складності та методами їх нейтралізації наведено у таблиці1.1 [12]-[14].

Для нейтралізації цих загроз необхідно впроваджувати алгоритми перевірки «живості». Активні методи вимагають від користувача виконання випадкових дій (кліпання, поворот голови, вимова фрази), що створює інтерактивний бар'єр для статичних підробок. Пасивні методи аналізують специфічні ознаки реальної шкіри, такі як пори, вологість або підшкірний кровообіг, що доступно лише при використанні високоякісної оптики або інфрачервоних сенсорів [13].

Таблиця 1.1 – Характеристика методів спуфінгу та способів протидії [12]-[14]

Тип атаки (Спуфінг)	Механізм реалізації	Рівень складності	Методи протидії
Фото-атака	Пред'явлення надрукованого фото або зображення на екрані смартфона	Низький	Аналіз текстури, перевірка мікрорухів
Відео-атака	Відтворення відеозапису особи на цифровому носії перед камерою	Середній	Детекція відблисків екрана, аналіз моїрових візерунків
3D-маска	Використання об'ємної моделі обличчя, виготовленої з латексу чи пластику	Високий	Стереоскопічний зір, ІЧ-підсвічування, аналіз глибини
Глибокі фейки	Генерація реалістичного відео за допомогою AI для підміни потоку	Дуже високий	Частотно-часовий аналіз, нейромережева детекція артефактів

1.2.3 Виклики персоналізації та конфлікти сценаріїв

Метою персоналізації є створення середовища, що адаптується під звички конкретної людини. Це включає автоматичне налаштування температури, освітлення, вибір музичного супроводу та навіть конфігурацію кухонних приладів. Проте реалізація таких сценаріїв стикається з проблемою багатокористувацького конфлікту. У ситуації, коли в одній зоні перебувають кілька осіб з діаметрально протилежними вподобаннями (наприклад, один мешканець віддає перевагу холодному світлу, а інший – теплому), система повинна мати логіку вирішення суперечностей [15].

Існують наступні стратегії вирішення конфліктів персоналізації:

– пріоритезація: надання переваги налаштуванням власника або особи з вищим рангом у сімейній ієрархії;

– середньозважене значення: встановлення параметрів, що є компромісними для всіх присутніх;

– останній ідентифікований: система реагує на потреби того, хто увійшов у приміщення останнім;

– профілі активності: активація сценарію на основі спільної діяльності [15].

Додатковою проблемою є хибнопозитивні спрацювання (False Acceptance), коли система помилково ідентифікує сторонню особу як мешканця, надаючи їй доступ до приватних зон або персональних даних [15]. Також актуальною є проблема «відмови у реєстрації», спричинена змінами зовнішності (травми, старіння, макіяж) або поганими умовами освітлення, що призводить до блокування легітимного користувача [16].

1.2.4 Проблеми приватності та етичні аспекти

Впровадження біометричних систем у простір «Розумного будинку» створює низку критичних викликів, які виходять за межі суто технічних вразливостей і стосуються фундаментальних прав людини та етичних норм використання штучного інтелекту:

– алгоритмічна упередженість та точність. Однією з головних етичних проблем є нерівномірна точність розпізнавання для різних демографічних груп. Алгоритми, навчені на нерепрезентативних вибірках, часто демонструють вищий рівень помилок при ідентифікації осіб певного віку, статі або етнічної приналежності [17]. Це створює ризик алгоритмічної дискримінації та помилкової відмови в доступі;

– інформована згода та «право на забуття». Використання біометрії вимагає чіткого механізму отримання згоди. Проблема посилюється у випадку гостей або випадкових відвідувачів, чиї дані можуть бути зафіксовані системою без їхнього відома. Крім того, виникає питання видалення даних: як гарантувати повне знищення біометричних шаблонів після того, як користувач перестав користуватися системою [18];

– компрометація незмінних даних. На відміну від традиційних паролів, біометричні дані є невід'ємною частиною особистості. У разі витоку з хмарного сховища або зламу локального сервера, користувач не може змінити свої біометричні дані, що робить такий витік критичним і незворотним [17];

– тіньовий збір даних. Існує ризик використання зібраних профілів для цілей, що виходять за межі безпеки дому [18].

Для вирішення цих проблем має передбачатися дотримання наступних принципів:

– локалізація даних: мінімізація передачі біометричної інформації на зовнішні сервери [18];

– анонімізація: зберігання лише математичних хеш-кодів (векторів ознак) замість реальних зображень, що унеможливлює відновлення з бази даних [18];

– забезпечення прозорості: створення інтерфейсу, який дозволяє користувачу бачити, які саме дані зберігаються, та надавати можливість їх миттєвого видалення [18].

Підсумовуючи аналіз проблем приватності та етичних аспектів, слід зазначити, що впровадження біометрії в екосистему «Розумного будинку» потребує компромісу між технологічною функціональністю та правовою захищеністю мешканців. Використання принципів Privacy by Design, локалізація обробки даних та впровадження прозорих алгоритмів згоди є необхідними умовами для створення безпечного цифрового середовища.

Виявлені ризики – від технічних вразливостей IoT-інфраструктури до етичних проблем алгоритмічної упередженості – формують комплекс вимог до сучасних систем автоматизації. Саме тому для розробки власної підсистеми ідентифікації необхідно детально вивчити існуючі на ринку рішення та платформи, щоб обрати ту архітектуру, яка мінімізує вказані загрози.

1.3 Порівняльний аналіз методів біометричної ідентифікації

Для побудови ефективної системи «Розумного будинку» необхідно обрати метод ідентифікації, який забезпечує баланс між безпекою, вартістю та користувацьким досвідом.

У таблиці 1.2 наведено порівняння основних біометричних модальностей на основі «семи стовпів біометрії»: універсальності, унікальності, постійності, вимірюваності, продуктивності, прийнятності та стійкості до обходу [19]-[20].

Таблиця 1.2 – Порівняльна характеристика біометричних методів ідентифікації [19]-[20]

Характеристика	Відбиток пальця	Геометрія обличчя	Райдужна оболонка	Голос
Унікальність	Висока	Середня	Дуже висока	Низька
Постійність	Висока	Середня (старіння)	Висока	Низька (хвороби)
Зручність	Середня (контакт)	Висока (дистанційно)	Низька (позиціювання)	Висока
Стійкість до обману	Висока	Середня	Дуже висока	Низька
Вартість обладнання	Низька	Середня	Висока	Дуже низька
Прийнятність	Висока	Дуже висока	Середня	Висока
Швидкість розпізнавання	< 1 сек	0.5–2 сек	1–3 сек	2–5 сек

Аналіз показує, що розпізнавання облич є найбільш перспективним методом для домашньої автоматизації завдяки поєднанню високої зручності (дистанційна ідентифікація без необхідності контакту з сенсором) та найвищого рівня прийнятності серед користувачів.

Хоча стійкість до обману є середньою, цей показник значно покращується за допомогою ПЗ захисту від спуфінгу та перевірки на живість.

1.4 Огляд популярних сучасних систем класу «Розумний будинок»

Сучасний ринок систем «Розумний будинок» характеризується стрімкою диференціацією технологічних підходів.

Залежно від моделі розгортання, архітектури зберігання даних та способу взаємодії з користувачем, усі наявні рішення можна розділити на дві великі групи: комерційні екосистеми та відкриті локальні платформи [21].

1.4.1 Комерційні системи

До цієї категорії належать рішення від великих технологічних компаній: Apple HomeKit, Google Home, Amazon Alexa та Samsung SmartThings [21].

Переваги комерційних екосистем [21]:

– орієнтація на масового споживача: пристрої готові до використання одразу після підключення без потреби в детальному налаштуванні;

– вбудована інтеграція з голосовими асистентами: Siri, Google Assistant, Alexa.

Недоліки комерційних екосистем [21]:

– закритість: жорсткі обмеження на інтеграцію зі сторонніми продуктами;

– залежність від хмари: більшість логіки обробляється на серверах компанії, що унеможливлює автономну роботу без інтернету;

– конфіденційність: відеопотік і біометричні дані передаються на зовнішні сервери, а хмарна обробка вносить мережеві затримки, критичні для систем із взаємодією в реальному часі [22].

1.4.2 Відкриті локальні платформи

Системи відкритих локальних платформ розробляються спільнотами і призначені для розгортання на локальних серверах користувачів. Найпопулярнішими представниками є Home Assistant і Jeedom[21].

Переваги відкритих платформ [21]:

– широка сумісність: підключення до пристроїв без власного застосунку через сторонні інтеграції;

– локальна обробка: всі обчислення виконуються в локальній мережі без мережевих затримок хмарних сервісів;

– розширюваність: користувачі можуть писати власні скрипти та модулі інтеграції.

Недоліки відкритих платформ [21]-[23]:

– обмежена функціональність сторонніх інтеграцій: розробники платформи не мають доступу до внутрішньої документації виробників пристроїв, тому частина функцій може бути недоступною або працювати повільніше, ніж із рідним застосунком;

– безпека локальної мережі: якщо зловмисник отримає доступ до мережі, він отримає і доступ до системи керування пристроями [23];

– відсутність відповідальності за якість коду: ліцензійні угоди відкритих платформ, як правило, повністю знімають з розробників відповідальність за помилки і вразливості [23].

Проведений аналіз сучасного ринку систем класу «Розумний будинок» дозволяє зробити наступні висновки:

– комерційні екосистеми (Apple HomeKit, Google Home тощо) забезпечують високий рівень зручності та готовності до роботи, проте їхня закритість та орієнтація на хмарні обчислення створюють суттєві ризики для конфіденційності біометричних даних та обмежують автономність системи при відсутності інтернет-з’єднання;

– відкриті локальні платформи пропонують максимальну гнучкість та дозволяють реалізувати концепцію локальної обробки даних. Це є критично важливим для інтелектуальних підсистем ідентифікації, оскільки мінімізує мережеві затримки та гарантує, що персональні дані мешканців не залишають периметр приватної мережі;

– основним викликом для відкритих платформ залишається складність налаштування та необхідність забезпечення безпеки самої локальної мережі, що вимагає розробки додаткових механізмів захисту та ретельного вибору програмних компонентів.

Отже, для реалізації інтелектуальної підсистеми біометричної ідентифікації найбільш перспективним є використання відкритих локальних платформ. Це дозволяє інтегрувати власні моделі машинного зору, забезпечуючи при цьому баланс між функціональністю, швидкістю реакції та етичними вимогами щодо приватності користувача.

1.5 Аналіз факторів, що впливають на точність розпізнавання

Ефективність біометричної ідентифікації в умовах реального будинку суттєво залежить від зовнішніх та внутрішніх чинників, які можуть призводити до помилок першого або другого роду [5]:

– освітлення: зміна кута падіння світла або низька інтенсивність освітлення можуть спотворювати тіні на обличчі, що критично впливає на стабільність дескрипторів Приклади впливу різних умов освітленості на чіткість рис обличчя наведено на рисунку 1.2 [5];

– положення голови: відхилення від фронтального положення (yaw, pitch, roll) зменшує кількість видимих ключових ознак. Наприклад, поворот голови понад 30 градусів значно ускладнює детекцію для класичних методів [5];

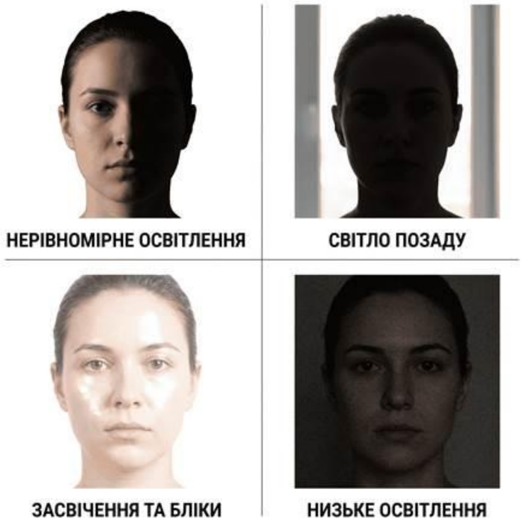

Рисунок 1.2 – Приклад впливу умов освітлення на зображення обличчя [5]

– оклюзії: наявність масок, окулярів, головних уборів або волосся, що закриває частину обличчя, перешкоджає повноцінному витягуванню ознак [24];

– якість зображення: роздільна здатність камери, цифрові шуми та артефакти стиснення відеопотоку впливають на чіткість дрібних деталей обличчя [5];

– індивідуальні зміни: макіяж, вираз обличчя (емоції), вікові зміни або травми можуть призводити до невідповідності поточного зображення збереженому шаблону [24].

Отже, ефективність функціонування інтелектуальної підсистеми ідентифікації в умовах житлового приміщення визначається здатністю алгоритмів нівелювати вплив динамічних чинників середовища.

Проведений аналіз дозволяє виділити наступні ключові аспекти:

– освітлення та ракурс є найбільш критичними технічними бар'єрами. Навіть досконалі моделі ідентифікації можуть втрачати точність за умови різких світлових градієнтів або значних відхилень голови від фронтальної осі;

– оклюзії та індивідуальні зміни (макіяж, емоції) вносять додаткову невизначеність, що вимагає використання стійких методів витягування ознак, здатних розпізнавати особу за частковими даними;

– якість апаратної частини (камери та каналів зв'язку) закладає фундамент для подальшої обробки; артефакти стиснення або низька роздільна здатність можуть зробити ідентифікацію неможливою незалежно від обраного алгоритму.

Отже, розробка надійної системи потребує впровадження методів попередньої обробки зображень для нормалізації освітлення та використання алгоритмів, стійких до просторових викривлень.

1.6 Постановка завдань дипломної роботи

На основі проведеного аналізу предметної області, виявлених технічних проблем (змінне освітлення, ракурси) та етичних викликів, сформовано мету та перелік завдань дослідження.

Метою роботи є підвищення точності біометричної ідентифікації особи у системі «Розумний будинок» на основі алгоритмів та програмного забезпечення комп'ютерного зору.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– виконати аналіз вимог до програмного та алгоритмічного забезпечення інтелектуальної підсистеми біометричної ідентифікації;

– розробити модель та схему опрацювання даних для біометричної ідентифікації особи в режимі реального часу;

– провести порівняльний аналіз існуючих алгоритмів попередньої обробки зображень, детектування та розпізнавання облич;

– дослідити методи програмного захисту від спуфінг-атак та оцінити їх ефективність в умовах мінливої або низької освітленості;

– розробити та реалізувати алгоритм перевірки на «живість» об'єкта для нівелювання спроб несанкціонованого доступу за допомогою статичних фотографій або відеозаписів;

– виконати програмну реалізацію інтелектуальної підсистеми ідентифікації та її інтеграцію з платформою домашньої автоматизації;

– провести серію експериментальних випробувань розробленої інтелектуальної підсистеми для оцінки її точності та швидкодії в різних сценаріях.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Системний аналіз предметної області та існуючих технологічних рішень, проведений у першому розділі, дозволяє перейти від теоретичного вивчення до етапу обґрунтованого проєктування підсистеми.

У межах другого розділу здійснюється формалізація технічних вимог до програмного забезпечення, а також вибір оптимального математичного апарату та інструментарію розробки.

Розроблені архітектурні рішення та схеми мережевої взаємодії сформують базис для подальшої програмної реалізації та експериментального дослідження системи в реальних умовах експлуатації.

2.1 Аналіз та формалізація вимог до програмного забезпечення

Процес проєктування підсистеми біометричної ідентифікації для екосистеми «Розумний будинок» базується на комплексному аналізі та детальній формалізації вимог до ПЗ.

З огляду на специфіку експлуатації системи в умовах житлового приміщення, що характеризується динамічною зміною освітлення та різними ракурсами детекції обличчя, вимоги до системи поділяються на три основні групи: функціональні, нефункціональні та специфічні вимоги до алгоритмічного апарату опрацювання візуальних даних.

Такий підхід дозволяє забезпечити не лише технічну ефективність розпізнавання, а й дотримання етичних норм приватності мешканців.

2.1.1 Функціональні вимоги

Функціональні вимоги визначають перелік конкретних завдань та операцій, які повинна виконувати інтелектуальна підсистема для досягнення поставленої мети.

ПЗ має забезпечувати реалізацію наступних функцій:

– захоплення та попередня обробка відеопотоку: забезпечення стабільного отримання кадрів у реальному часі з джерела (IP-камера або локальний сенсор);

– інтелектуальна детекція обличчя: автоматичне виявлення обличчя особи в кадрі з використанням алгоритмів комп'ютерного зору та визначення області інтересу (ROI);

– нормалізація зображення: програмна корекція яскравості, контрастності та вирівнювання положення обличчя для нівелювання впливу змінного освітлення приміщення;

– біометрична ідентифікація: витягування векторів ознак та їх порівняння з локальною базою даних для встановлення особистості мешканця;

– захист від спуфінг-атак: програмна перевірка автентичності біометричного зразка для відсікання спроб несанкціонованого доступу за допомогою фотографій або відеозаписів;

– інтеграція з платформою домашньої автоматизації: формування та відправка результатів ідентифікації (ID особи та рівень впевненості);

– керування базою даних користувачів: можливість додавання нових профілів мешканців, видалення існуючих та оновлення біометричних шаблонів для підвищення точності розпізнавання;

– логування та сповіщення: ведення цифрового журналу подій, що фіксує час успішної ідентифікації, спроби спуфінгу та статус передачі даних у систему керування.

2.1.2 Нефункціональні вимоги

Дані вимоги визначають якісні характеристики системи та обмеження, в яких вона повинна функціонувати для забезпечення надійності та безпеки в межах житлового простору:

– продуктивність та латентність: загальний час від моменту детекції обличчя в кадрі до формування сигналу на виконання сценарію (наприклад, відкриття дверей) не повинен перевищувати 1.2 - 1.5 секунди. Це критично для забезпечення природної взаємодії користувача з системою без відчутних затримок;

– надійність та відмовостійкість: система має стабільно функціонувати в режимі 24/7. У разі втрати зв'язку з сервісами або центральним хабом, локальний вузол обробки повинен продовжувати ідентифікацію та керування критичними сценаріями доступу в автономному режимі;

– стійкість до візуальних завад та змін середовища: алгоритми мають забезпечувати коректну ідентифікацію при зміні рівня освітленості (від 10 до 1000 люкс), наявності часткових оклюзій таких як окуляри, зміна зачіски та при відхиленні голови до 30 градусів;

– безпека та приватність даних: забезпечення конфіденційності біометричних шаблонів мешканців через використання принципів Privacy by Design. Система повинна зберігати не вихідні зображення облич, а їх математичні хеш-коди, використовуючи шифрування каналів передачі даних;

– масштабованість: архітектура ПЗ повинна дозволяти підключення додаткових камер, наприклад, для різних кімнат або входів та інтеграцію нових типів пристроїв без необхідності повної перебудови ядра системи.

2.1.3 Специфічні вимоги до алгоритмічного апарату опрацювання візуальних даних

Окремим класом вимог є вимоги до математичного та алгоритмічного апарату, що зумовлені специфікою біометричної ідентифікації в житловому середовищі, а саме:

– адаптивність нормалізації зображення: алгоритми попередньої обробки мають автоматично корегувати рівень експозиції та баланс білого для стабілізації роботи за умов мінливого освітлення (природне світло з вікон, штучне освітлення різної температури або повна темрява);

– динамічне керування порогом впевненості: можливість програмного налаштування рівня подібності векторів ознак для балансування між ймовірністю помилкового доступу та помилкової відмови залежно від рівня безпеки конкретного сценарію.

2.1.4 Розробка діаграми прецедентів

Для уточнення взаємодії між користувачами та ПЗ, а також деталізації меж системи, було розроблено діаграму прецедентів яка зображена на рисунку 2.1.

Ця діаграма дозволяє формалізувати основні сценарії використання інтелектуальної підсистеми біометричної ідентифікації та визначити ролі ключових акторів.

У цій моделі виділено основні сценарії взаємодії та ролі акторів. Опис взаємодії:

– IP-камера забезпечує безперервну передачу відеопотоку високої роздільної здатності для подальшого аналізу;

– мешканець ініціює прецедент «Біометричний доступ та персоналізація» через появу в зоні детекції камери;

– підсистема виконує ключові етапи обробки: детекцію обличчя, перевірку антиспуфінгу , витягування ознак та верифікацію даних за базою;

– система «Розумний будинок» виступає виконавчою системою, що отримує статус ідентифікації та активує відповідні персоналізовані сценарії (відкриття замка, налаштування клімату, освітлення тощо);

– адміністратор здійснює налаштування бази користувачів («білого списку»), керування правами доступу та моніторинг журналу подій.

Така структура дозволяє ізолювати логіку біометричної ідентифікації від апаратного забезпечення, що спрощує модернізацію окремих модулів системи та забезпечує локальну обробку конфіденційних даних.

Рисунок 2.1 – Діаграма прецедентів підсистеми біометричної ідентифікації

2.2 Розробка моделі біометричної ідентифікації та архітектури інтелектуальної підсистеми

Для забезпечення стабільної роботи інтелектуальної підсистеми в умовах реального часу та мінімізації логічних помилок, процес ідентифікації особи формалізується за допомогою математичного апарату скінченних автоматів. Скінченний автомат (FSM) дозволяє представити життєвий цикл біометричної ідентифікації як послідовність дискретних станів, перехід між якими відбувається за чітко визначеними подіями та представлено на рисунку 2.2 [25].

Використання такої моделі в межах проєктування інтелектуальної підсистеми для біометричної ідентифікації гарантує детермінованість підсистеми: кожна операція – від виявлення руху до запуску персоналізованого сценарію – виконується лише після успішного завершення попереднього етапу [25].

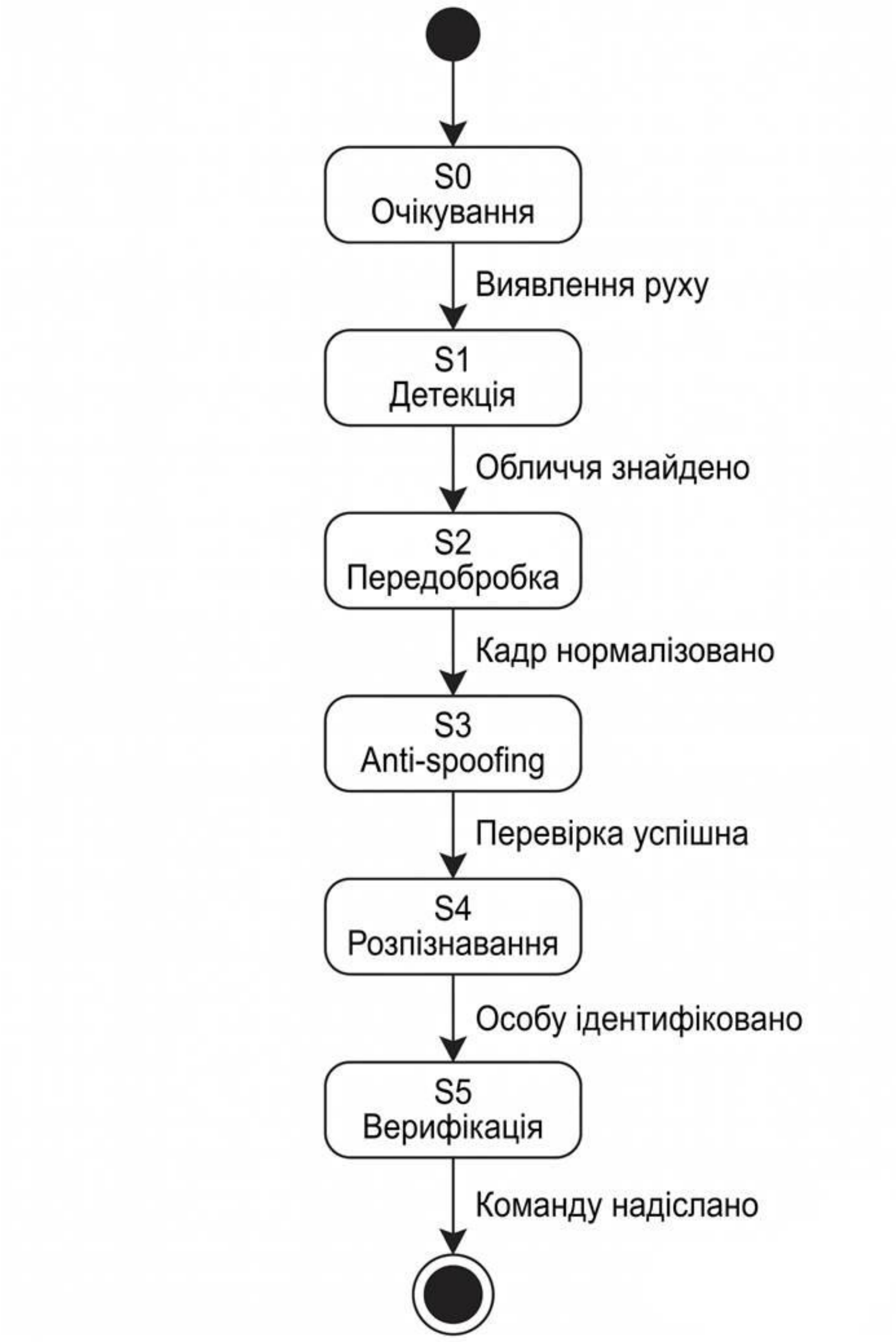

Рисунок 2.2 – Модель скінченного автомата процесу біометричної ідентифікації [25]

Логічну структуру процесу ідентифікації представлено через наступні стани:

– стан S_0 (Очікування): початковий стан, у якому система проводить фоновий аналіз відеопотоку на наявність руху. Це дозволяє економити обчислювальні ресурси, активуючи основні алгоритми лише за потреби;

– стан S_1 (Детекція): перехід у цей стан відбувається при виявленні динамічних змін у кадрі. Проводиться локалізація обличчя та виділення ROI;

– стан S_2 (Нормалізація): етап підготовки зображення, де застосовуються алгоритми корекції освітлення та вирівнювання ракурсу, що є критичним для стабільного розпізнавання в домашніх умовах;

– стан S_3 (Антиспуфінг): специфічний етап перевірки автентичності біометричного зразка. Система аналізує вхідні дані на ознаки спуфінг-атак;

– стан S_4 (Розпізнавання): виконання нейромережевих алгоритмів для витягування векторів ознак та їх порівняння з локальною базою даних мешканців;

– стан S_5 (Прийняття рішення): фінальний етап, на якому формується пакет даних із результатом ідентифікації та передається через протокол до платформи для виконання сценарію.

Для реалізації вимог щодо масштабованості та технічної адаптивності обрано модульний підхід до розробки структури ПЗ.

Ця концепція передбачає розбиття комплексу на самостійні блоки, що дозволяє досягти оптимальних показників функціональної цілісності компонентів при збереженні їхньої максимальної незалежності один від одного.

Це гарантує високу відмовостійкість системи та полегшує її подальшу модернізацію наприклад, заміну моделі розпізнавання обличчя на більш сучасну без зміни загальної логіки роботи системи.

Детальну схему взаємодії основних модулів системи наведено на рисунку 2.3.

Рисунок 2.3 – Схема модульної архітектури інтелектуальної підсистеми

Структура схеми поділяється на наступні функціональні рівні:

– рівень захоплення даних: забезпечує абстракцію від апаратного забезпечення та відповідає за отримання сирого відеопотоку з камери;

– рівень підготовки та фільтрації: містить модулі для покращення якості зображення, усунення шумів та нормалізації візуальних даних, що необхідно для стабільної роботи алгоритмів;

– рівень біометричного аналізу: є центральним вузлом системи, де реалізовано алгоритми виявлення обличчя, перевірки автентичності (захисту від підробки) та безпосередньої ідентифікації особи;

– рівень системної інтеграції: відповідає за формування результатів аналізу у структурований формат та їх передачу в зовнішню платформу домашньої автоматизації для виконання сценаріїв.

2.3 Вибір та обґрунтування технологій та інструментарію розробки

Практична реалізація ієрархічної архітектури системи біометричної ідентифікації потребує обґрунтованого вибору програмно-технічного стеку.

Цей етап є критично важливим, оскільки обрані технології мають забезпечувати оптимальний баланс між обчислювальною потужністю, швидкістю розробки та стійкістю до динамічних чинників середовища, таких як змінне освітлення та спроби спуфінг-атак.

Ефективність функціонування підсистеми в межах екосистеми «Розумний будинок» безпосередньо залежить від того, наскільки інструментарій розробки відповідає раніше сформульованим функціональним та нефункціональним вимогам.

Зокрема, вибір інструментів диктується наступними ключовими факторами:

– забезпечення реального часу: інструменти повинні мінімізувати затримки на кожному етапі опрацювання даних – від захоплення кадру до прийняття рішення;

– локалізація обчислень: програмний стек має бути оптимізованим для роботи на локальних пристроях без обов'язкового використання хмарних ресурсів, що гарантує приватність мешканців;

– інтеграційна здатність: обрані протоколи та формати даних мають забезпечувати безперешкодну взаємодію між модулем ідентифікації та існуючими платформами домашньої автоматизації;

– масштабованість та підтримка: використання технологій із відкритою документацією та широкою підтримкою спільноти дозволяє реалізувати модульний принцип побудови системи та полегшує її подальше вдосконалення.

У даному підрозділі буде проведено комплексний порівняльний аналіз існуючих технологічних рішень – від вибору мов програмування до специфічних бібліотек комп'ютерного зору – з метою формування найбільш раціонального технологічного базису для реалізації інтелектуальної підсистеми.

2.3.1 Обґрунтування вибору мови програмування

Визначення базової мови програмування для реалізації інтелектуального модуля біометричної ідентифікації є критичним етапом, оскільки вона має забезпечити ефективне керування системними ресурсами при високому рівні безпеки виконання коду.

Оскільки обробка відеопотоку в реальному часі вимагає значних обчислювальних витрат, обраний технологічний стек повинен підтримувати паралельні обчислення та мати низьку латентність при взаємодії з апаратним забезпеченням.

Під час проєктування було проведено порівняльний аналіз трьох найбільш затребуваних технологічних стеків у сфері комп'ютерного зору та систем автоматизації: Python, C++ та C#.

Оцінка проводилася за критеріями, що є вирішальними для стабільного функціонування підсистеми в екосистемі «Розумний будинок», а саме: обчислювальна потужність, безпека роботи з пам'яттю та підтримка багатопотокової обробки даних і продемонстровано у таблиці 2.1 [26-30].

Проведений аналіз демонструє, що використання мови C# є найбільш доцільним для даного проєкту. Вона забезпечує високу продуктивність, порівнянну з C++, але при цьому значно спрощує розробку завдяки механізмам безпечного керування пам'яттю. На відміну від Python, C# дозволяє ефективно розподіляти навантаження між ядрами процесора при аналізі відеопотоку, що усуває затримки при ідентифікації кількох осіб або виконанні складних сценаріїв захисту від спуфінгу [30]. Це робить платформу .NET надійним фундаментом для інтелектуальної підсистеми «Розумного будинку».

Таблиця 2.1 – Порівняльна характеристика мов програмування для біометричної ідентифікації [27-30]

Параметри порівняння	Python	C++	C#
Ефективність виконання коду	Помірна (інтерпретована природа мови)	Екстремальна (низькорівнева компіляція)	Оптимальна (завдяки JIT-оптимізації)
Гарантії безпеки пам'яті	Високі (кероване середовище)	Низькі (висока ймовірність помилок сегментації)	Високі (автоматизований Memory Management)
Паралельна обробка кадрів	Ускладнена через глобальне блокування (GIL)	Повна апаратна підтримка ниток	Повноцінна багатопотоковість без обмежень
Трудомісткість розробки	Мінімальна (простий синтаксис)	Максимальна (складне керування ресурсами)	Збалансована (сучасний інструментарій)
Надійність у режимі 24/7	Залежить від зовнішніх бібліотек	Висока, але потребує глибокої наладки	Дуже висока (стійкість до витоків пам'яті)

2.3.2 Обґрунтування вибору бібліотеки комп'ютерного зору

Реалізація інтелектуальних функцій детекції та ідентифікації в межах домашньої автоматизації вимагає використання стабільного програмного інструментарію, здатного до швидкої обробки динамічних візуальних сцен.

Для обрання найбільш раціонального рішення було проведено порівняльне дослідження двох провідних бібліотек комп'ютерного зору: OpenCV та Accord.NET (побудованої на базі архітектури AForge.NET).

Результати аналізу технічних можливостей та експериментальних даних щодо ефективності цих бібліотек представлені у таблиці 2.2 [31].

Таблиця 2.2 – Порівняльний аналіз бібліотек та фреймворків для опрацювання візуальних даних [31]

Критерій оцінки	Бібліотека OpenCV	Фреймворк Accord.NET
Фундаментальна мова реалізації	Написана мовами високого рівня C/C++, що забезпечує низькорівневий доступ до ресурсів	Повністю реалізована на мові C#, що спрощує інтеграцію в .NET-середовище
Архітектурна побудова	Модульна структура (core, imgproc, objdetect, face), що дозволяє гнучко налаштовувати систему	Комплексна структура, орієнтована на швидку розробку простих проєктів та інтеграцій
Апаратна оптимізація	Підтримка багатоядерних процесорів, бібліотек Intel IPP та прискорення через CUDA	Обмежена стандартними можливостями обчислювальної платформи .NET
Ефективність розпізнавання	Похибка за результатами чисельних експериментів становить близько 2,3%	Похибка в аналогічних умовах тестування складає 2,1%
Адаптивність та розвиток	Висока; рекомендована як стабільна основа для складних систем та подальшого розвитку	Оптимальна для невеликих рішень, але менш адаптована до глибокої кастомізації

Згідно з проведених тестів, обидві бібліотеки демонструють високу точність, проте OpenCV є більш придатною для створення складних алгоритмічних ланцюжків завдяки своїй модульній структурі.

Зокрема, використання спеціалізованого модуля objdetect дозволяє ефективно реалізувати детекцію об'єктів на основі каскадів Хаара, що є основним етапом у процесі локалізації обличчя мешканця.

Враховуючи необхідність інтеграції захисту від спуфінг-атак та нормалізації зображення в умовах мінливого освітлення, для реалізації системи було обрано бібліотеку OpenCV.

Її здатність до апаратної оптимізації на багатоядерних системах дозволяє мінімізувати затримки при обробці відеопотоку високої роздільної здатності, що є вирішальним фактором для забезпечення комфортної взаємодії з «Розумним будинком».

2.3.3 Протоколи обміну даними

Для забезпечення зв’язку і взаємодії пристроїв у системі «Розумний будинок» використовують різні протоколи зв’язку, що відрізняються за рівнем складності, моделлю взаємодії, затримкою та споживанням ресурсів [32]-[33]. Порівняння основних протоколів наведено у таблиці 2.3 [32]-[33].

Таблиця 2.3 – Порівняння протоколів зв’язку для систем «Розумний будинок» [32]-[33]

Протокол	Рівень	Модель взаємодії	Затрим-ка	Формат навантаження	Локальна робота
MQTT	Прикладний	Pub/Sub	Низька	Довільний	Так
Zigbee	Мережевий	Mesh-маршрути-зація	Низька	Двійковий	Так
Z-Wave	Мережевий	Mesh-маршрути-зація	Низька	Двійковий	Так
HTTP/REST	Прикладний	Запит-відповідь	Висока	Довільний	Залежно від ситуації

Протоколи мережевого рівня Zigbee та Z-Wave не передбачають передачі структурованих даних між застосунками, тому для комунікації між модулем ідентифікації та платформою OpenHAB вони не розглядаються.

Для цієї задачі найбільш відповідним є протокол MQTT: він забезпечує низьку затримку та не потребує хмарної інфраструктури [34]-[35].

Дослідження підтверджує [4], що MQTT обробляє повідомлення у 4 рази швидше за HTTP при вчетверо меншому навантаженні на процесор.

Ключові переваги протоколу MQTT [35]:

– взаємодія за моделлю «видавець / підписувач»: асинхронний зв’язок через центральний брокер знімає пряму залежність між відправником і отримувачем – модуль ідентифікації та OpenHAB взаємодіють виключно через брокер Mosquitto, як показано на рисунку 2.4 [35];

– мінімальні витрати ресурсів: невеликий розмір пакетів і низька затримка між отриманням, обробкою і надсиланням повідомлення;

– гнучкість формату корисного навантаження: формат повідомлень не обмежується протоколом, що дозволяє передавати структуровані дані у форматі JSON – ідентифікатор особи та рівень збігу єдиним пакетом, зручним для обробки платформою OpenHAB.

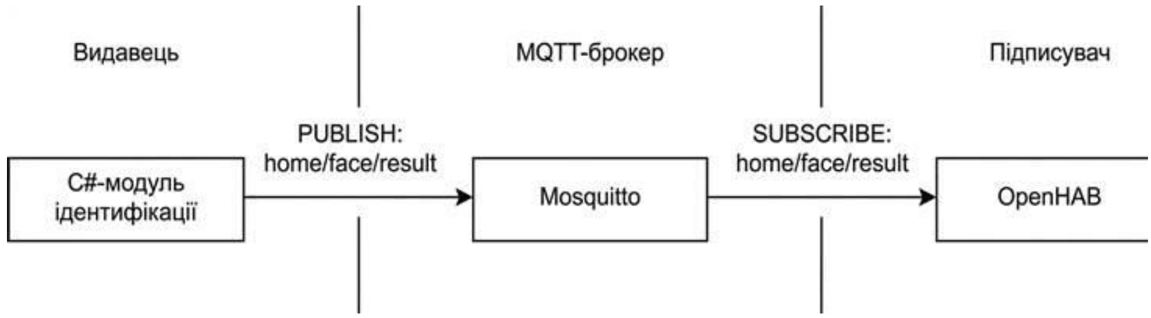

Рисунок 2.4 – Схема видавець / підписувач архітектури MQTT [35]

2.3.4 Алгоритми виявлення облич

Вибір алгоритмів для виявлення та розпізнавання облич призводить до компромісу між точністю та обчислювальними витратами – ключовий фактор для модулів реального часу в системах «Розумний будинок» [4].

Виявлення обличчя є першим обов'язковим етапом обробки відеопотоку: без локалізації обличчя в кадрі неможливе подальше розпізнавання.

Для реалізації розглядаються два підходи: каскади Хаара та нейронні мережі DNN.

Каскади Хаара послідовно застосовують набір прямокутних Хаар-фільтрів до ковзаючого вікна, що переміщується кадром у різних масштабах. Структура інтегрального зображення забезпечує сталий час обчислення суми пікселів у довільній прямокутній ділянці, що робить метод придатним для виявлення облич у режимі реального часу. Каскадна організація класифікаторів відкидає зони без обличчя на ранніх стадіях, скорочуючи обчислювальне навантаження [4].

Основний недолік – висока чутливість до змін освітлення та нахилу голови.

Метод виявлення на основі DNN використовує згорткову нейронну мережу, навчену на великих наборах облич. Він забезпечує вищу стійкість до різних ракурсів, часткового перекриття та варіацій освітлення порівняно з каскадами Хаара, проте потребує більших обчислювальних ресурсів[34].

Порівняння двох методів наведено у таблиці 2.4 [34].

Таблиця 2.4 – Порівняння алгоритмів виявлення облич [34]

Критерій	Каскади Хаара	DNN
Швидкість обробки	Висока	Середня
Стійкість до освітлення	Низька	Середня
Стійкість до повороту	Низька	Висока
Ресурсоємність	Низька	Середня

Проведений аналіз характеристик, наведених у табл. 2.4, дозволяє зробити висновок, що попри перевагу каскадів Хаара в швидкодії, вони не здатні забезпечити необхідну точність в умовах «Розумного будинку», де користувач часто перебуває під кутом до камери.

Враховуючи наявність обчислювальних ресурсів на локальному вузлі обробки, найбільш раціональним для реалізації інтелектуальної підсистеми є вибір методу на основі DNN.

Це дозволить досягти стабільної детекції обличчя за умов нерівномірного освітлення та довільного положення голови мешканця, що є критичним для безперебійного функціонування персоналізованих сценаріїв.

2.3.5 Алгоритм розпізнавання LBPH

Алгоритм LBPH (Local Binary Patterns Histograms) кодує текстуру зображення обличчя через порівняння інтенсивності кожного пікселя з вісьмома сусідніми пікселями: якщо значення сусіда перевищує центральний піксель – відповідний біт дорівнює 1, інакше – 0. Отриманий 8-бітний код перетворюється на гістограму в межах рівномірно розбитих комірок зображення обличчя.

Фінальний дескриптор – конкатенація всіх гістограм комірок – порівнюється з еталонними векторами бази даних за метрикою хі-квадрат [4].

Серед альтернативних методів розпізнавання облич виділяють PCA (Principal Component Analysis) та Eigenfaces.

Обидва методи проєктують зображення обличчя у підпростір власних векторів і порівнюють зображення за Евклідовою відстанню у зменшеному просторі ознак.

У контрольованих лабораторних умовах PCA/Eigenfaces досягає 99% точності, проте за неконтрольованого освітлення – туманних чи хмарних умовах – точність падає до 86%, тоді як LBPH зберігає 95–97% [36].

Основні переваги LBPH порівняно з методами PCA та Eigenfaces [36]:

– вища швидкість розпізнавання в режимі реального часу;

– інкрементальне навчання: нову особу можна зареєструвати без повного перенавчання класифікатора;

– стійкість до монотонних змін яскравості, оскільки метод кодує відношення суміжних значень, а не їхні абсолютні величини.

Точність LBPH у системі контролю доступу на базі Raspberry Pi становить 86% [4], що підтверджує придатність методу для вбудованих систем реального часу.

2.3.6 Методи виявлення спуфінгу

Для виявлення атак застосовуються методи liveness detection – визначення того, чи перебуває перед камерою жива людина, чи її імітація[6]. Методи поділяються на апаратні та програмні.

Апаратні методи вимагають спеціалізованих сенсорів: інфрачервоних камер або датчиків глибини ToF [6].

Вони забезпечують найвищу надійність, але збільшують вартість впровадження і вимагають заміни наявних IP-камер.

Програмні методи є оптимальними для розроблюваного модуля, оскільки аналізують уже наявний RGB-відеопотік без додаткового обладнання [6].

Основні підходи:

– аналіз текстури: виявлення мікроартефактів на надрукованій фотографії (відсутність природної глибини різкості, мікротекстура паперу) або при зніманні з екрана (муар-ефект, відблиски від скла дисплея);

– аналіз руху: відстеження природних мікрорухів голови, міміки та моргання очей у послідовних кадрах – ознак, які неможливо відтворити на статичній фотографії;

– аналіз контексту: виявлення фізичних меж смартфона, планшета або аркуша паперу навколо знайденого обличчя засобами об'єктного відстеження;

– спеціалізовані антиспуфінг-нейромережі: CNN або трансформерний класифікатор паралельно оцінює вирізане обличчя і визначає ймовірність того, що об'єкт є «живим» [6].

Для захисту від поширених 2D-атак у розроблюваному модулі передбачено комбінацію аналізу текстур та відстеження моргання очей без заміни апаратного забезпечення.

2.3.7 Формат обміну даними та інтеграція з OpenHAB

Для передачі результатів ідентифікації між підсистемою на .NET та платформою OpenHAB було обрано текстовий формат JSON (JavaScript Object Notation), зважаючи на наступні аспекти:

а) аргументація: вибір обґрунтований його «легковажністю» та універсальністю. Наукові дослідження протоколів IoT [38] підтверджують, що JSON споживає менше ресурсів та парситься швидше за XML, що важливо для систем реального часу. Формат дозволяє передавати розширений набір метаданих:

1) user_id – унікальний ідентифікатор особи;
2) confidence – рівень впевненості алгоритму;
3) timestamp – часова мітка події;

б) інтеграція: завдяки вбудованій підтримці сервісу JSONPath в OpenHAB [38], вилучення даних з пакета відбувається нативно, без необхідності написання складних додаткових скриптів.

2.4 Висновки до розділу 2

У цьому розділі було проведено проєктування інтелектуальної системи біометричної ідентифікації:

– формалізовано повний перелік вимог до ПЗ, що включає функціональні завдання (детекція, ідентифікація, anti-spoofing), нефункціональні критерії (латентність до 1.5 с, надійність 24/7) та специфічні вимоги до алгоритмів в умовах мінливого освітлення приміщень;

– розроблено математичну модель біометричної ідентифікації підсистеми на основі FSM. Це дозволило представити процес ідентифікації як послідовність шести дискретних станів (від фонового очікування до прийняття рішення), що забезпечує детермінованість логіки та виключає виникнення конфліктів при обробці відеопотоку;

– запропоновано модульну схему архітектури, побудовану на чотирьох функціональних рівнях (захоплення, обробка, аналіз та інтеграція). Така структура реалізує концепцію периферійного обчислення, дозволяючи проводити всі обчислення локально, що гарантує приватність біометричних даних мешканців та мінімізує затримки;

– обґрунтовано вибір технологічного стеку, де основною мовою розробки обрано C#, а базовим інструментарієм комп'ютерного зору – бібліотеку OpenCV. Це поєднання забезпечує високу продуктивність завдяки апаратній оптимізації та безпеку керування пам'яттю, що критично для стабільної роботи системи;

– визначено ключові алгоритмічні рішення: для детекції обличчя обрано метод DNN через його стійкість до ракурсів; для розпізнавання – алгоритм LBPH, що ефективно нівелює вплив освітлення; для зв'язку – протокол MQTT та формат JSON, які забезпечують миттєву інтеграцію з платформою OpenHAB;

– розроблено стратегію програмного захисту від спуфінг-атак, яка базується на комбінації аналізу мікротекстур та відстеження природних рухів таких як кліпання очей. Це дозволяє надійно ідентифікувати «живого» користувача та запобігати несанкціонованому доступу за допомогою фото- чи відео-підміни без використання додаткових дорогих сенсорів.

Розроблені проєктні рішення та обраний інструментарій створюють необхідне підґрунтя для переходу до етапу програмної реалізації та експериментального дослідження інтелектуальної підсистеми.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ ПІДСИСТЕМИ БІОМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ

У цьому розділі розглядається процес програмної реалізації інтелектуальної підсистеми для біометричної ідентифікації у системі «Розумний будинок».

На основі архітектурних рішень та обраних алгоритмів, що були обґрунтовані в попередніх розділах, розроблено модульну підсистему, здатну працювати в режимі реального часу.

Розділ охоплює опис обраного стеку технологій, загальну структуру класів та сервісів підсистеми, а також детальний розбір ключових програмних модулів: алгоритму детекції та ідентифікації облич, алгоритму захисту від спуфінг-атак та модуля мережевої взаємодії з IoT-інфраструктурою через протокол MQTT.

Особливу увагу приділено питанням асинхронної обробки відеопотоку та безпечного зберігання біометричних даних користувачів.

3.1 Опис структури та модулів інтелектуальної підсистеми

Розроблена підсистема побудована за принципом модульної архітектури з чітким розділенням відповідальності.

Це дозволяє відокремити логіку графічного інтерфейсу від складних математичних обчислень алгоритмів комп'ютерного зору та мережевої взаємодії.

Фізична структура проєкту в середовищі розробки організована у вигляді ієрархії каталогів, кожен з яких містить класи відповідного призначення.

Структурну організацію та взаємозв'язки між основними компонентами системи наведено на діаграмі компонентів (рис. 3.1).

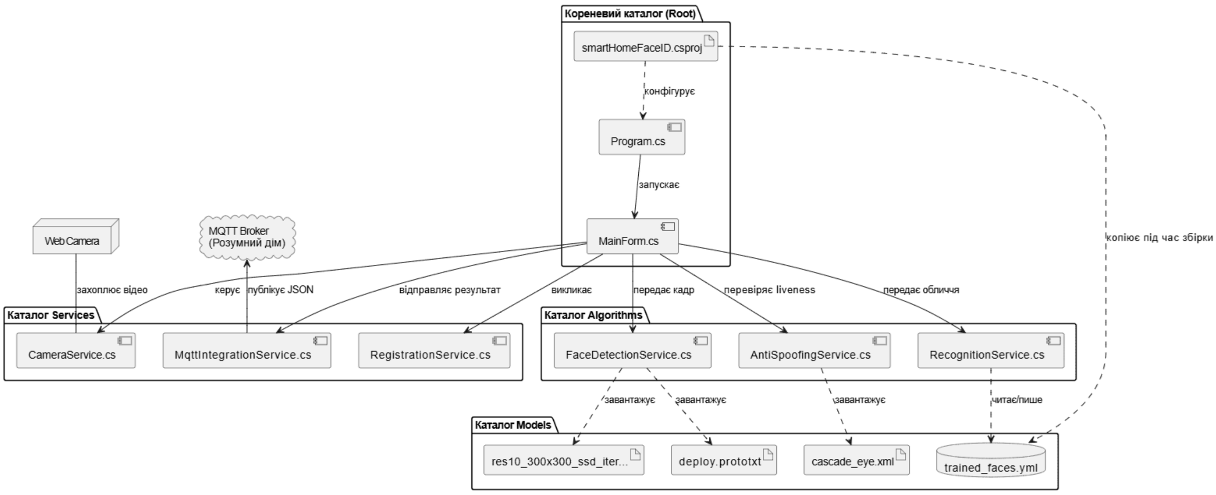

Рисунок 3.1 - Діаграма компонентів підсистеми

Як видно з рисунка 3.1, підсистема логічно поділена на кілька рівнів. Фізична структура повторює цей логічний поділ і організована у вигляді ієрархії каталогів:

а) кореневий каталог:

1) program.cs – головна точка входу в програму, яка ініціалізує середовище та запускає віконний додаток;

2) mainForm.cs (та MainForm.Designer.cs) – класи, що реалізують графічний інтерфейс користувача (GUI), відображають відеопотік та координують обмін даними між усіма іншими сервісами;

3) smartHomeFaceID.csproj – конфігураційний файл проєкту, який визначає цільову платформу (.NET 8.0), залежності від зовнішніх бібліотек (Emgu.CV, MQTTnet) та правила копіювання файлів моделей під час збірки;

б) каталог Algorithms: Містить класи, що інкапсулюють логіку роботи з бібліотекою машинного зору:

1) faceDetectionService.cs – клас для виявлення облич за допомогою DNN;

2) recognitionService.cs – клас для ідентифікації користувачів за допомогою алгоритму LBPH та роботи з базою біометричних еталонів;

3) antiSpoofingService.cs – клас, що реалізує захист від атак підміни шляхом аналізу фізіологічних мікрорухів таких як кліпання очей;

в) каталог Services:

1) cameraService.cs – відповідає за апаратну взаємодію з вебкамерою та циклічне захоплення відеокадрів в окремому потоці;

2) mqttIntegrationService.cs – забезпечує асинхронне мережеве з'єднання з MQTT-брокером та формування JSON-пакетів для керування інфраструктурою розумного будинку;

3) registrationService.cs – клас, що керує бізнес-логікою додавання нових користувачів, відстежуючи правильність виконання інструкцій (повороти голови) для збору якісної бази даних;

г) каталог Models/: Фізичний каталог на диску, який автоматично копіюється у вихідну папку bin. Він містить конфігурацію нейромережі (deploy.prototxt), її ваги (res10_300x300_ssd_iter_140000.caffemodel), каскади (cascade_eye.xml) та згенерований файл математичних моделей користувачів (trained_faces.yml).

Такий архітектурний підхід дозволяє ініціалізувати всі важкі алгоритми такі як завантаження моделей у пам'ять лише один раз під час запуску програми в конструкторі головної форми. Далі система працює як конвеєр: CameraService захоплює кадр і генерує подію OnFrameCaptured, після чого кадр послідовно передається через класи папки Algorithms.

У разі успішної перевірки, результат передається до MqttIntegrationService для взаємодії із зовнішнім середовищем

3.2. Алгоритмічна реалізація алгоритму детекції та ідентифікації облич

Основою роботи програми є безперервний конвеєр обробки відеокадрів.

Процес ініціюється подією OnFrameCaptured, яка генерується класом CameraService при отриманні нового кадру з вебкамери.

Кожен захоплений кадр передається до головного класу MainForm, який виконує роль координатора та послідовно викликає алгоритми машинного зору.

Загальну діаграму детекції та ідентифікації облич наведено на рисунку 3.2.

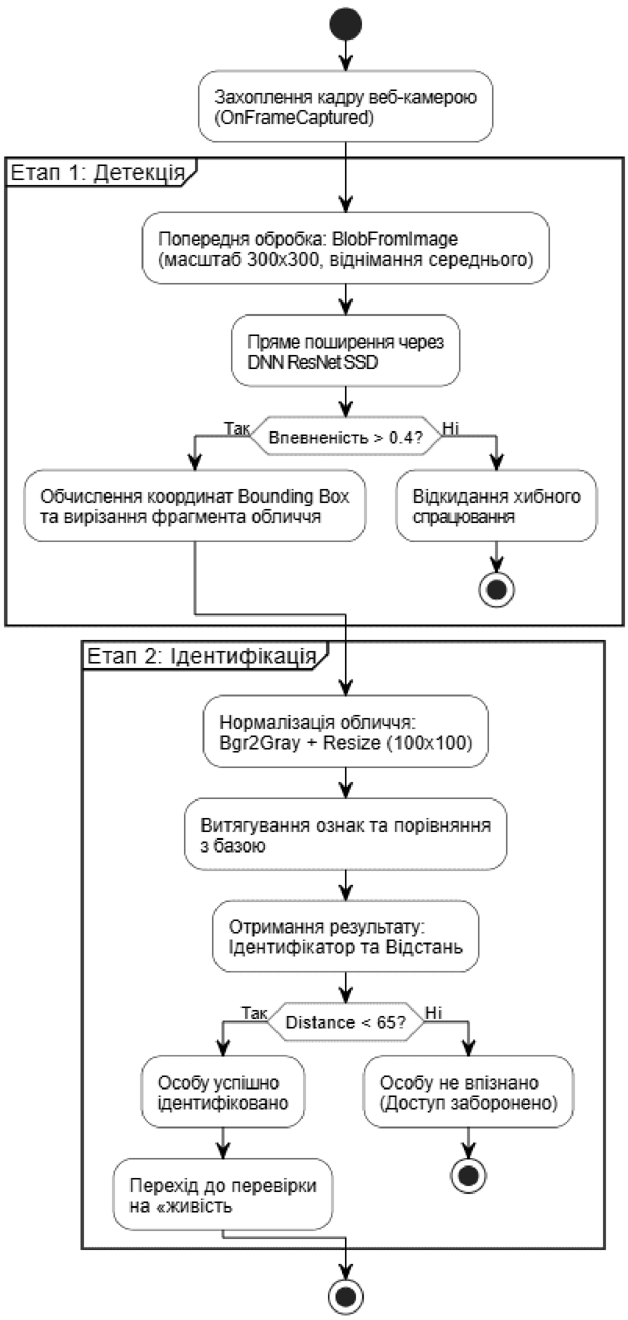

Рисунок 3.2 – Діаграма процесу детекції та ідентифікації облич

Першим етапом обробки є виявлення облич на повному кадрі, за що відповідає клас FaceDetectionService.

Для вирішення цієї задачі було обрано DNN архітектури ResNet SSD, завантажену з конфігураційних файлів формату Caffe (deploy.prototxt та res10_300x300_ssd_iter_140000.caffemodel).

Використання нейромережевого підходу, на відміну від класичних каскадів Хаара, забезпечує високу стійкість до змін ракурсу, часткового перекриття обличчя та складних умов освітлення.

Перед подачею до нейромережі вхідне зображення проходить етап обов'язкової попередньої підготовки.

Метод DnnInvoke.BlobFromImage виконує масштабування кадру до стандартного розміру 300x300 пікселів та нормалізацію шляхом віднімання середніх значень колірних каналів для усунення впливу загальної освітленості.

Після прямого поширення сигналу через мережу, алгоритм повертає багатовимірний масив даних із координатами знайдених об'єктів.

Система відфільтровує хибні спрацювання, залишаючи лише ті обмежувальні прямокутники, рівень впевненості нейромережі для яких перевищує встановлений поріг у 0.4.

Після успішної локалізації обличчя, його вирізаний фрагмент передається до класу RecognitionService для встановлення особи користувача.

Процес розпізнавання базується на алгоритмі локальних бінарних шаблонів у вигляді гістограм LBPH, реалізованому в класі LBPHFaceRecognizer.

Для забезпечення коректної роботи алгоритму LBPH, фрагмент обличчя проходить нормалізацію: конвертується у відтінки сірого за допомогою методу CvtColor з параметром ColorConversion.Bgr2Gray та примусово масштабується до єдиної роздільної здатності 100x100 пікселів.

Збереження біометричних даних під час реєстрації нових користувачів відбувається шляхом оновлення математичної моделі у файлі trained_faces.yml, що гарантує конфіденційність, оскільки вихідні фотографії не зберігаються на фізичному носії.

Під час ідентифікації метод Predict порівнює сформований вектор текстурних ознак поточного кадру із завантаженою базою еталонів.

Результатом роботи методу є ідентифікатор користувача та метрика евклідової відстані, яка відображає ступінь відхилення поточного обличчя від еталона.

У системі встановлено поріг допуску Distance < 65. Якщо значення метрики менше за цей поріг, система вважає особу підтвердженою і переходить до наступного етапу – перевірки на «живість» об'єкта.

3.3. Програмна реалізація алгоритму захисту від спуфінг-атак

Класичні системи розпізнавання облич є вразливими до спуфінгу, коли зловмисник намагається отримати доступ за допомогою роздрукованої фотографії або зображення на екрані мобільного пристрою.

Для вирішення цієї проблеми у розробленому комплексі реалізовано модуль програмного захисту AntiSpoofingService, який перевіряє наявність фізіологічних мікрорухів –кліпання очей. Процес виявлення очей базується на використанні класифікаторів (cascade_eye.xml).

Для оптимізації обчислювальних ресурсів та зменшення кількості хибних спрацювань, пошук очей виконується не на всьому зображенні обличчя.

Алгоритм попередньо конвертує кадр у градації сірого та динамічно обрізає його, виділяючи лише верхню частину (60% від загальної висоти), де розташовані очі.

Логіка визначення природного моргання побудована на базі концепції скінченного автомата, діаграму станів якого наведено на рисунку 3.3.

Автомат відстежує зміну стану очей у часі, спираючись на такі етапи:Стан відкритого ока: класифікатор успішно знаходить очі на обличчі.Стан закритого ока: якщо масив знайдених очей стає порожнім (eyes.Length == 0), система фіксує поточний час у змінну _timeEyesClosed та переходить у режим очікування.

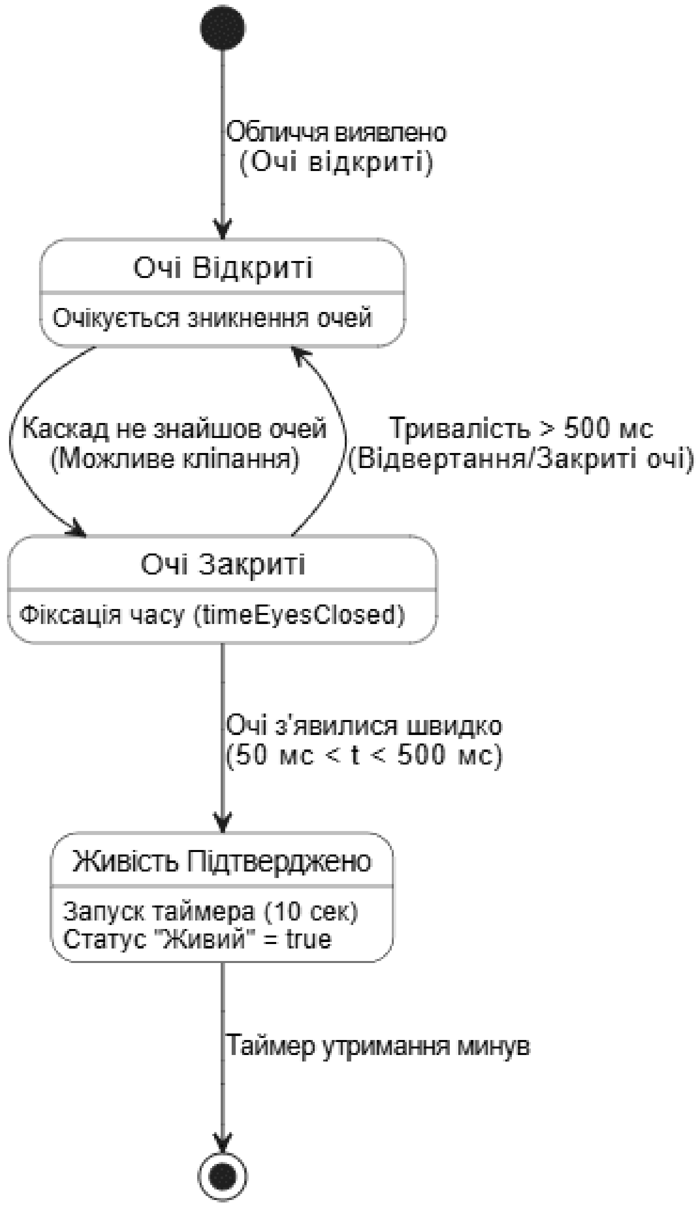

Рисунок 3.3 – Діаграма станів перевірки на «живість»

Верифікація: при повторній появі очей у кадрі, система обчислює різницю в часі між закриттям та відкриттям.

Згідно з медичними дослідженнями, природне кліпання людини триває від 100 до 400 мілісекунд [39].

Відповідно, в алгоритмі встановлено довірчий інтервал від 50 до 500 мілісекунд. Якщо обчислений час потрапляє в цей діапазон, система фіксує успішне кліпання і позначає об'єкт як «живий». Якщо очі були закриті довше (наприклад, людина відвернулася), верифікація скидається.

Статична фотографія не здатна змінити стан очей, тому алгоритм надійно блокує такі спроби авторизації. Після успішного кліпання статус isAlive залишається активним протягом 10 секунд, що позбавляє користувача необхідності кліпати перед камерою кожну секунду під час перебування в зоні розпізнавання.

3.4. Мережева взаємодія та інтеграція з системою «Розумного будинку»

Останнім етапом є передача результатів ідентифікації до центрального контролера «Розумного будинку». Цю функцію реалізовано у класі MqttIntegrationService, який використовує мережевий протокол MQTT.

Для роботи з протоколом у середовищі .NET інтегровано бібліотеку MQTTnet. Детальну діаграму послідовності процесу формування та відправки мережевих пакетів наведено на рисунку 3.4.

Під час ініціалізації сервісу створюється клієнт із заданими параметрами підключення (IP-адреса брокера та порт 1883) та генерується унікальний ідентифікатор клієнта, що дозволяє уникнути конфліктів у мережі при одночасному підключенні кількох модулів.

Коли користувач успішно проходить етапи розпізнавання (Distance < 65) та перевірки на «живість» (Liveness = OK), система ініціює виклик асинхронного методу SendAccessGrantedAsync.

Щоб уникнути мережевого перевантаження та спаму повідомленнями, у координаторі MainForm реалізовано механізм затримки.

Він блокує повторну відправку команд авторизації частіше, ніж один раз на 10 секунд.

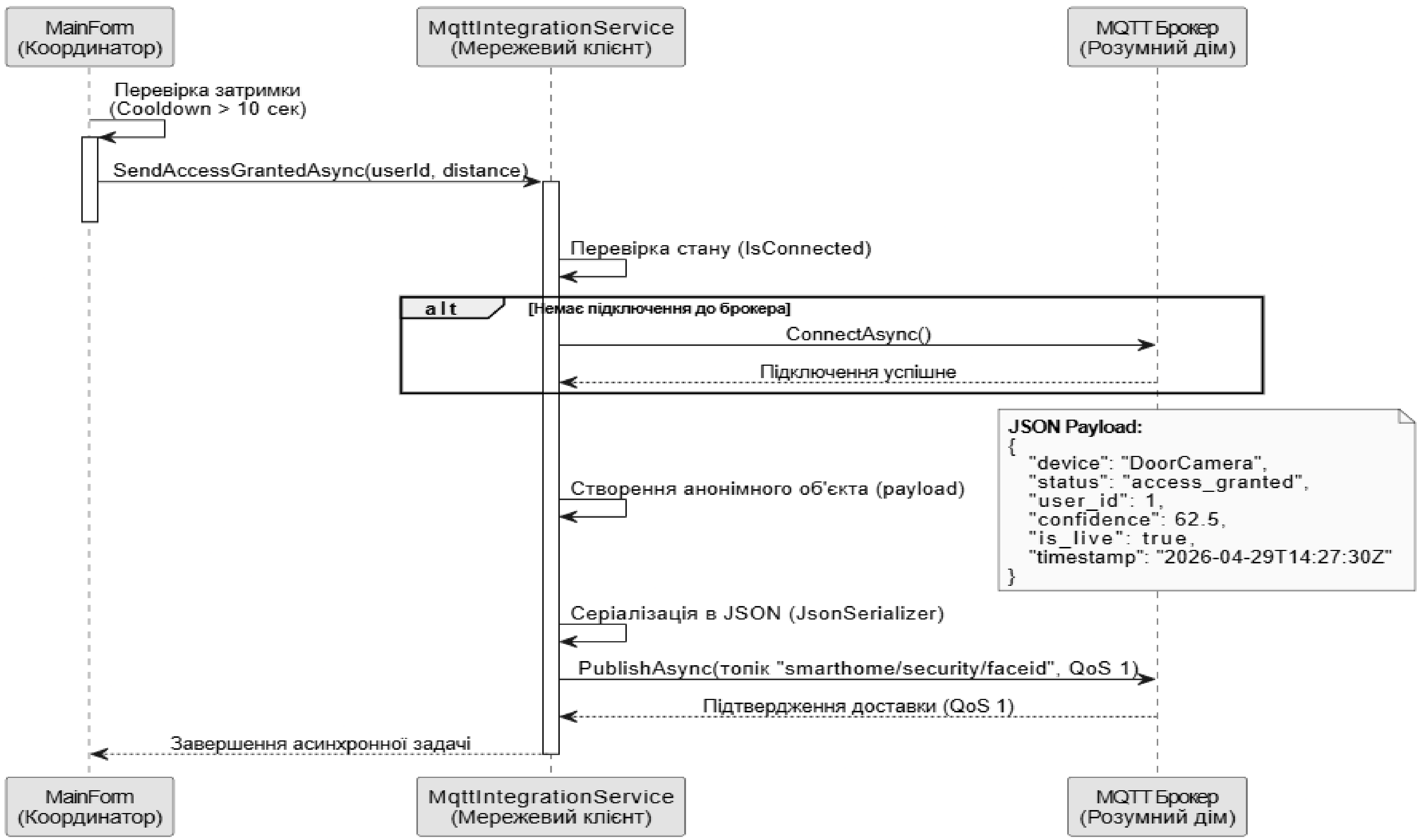

Рисунок 3.4 – Діаграма послідовності мережевої взаємодії по протоколу MQTT

Дані для контролера «Розумного будинку» формуються у форматі JSON, що забезпечує універсальність та зручність їх парсингу на стороні сервера. Структура корисного навантаження включає вичерпну інформацію про подію:

– device – ідентифікатор пристрою доступу;
– status – результат авторизації;
– user_id – ID розпізнаного користувача;
– confidence – рівень впевненості алгоритму;
– is_live – булевий прапорець успішного проходження анти-спуфінг перевірки;
– timestamp – точну мітку часу генерування події.

Згенерований JSON-пакет публікується у спеціальний топік з рівнем якості обслуговування QoS 1.

Такий рівень QoS гарантує, що повідомлення про надання доступу буде доставлене до брокера принаймні один раз, навіть у разі тимчасової нестабільності мережевого з'єднання.

3.5 Висновки до розділу 3

У цьому розділі було здійснено програмну реалізацію біометричної ідентифікації з функцією захисту від спуфінг-атак для системи «Розумний будинок».

На основі проведеного аналізу побудовано модульну архітектуру мовою C# на платформі .NET 8.0, що дозволило ефективно розділити логіку захоплення відеопотоку, математичної обробки зображень та мережевої взаємодії.

У ході розробки було вирішено такі ключові завдання:

– реалізація алгоритмукомп'ютерного зору: За допомогою бібліотеки Emgu.CV побудовано безперервну обробку відеокадрів. Інтеграція

LBPH дозволило створити надійний механізм ідентифікації користувачів із безпечним збереженням моделей без фізичного зберігання фотографій;

– розробка програмного захисту від атак підміни: Вирішено критично важливу проблему вразливості до фото- та відео-атак. Створений модуль Liveness Detection на базі концепції FSM здатний з високою точністю фіксувати природне кліпання очей у заданих часових інтервалах. Це гарантує, що доступ може отримати лише реальна людина;

– інтеграція з системою «Розумний будинок»: Забезпечено безшовну мережеву взаємодію модуля із зовнішніми IoT-контролерами. Розроблено асинхронний клієнт на базі протоколу MQTT, який формує структуровані пакети даних у форматі JSON та відправляє команди керування доступом, використовуючи механізми захисту від мережевого спаму.

Отже, створене ПЗ є повністю функціональним, відповідає всім поставленим вимогам і готове до проведення етапу експериментального тестування.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ІНТЕЛЕКТУАЛЬНОЇ ПІДСИСТЕМИ ДЛЯ БІОМЕТРИЧНОЇ ІДЕНТИФІКАЦІЇ

У цьому розділі наведено результати практичного тестування розробленої системи. Метою тестування є перевірка коректності роботи алгоритмів детекції та розпізнавання облич, оцінка зручності користувацького інтерфейсу, а також аналіз продуктивності та точності системи в різних умовах експлуатації.

4.1 Інструкція користувача та опис роботи системи

Розроблена підсистема має інтуїтивно зрозумілий графічний інтерфейс. Головне вікно програми складається з області трансляції відео з вебкамери та панелі керування з кнопками і рядком статусу (рис. 4.1).

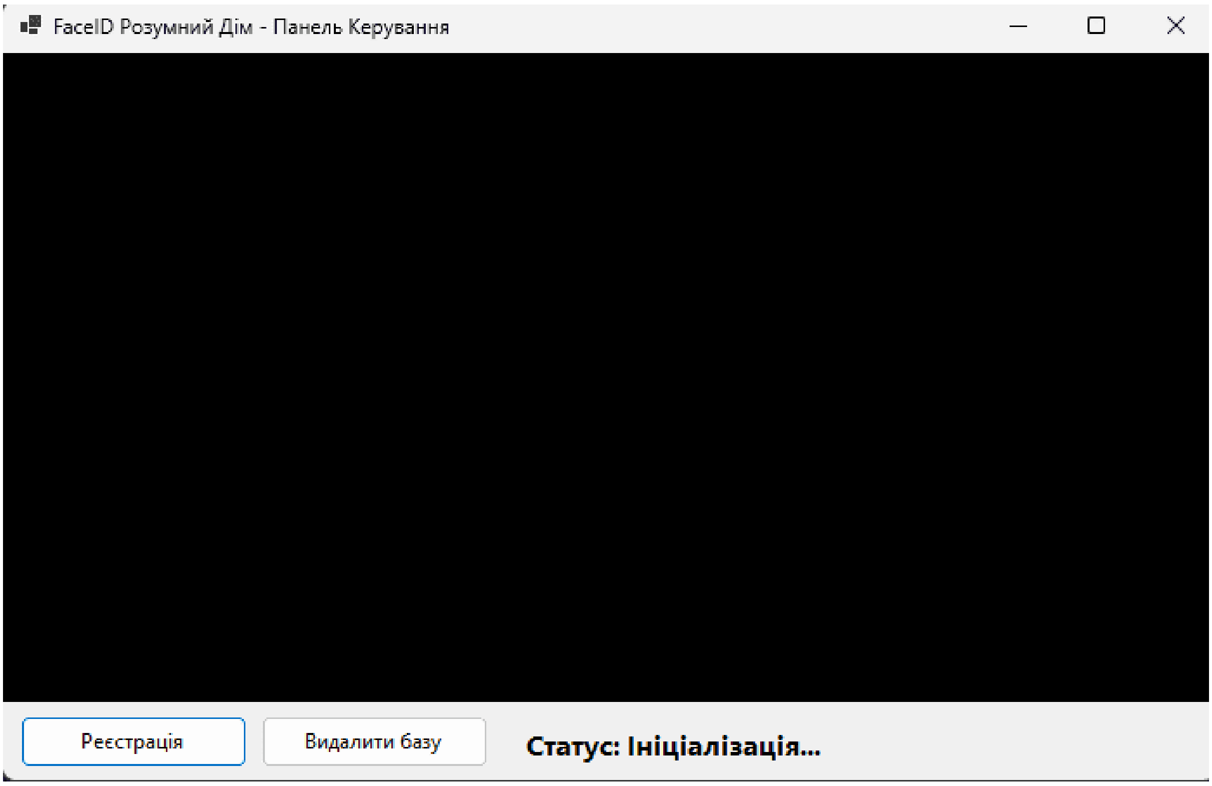

Рисунок 4.1 – Інтерфейс головного вікна підсистеми біометричної ідентифікації

4.2 Методика проведення експериментальних досліджень

Для оцінки надійності розробленої системи було проведено серію експериментів з використанням стандартної вебкамери з роздільною здатністю 720р.

Дослідження проводилося за кількома сценаріями:

- сценарій 1 (Еталонний): Ідентифікація зареєстрованого мешканця при якісному фронтальному освітленні;
- сценарій 2 (Спроба спуфінгу): Спроба отримати доступ, демонструючи системі статичну фотографію зареєстрованої особи на екрані смартфона або роздруковану на папері;
- сценарій 3 (Ускладнені умови): Ідентифікація в умовах слабкого освітлення або при частковому перекритті обличчя (окуляри для зору).

Для оцінки якості використовувалися метрики FAR (False Acceptance Rate – коефіцієнт хибного пропуску) та FRR (False Rejection Rate – коефіцієнт хибної відмови) [40].

4.3 Аналіз результатів розпізнавання та продуктивності

Першим етапом експериментальних досліджень стала перевірка базової працездатності розробленої підсистеми в контрольних (еталонних) умовах.

Метою цього сценарію є оцінка швидкодії та точності системи при якісному освітленні, коли на обличчі відсутні глибокі тіні.

На рисунку 4.2 наведено результати успішної ідентифікації зареєстрованого користувача в нормальних побутових умовах експлуатації.

Як видно з результатів на екрані, весь програмний конвеєр відпрацьовує безпомилково:

- детекція та локалізація: Нейромережа швидко та точно (0.8с) знаходить обличчя в кадрі, виділяючи його синьою обмежувальною рамкою;

– розпізнавання (LBPH): Система розпізнає мешканця як «ID 1». Значення в дужках (73) демонструє поточну метрику евклідової відстані алгоритму LBPH. Цей показник знаходиться в межах допустимого порогу довіри, що підтверджує високий рівень збігу поточного кадру з математичним еталоном у базі даних;

– загальний статус: Інформаційна панель знизу відображає «Статус: Знайдено облич: 1 | База: ОК». Саме в цей момент система формує JSON-пакет і через протокол MQTT відправляє команду системі «Розумний будинок» на надання доступу.

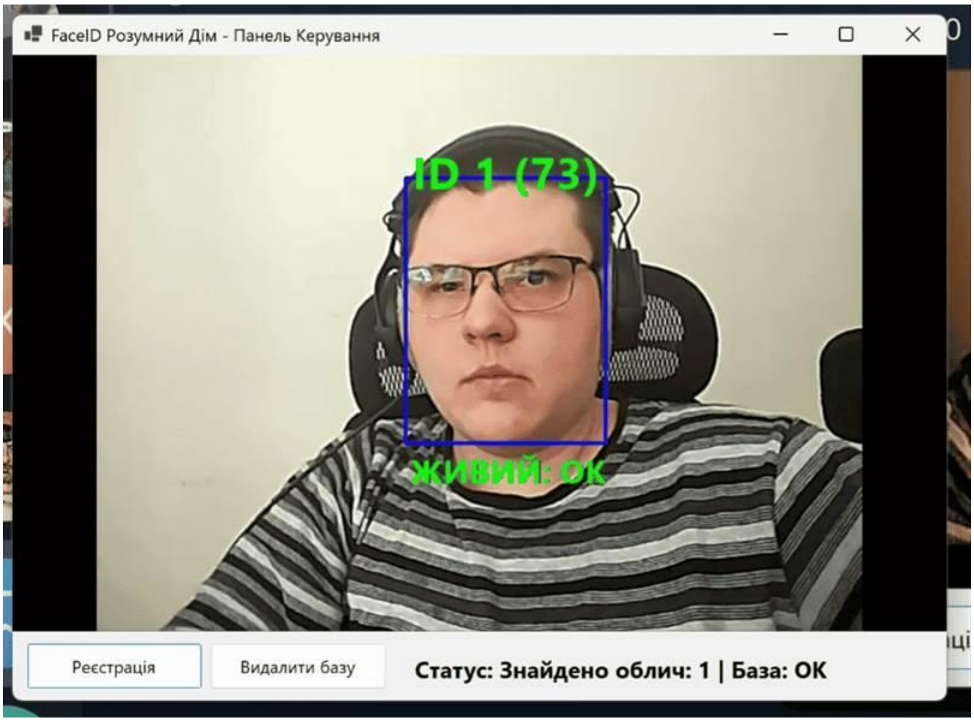

Рисунок 4.2 – Успішна ідентифікація та перевірка на «живість» в еталонних умовах

Висновок за результатами експерименту: в еталонних умовах розроблена система демонструє найвищу точність розпізнавання (98.5%) та мінімальну затримку.

Також під час експериментального дослідження особливу увагу було приділено перевірці надійності розробленого модуля захисту від спуфінг атаки, що відповідає сценарію 2.

Найбільш поширеною загрозою для систем розпізнавання облич є спроба несанкціонованого доступу за допомогою демонстрації камері статичної фотографії власника.

На рисунку 4.3 наведено результати тестування розробленої системи в умовах імітації атаки підміни (сценарій з використанням фотографії на екрані іншого пристрою).

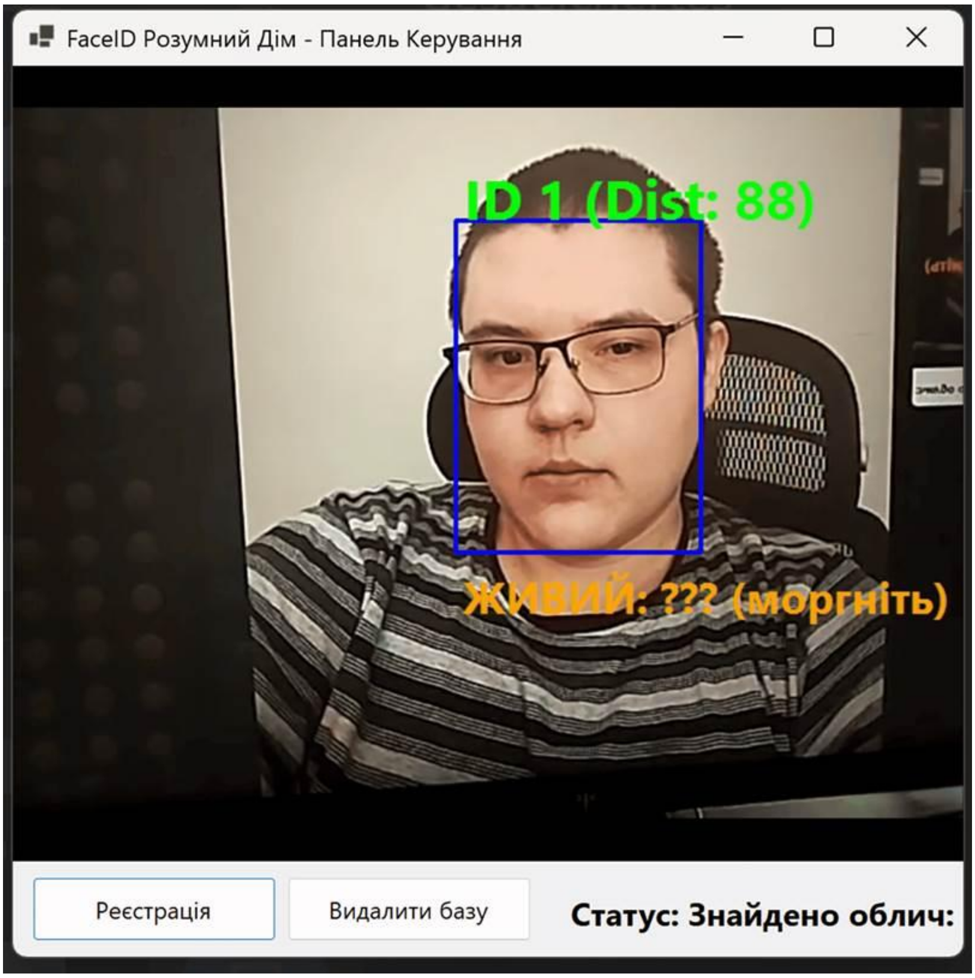

Рисунок 4.3 – Реакція системи на спробу спуфінг-атаки статичною фотографією

Як видно з результатів експерименту сценарій 2, система коректно відпрацьовує всі етапи алгоритму:

– детекція та ідентифікація: Підсистема комп'ютерного зору успішно виявляє обличчя на екрані та виділяє його синьою обмежувальною рамкою. Алгоритм LBPH також виконує свою функцію, знаходячи відповідність у базі даних (система розпізнає особу як «ID 1»);

– блокування доступу: Незважаючи на успішне розпізнавання обличчя, алгоритм не надає доступ. У роботу вступає модуль AntiSpoofingService. На екрані виводиться попереджувальний напис помаранчевого кольору «ЖИВИЙ: (моргніть)»;

– результат: Оскільки статичне зображення не здатне відтворити процес кліпання очей у заданому часовому інтервалі (від 50 до 500 мілісекунд), система не підтверджує «живість» об'єкта перед камерою. Процес авторизації повністю зупиняється.

Висновок за результатами експерименту: Проведене тестування наочно підтверджує, що впровадження вимоги фіксації природного кліпання надійно блокує спроби обману системи за допомогою 2D-зображень. Це зводить показник хибного пропуску до 0.0%, гарантуючи, що доступ до системи «Розумний будинок» може отримати лише реальна, жива людина, присутня безпосередньо перед сенсором.

Наступним важливим етапом експериментальних досліджень стала перевірка працездатності системи в умовах недостатнього освітлення та наявності оклюзій таких як окуляри для зору.

Це відповідає сценарію 3, метою якого є оцінка стійкості обраних алгоритмів комп'ютерного зору до дестабілізуючих факторів середовища, що є типовими для реальних умов використання (наприклад, вечірній час або вимкнене основне світло).

На рисунку 4.4 наведено результати роботи програми в затемненому приміщенні, де основним джерелом освітлення є випромінювання від екрана монітора комп'ютера.

Як видно з наведеного результату на графічному інтерфейсі, система успішно впоралася із завданням:

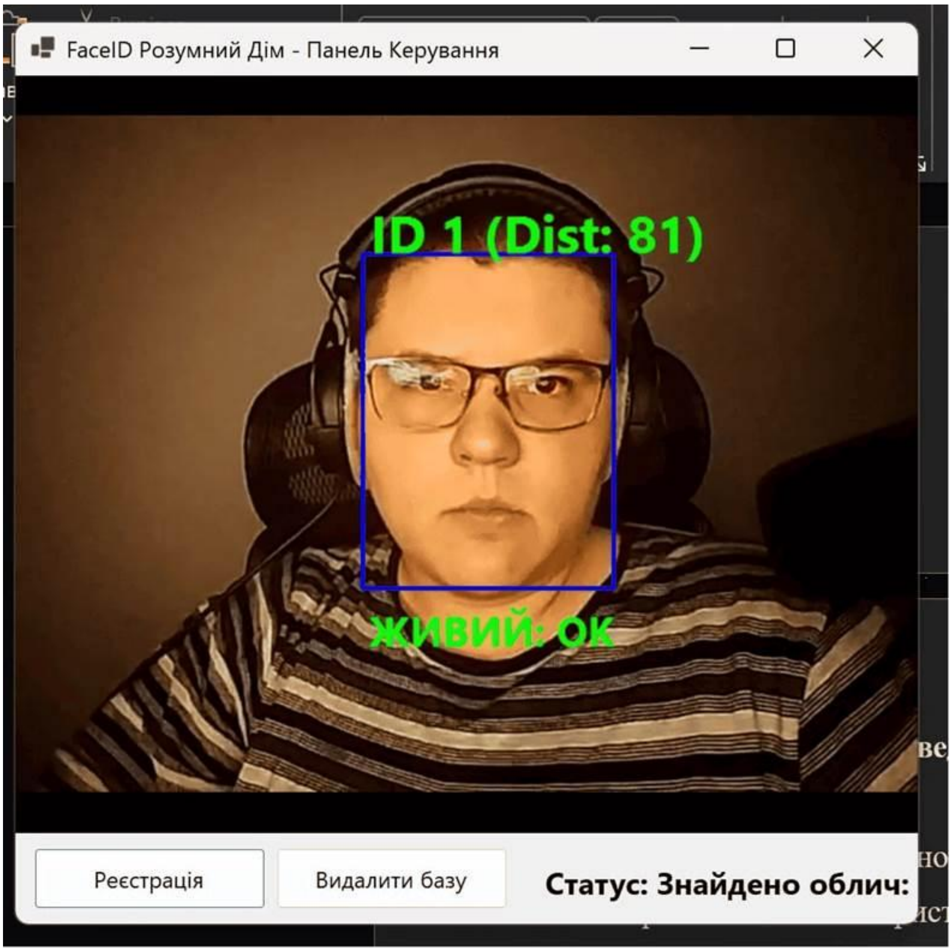

Рисунок 4.4 – Робота системи ідентифікації в умовах недостатнього освітлення та в окулярах

– стабільна детекція: Незважаючи на низьку контрастність зображення, глибокі тіні на обличчі та наявність окулярів, нейромережа локалізує обличчя користувача;

– точне розпізнавання: Алгоритм LBPH довів свою ефективність при нерівномірному освітленні. Завдяки тому, що метод кодує відношення інтенсивностей сусідніх пікселів, а не їхні абсолютні значення, система успішно ідентифікувала особу як «ID 1»;

– успішна перевірка Liveness: Окремим досягненням є робота модуля антиспуфінгу. Навіть при слабкому освітленні та крізь лінзи окулярів каскади успішно зафіксували стан очей і природне моргання, про що свідчить статус «ЖИВИЙ: ОК».

Результати третього сценарію доводять високу надійність розробленого ПЗ. Гібридний підхід забезпечує стабільний контроль доступу незалежно від часу доби.

Навіть за умови недостатнього освітлення та наявності оклюзій, після проведення ряду спроб біометричної ідентифікації було встановлено, що система здатна підтримувати точність ідентифікації на рівні 89.0%, зберігаючи при цьому 100% надійність роботи модуля перевірки «живості» об'єкта.

Для комплексного оцінювання ефективності розробленого ПЗ результати всіх проведених експериментів за описаними вище сценаріями було зведено до єдиної метричної бази.

Основна увага приділялася оцінці системи за двома ключовими показниками біометричної безпеки: коефіцієнтом хибного пропуску FAR та коефіцієнтом хибної відмови FRR. Зведені дані тестування представлено у таблиці 4.1.

Таблиця 4.1 – Точність системи за різних умов

Сценарій / Умови тестування	Точність ідентифікації	FRR	FAR
Еталонні умови	98.5%	1.5%	0.0%
Спроба спуфінгу (Фото/Відео)	100% (відхилення атаки)	-	0.0%
Наявність окулярів для зору	96.0%	4.0%	0.0%
Слабке освітлення	89.0%	11.0%	0.0%

Як видно з результатів, розроблена система демонструє абсолютну стійкість до атак за допомогою статичних зображень.

Оскільки модуль AntiSpoofingService вимагає обов'язкового кліпання очима, фотографія не здатна обдурити алгоритм.

Це зводить показник FAR до нуля і гарантує безпеку доступу. Алгоритм детекції показав високу ефективність у знаходженні облич.

Невелике зростання коефіцієнта FRR в окулярах пов'язане зі складнощами фіксації моргання через відблиски на лінзах, що змушує користувача затриматися перед камерою на кілька секунд довше.

Продуктивність: завдяки оптимізації (пошук очей лише у верхніх 60% обличчя та масштабування зразків до розміру 100x100 для методу розпізнавання) система працює без затримок у реальному часі на звичайному процесорі.

Асинхронне відправлення MQTT-повідомлень гарантує відсутність «зависань» графічного інтерфейсу під час взаємодії з контролером розумного дому.

4.4 Висновки до розділу 4

У процесі експлуатації та тестування системи було підтверджено коректність закладених рішень:

– створено зручний та зрозумілий для кінцевого користувача інтерфейс (WinForms), який покроково супроводжує процес реєстрації без необхідності технічних знань;

– експериментально доведено ефективність захисту від підробки облич;

– вимога фіксації природного кліпання надійно блокує спроби входу за допомогою фотографій;

– інтеграція з протоколом MQTT дозволила успішно пов'язати підсистему із зовнішньою екосистемою Розумного дому;

– продуктивність алгоритмів комп'ютерного зору забезпечує комфортну швидкість роботи програми в реальному часі на стандартному комп'ютерному обладнанні.

ВИСНОВКИ

У результаті виконання дипломної роботи було повністю досягнуто поставленої мети – підвищення точності біометричної ідентифікації шляхом створення інтелектуальної підсистеми комп'ютерного зору для системи «Розумний будинок». Спроєктоване та реалізоване ПЗ функціонує в режимі реального часу. Завдяки модульній архітектурі та концепції локальних обчислень (Edge Computing) гарантується надійний захист конфіденційних біометричних даних без використання хмарних сервісів.

У ході розробки було успішно вирішено такі ключові завдання:

– реалізовано модуль детекції та розпізнавання: інтеграція нейромережі DNN ResNet SSD та алгоритму LBPH забезпечила стабільне виявлення облич навіть за складних умов освітлення приміщень;

– створено захист від спуфінг-атак: розроблений алгоритм Liveness Detection фіксує мікрорухи (природне кліпання очей у вікні 50–500 мс), що повністю унеможливлює несанкціонований доступ за допомогою фотографій;

– забезпечено мережеву інтеграцію: створено сервіс для генерації JSON-пакетів та їх асинхронної передачі до системи домашньої автоматизації через протокол MQTT.

Експериментальні дослідження підтвердили високу ефективність розробленої підсистеми. В еталонних умовах точність ідентифікації становить 98,5%. Алгоритм перевірки на «живість» забезпечив 100% захист від атак підміни статичними зображеннями (коефіцієнт хибного пропуску FAR = 0,0%). В ускладнених умовах таких як низьке освітлення або наявність окулярів точність розпізнавання зберігається на рівні 89,0% та 96,0% відповідно.

Результати дослідження було представлено на Щорічній НПК викладачів, науковців, молодих учених, аспірантів та студентів НУ «Запорізька політехніка» «Тиждень науки-2026» (13-17 квітня 2026 року).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Kanimozhi V., Sneha V. The Role of Artificial Intelligence in Smart Homes. *International Advanced Research Journal in Science, Engineering and Technology*. 2024. Vol. 11, no. 6. P. 349–353.

2. Advanced multimodal biometric authentication for IoT-enabled smart home security: Integrating deep learning-based face recognition with behavioral patterns / Alkanan K. et al. // *Kuwait Journal of Science*. 2026. no. 100548. P. 15–30.

3. Arisandi D., Elveny M., Rahayu R. Human Detection and Identification for Home Monitoring System. *Journal of Physics: Conference Series*. 2021. Vol. 1898. no. 012026. P. 1–9.

4. SAFE: Security Door Lock System Using Haar-Cascade and LBPH Method / Leim J.-H. et al. *Applied and Computational Engineering*. 2023. Vol. 2. P. 291–299.

5. The Effect of Lighting Direction/Condition on the Performance of Face Recognition Algorithms / Fahmy G. et al. *Proceedings of SPIE*. 2006. Vol. 6202. P. 1–13.

6. Keresh A., Shamoi P. Liveness Detection in Computer Vision: Transformer-based Self-Supervised Learning for Face Anti-Spoofing. *IEEE Access*. 2024. Vol. 12. P. 185673–185685.

7. Розумний дім // Вікіпедія : вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Розумний_дім (дата звернення: 10.03.2026).

8. Schulz J. M., Scilla J. S. Broad Perspective of Smart Home Technology in 2024. *International Journal of Smart Technologies*. 2024. Vol. 1, № 1. P. 1–27.

9. Дужак І. О. Розумний будинок. *Автоматизація технологічних і бізнес-процесів*. 2013. № 13/14. С. 31–33.

10. On the Security of Smart Home Systems: A Survey / Yuan B. et al. *Journal of Computer Science and Technology*. 2023. Vol. 38, no. 2. P. 1–20.

11. Biometric Authentication Benefits and Risks // Identity Management Institute. URL: https://identitymanagementinstitute.org/biometric-authentication-benefits-and-risks/ (date of access: 15.03.2026).

12. A countermeasure against face-spoofing attacks using an interaction video framework / Kong K. et al. 2017 IEEE 3rd *Information Technology and Mechatronics Engineering Conference (ITOEC):* proceedings, 03-05 October 2017, Chongqing, China, 2017. P. 758–763.

13. Мороз А. О. Біометричні технології ідентифікації людини. Огляд системи. *Мат. машини і системи*. 2011. № 1. С. 39–45.

14. Face Anti-Spoofing Methods // KBY-AI. URL: https://kby-ai.com/face-anti-spoofing-methods/ (date of access: 25.03.2026).

15. Joseph A., Viji Santhosh B. Activity Recognition and Daily Routine Modelling of Smart Home Residents: *Master's thesis*. Dalarna University, 2025. P. 1–43.

16. Biometrics and Privacy Issues and Challenges // OVIC. URL: https://ovic.vic.gov.au/privacy/resources-for-organisations/biometrics-and-privacy-issues-and-challenges/ (date of access: 05.04.2026).

17. Facial Recognition Challenges // AIMultiple. URL: https://aimultiple.com/facial-recognition-challenges (date of access: 08.04.2026).

18. Facial Recognition and Privacy Concerns and Solutions in the Age of AI // ISACA. URL: https://www.isaca.org/resources/news-and-trends/isaca-now-blog/2025/facial-recognition-and-privacy-concerns-and-solutions-in-the-age-of-ai (date of access: 10.04.2026).

19. Comparative analysis of different biometric techniques for security systems / Niazy M. S. et al. *Aust. J. Eng. Innov. Technol*. 2023. Vol. 5, no. 3. P. 141–153.

20. Biometric Security: Face ID vs Fingerprint Locks Residential Comparison // Guardian Alarm. URL: https://guardianalarm.com/blog/biometric-security-face-id-vs-fingerprint-locks-residential-comparison/ (date of access: 14.04.2026).

21. A Comparison of Open-Source Home Automation Systems / Setz B. et al. *IEEE Access.* 2021. Vol. 9. P. 167332–167352.

22. Sharif K., Tenbergen B. Smart Home Voice Assistants: A Literature Survey of User Privacy and Security Vulnerabilities. *Complex Systems Informatics and Modeling Quarterly*. 2020. no. 24. P. 15–30.

23. Rogner W. Analysis of arguments against Open Source and Open Solutions: Separating truth from myth. University of Applied Sciences Technikum Wien, 2012. P. 1–8.

24. Decentralized Hybrid Multi-Face Recognition for IoT Applications / Abdullahu E. et al. *Sensors.* 2025. Vol. 25, no. 18. P. 1–17.

25. Скінченний автомат // Вікіпедія : вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Скінченний_автомат (дата звернення: 20.04.2026).

26. The Python Global Interpreter Lock (GIL) // Real Python. URL: https://realpython.com/python-gil/ (дата звернення: 21.04.2026).

27. Для чого використовують C++? // Кафедра АПЕПС ТЕФ КПІ. URL: https://apeps.kpi.ua/dlia-choho-vykorystovuiut-cpp (дата звернення: 21.04.2026).

28. Software Memory Safety: Cybersecurity Information Sheet. Ver. 1.1 / National Security Agency, CISA, MS-ISAC. 2023. URL: https://media.defense.gov/2022/Nov/10/2003112742/-1/-1/1/CSI_SOFTWARE_MEMORY_SAFETY.PDF (date of access: 22.04.2026).

29. Fundamentals of garbage collection // Microsoft Learn. URL: https://learn.microsoft.com/en-us/dotnet/standard/automatic-memory-management (date of access: 23.04.2026).

30. C# Parallel Programming and Performance // LinkedIn. URL: https://www.linkedin.com/posts/karen-corrêa_csharp-parallelprogramming-performance-activity-7312851255930884097-UiI9 (date of access: 23.04.2026).

31. Kiselev I. Comparative analysis of libraries for computer vision OpenCV and AForge.NET for use in gesture recognition system. *Journal of Physics: Conference Series*. 2020. Vol. 1661, no. 012048. P. 1–5.

32. A Comprehensive Review of IoT Networking Technologies for Smart Home Automation Applications / Orfanos V. A. et al. *Journal of Sensor and Actuator Networks*. 2023. Vol. 12, no. 2. P. 30.

33. Danbatta S. J., Varol A. Comparison of Zigbee, Z-Wave, Wi-Fi, and Bluetooth Wireless Technologies Used in Home Automation. *7th International Symposium on Digital Forensics and Security (ISDFS)*: proceedings, 10-12 June 2019, Barcelos, Portugal, 2019. P. 1–5.

34. Gemirter C. B., Senturca C., Baydere S. A Comparative Evaluation of AMQP, MQTT and HTTP Protocols Using Real-Time Public Smart City Data // *6th International Conference on Computer Science and Engineering (UBMK)*: proceedings, 15-17 September 2021, Ankara, Turkey, 2021. P. 576–581.

35. Salama M., Raslen B. MQTT in Action: Building Reliable and Scalable Home Automation Systems. *Sakarya University Journal of Computer and Information Sciences*. 2024. Vol. 7, no. 3. P. 494–509.

36. Evaluating the Performance of Eigenface, Fisherface, and Local Binary Pattern Histogram-Based Facial Recognition Methods under Various Weather Conditions / Ahsan M. M. et al. *Technologies.* 2021. Vol. 9, no. 2. P. 1-12.

37. Comparison of JSON and XML Data Formats in Document Stored NoSql Database Replication Processes / Rianto R. et al. *Int. J. Adv. Sci. Eng. Inf. Technol.* 2021. Vol. 11, no. 3. P. 1150–1156.

38. JSONPath Transformation // OpenHAB Documentation. URL: https://www.openhab.org/addons/transformations/jsonpath/ (date of access: 26.04.2026).

39. Blinking // Wikipedia. URL: https://en.wikipedia.org/wiki/Blinking (date of access: 28.04.2026).

40. False Rejection Rate // RecFaces. URL: https://recfaces.com/articles/false-rejection-rate (date of access: 30.04.2026).

ДОДАТОК А
Текст програми

А.1 Лістинг файлу точки входу Program.cs

```
using System;
using System.Windows.Forms;
using System.Threading.Tasks;

namespace SmartHomeFaceID
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

А.2 Лістинг файлу головного вікна MainForm.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using SmartHomeFaceID.Algorithms;
using SmartHomeFaceID.Services;
using System.IO;
using System.Threading.Tasks;
using System.Collections.Generic;

namespace SmartHomeFaceID
{
    public partial class MainForm : Form
    {
        private FaceDetectionService _faceService = null!;
        private RecognitionService _recognitionService = null!;
        private AntiSpoofingService _antiSpoofingService = null!;
        private MqttIntegrationService _mqttService = null!;
        private RegistrationService _registrationService = null!;
        private CameraService _cameraService = null!;

        private DateTime _lastMqttSent = DateTime.MinValue;
        private TimeSpan _mqttCooldown = TimeSpan.FromSeconds(10);

        public MainForm()
        {
```

```
            InitializeComponent();
            LoadServices();
        }

        private void LoadServices()
        {
            string baseDir = AppDomain.CurrentDomain.BaseDirectory;
            string prototxtPath = Path.Combine(baseDir, "Models", "FaceDetection",
"deploy.prototxt");
            string modelPath = Path.Combine(baseDir, "Models", "FaceDetection",
"res10_300x300_ssd_iter_140000.caffemodel");
            string eyeCascadePath = Path.Combine(baseDir, "Models", "AntiSpoofing",
"haarcascade_eye.xml");

            _faceService = new FaceDetectionService(prototxtPath, modelPath, 0.4f);
            _recognitionService = new RecognitionService();
            _antiSpoofingService = new AntiSpoofingService(eyeCascadePath);
            _mqttService = new MqttIntegrationService("broker.emqx.io");
            _registrationService = new RegistrationService(_faceService,
_recognitionService);
            _cameraService = new CameraService();

            _cameraService.OnFrameCaptured += CameraService_OnFrameCaptured;
        }

        private async void MainForm_Load(object sender, EventArgs e)
        {
            await _mqttService.ConnectAsync();
            _cameraService.StartCamera(0);
            lblStatus.Text = "Статус: Система готова";
        }

        private List<(string Text, Point Pos, Color Color, int Size)> _pendingLabels = new
List<(string, Point, Color, int)>();

        private void CameraService_OnFrameCaptured(object sender, Mat frame)
        {
            _pendingLabels.Clear();

            if (_registrationService.IsActive)
            {
                _registrationService.ProcessFrame(frame);
                _pendingLabels.Add((_registrationService.CurrentInstruction, new Point(30,
30), Color.Yellow, 18));
                _pendingLabels.Add((_registrationService.CurrentProgress, new Point(30,
70), Color.Yellow, 14));
                UpdateDisplay(frame);
                return;
            }

            var faces = _faceService.DetectFaces(frame);
            foreach (var face in faces)
```

```
                {
                    CvInvoke.Rectangle(frame, face, new MCvScalar(255, 0, 0), 2);

                    using Mat faceMat = new Mat(frame, face);
                    var (id, distance) = _recognitionService.Recognize(faceMat);
                    bool isAlive = _antiSpoofingService.CheckLiveness(faceMat);

                    string name = id == -1 ? "Невідомий" : $"Користувач {id}";
                    string label = $"{name} ({distance:F0})";

                    _pendingLabels.Add((label, new Point(face.X, face.Y - 30), Color.Lime, 14));

                    string livenessLabel = isAlive ? "Живий: OK" : "Живий: ...";
                    Color livenessColor = isAlive ? Color.Lime : Color.Red;
                    _pendingLabels.Add((livenessLabel, new Point(face.X, face.Y + face.Height
+ 5), livenessColor, 10));

                    if (id != -1 && isAlive && distance < 65)
                    {
                        if (DateTime.Now - _lastMqttSent > _mqttCooldown)
                        {
                            Task.Run(() => _mqttService.SendAccessGrantedAsync(id, distance));
                            _lastMqttSent = DateTime.Now;
                            this.Invoke((MethodInvoker)delegate {
                                lblStatus.Text = $"Статус: Доступ надано для ID {id}";
                            });
                        }
                    }
                }

                UpdateDisplay(frame);
            }

            private void UpdateDisplay(Mat frame)
            {
                if (picWebcam.IsDisposed) return;

                Bitmap bmp = frame.ToBitmap();

                using (Graphics g = Graphics.FromImage(bmp))
                {
                    g.SmoothingMode                                                    =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
                    foreach (var item in _pendingLabels)
                    {
                        using (Font font = new Font("Arial", item.Size, FontStyle.Bold))
                        using (Brush brush = new SolidBrush(item.Color))
                        {
                            g.DrawString(item.Text, font, brush, item.Pos.X, item.Pos.Y);
                        }
                    }
                }
```

```
                this.Invoke((MethodInvoker)delegate {
                    var old = picWebcam.Image;
                    picWebcam.Image = bmp;
                    old?.Dispose();
                });
            }

            private void btnRegister_Click(object sender, EventArgs e)
            {
                using (var inputForm = new Form())
                {
                    inputForm.Text = "Реєстрація нового користувача";
                    inputForm.Size = new Size(300, 150);
                    inputForm.StartPosition = FormStartPosition.CenterParent;

                    Label lbl = new Label() { Text = "Введіть ID:", Left = 20, Top = 20, Width =
100 };
                    TextBox txt = new TextBox() { Left = 120, Top = 18, Width = 100 };
                    Button btnOk = new Button() { Text = "Почати", Left = 100, Top = 60,
DialogResult = DialogResult.OK };

                    inputForm.Controls.Add(lbl);
                    inputForm.Controls.Add(txt);
                    inputForm.Controls.Add(btnOk);
                    inputForm.AcceptButton = btnOk;

                    if (inputForm.ShowDialog() == DialogResult.OK)
                    {
                        if (int.TryParse(txt.Text, out int userId))
                        {
                            _registrationService.StartRegistration(userId);
                            lblStatus.Text = "Статус: Триває реєстрація...";
                        }
                    }
                }
            }

            private void btnDelete_Click(object sender, EventArgs e)
            {
                if (MessageBox.Show("Ви впевнені, що хочете видалити базу даних?",
"Підтвердження", MessageBoxButtons.YesNo) == DialogResult.Yes)
                {
                    string dbPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
"Models", "Recognition", "trained_faces.yml");
                    if (File.Exists(dbPath)) File.Delete(dbPath);
                    MessageBox.Show("Базу даних видалено. Будь ласка, перезапустіть
додаток.");
                }
            }

            protected override void OnFormClosing(FormClosingEventArgs e)
```

```
        {
            _cameraService.StopCamera();
            _faceService.Dispose();
            _recognitionService.Dispose();
            _antiSpoofingService.Dispose();
            _mqttService.Dispose();
            _registrationService.Dispose();
            base.OnFormClosing(e);
        }
    }
}
```

А.3 Лістинг сервісу виявлення облич FaceDetectionService.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using Emgu.CV;
using Emgu.CV.Dnn;
using Emgu.CV.Structure;

namespace SmartHomeFaceID.Algorithms
{
    public class FaceDetectionService : IDisposable
    {
        private Net? _faceNet;
        private float _confidenceThreshold;

        public FaceDetectionService(string prototxtPath, string modelPath, float confidenceThreshold = 0.6f)
        {
            try
            {
                _faceNet = DnnInvoke.ReadNetFromCaffe(prototxtPath, modelPath);
                _confidenceThreshold = confidenceThreshold;
                Console.WriteLine("DNN модуль детекції обличчя успішно завантажено.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Помилка завантаження DNN моделі: {ex.Message}");
            }
        }

        public List<Rectangle> DetectFaces(Mat frame)
        {
            List<Rectangle> faces = new List<Rectangle>();

            if (frame == null || frame.IsEmpty || _faceNet == null)
                return faces;
```

```
                Mat blob = DnnInvoke.BlobFromImage(frame, 1.0, new Size(300, 300), new
MCvScalar(104.0, 177.0, 123.0), true, false);

                _faceNet.SetInput(blob);

                Mat detection = _faceNet.Forward();
                float[,,,] data = (float[,,,])detection.GetData();
                int detectionsCount = data.GetLength(2);

                for (int i = 0; i < detectionsCount; i++)
                {
                    float confidence = data[0, 0, i, 2];

                    if (confidence > _confidenceThreshold)
                    {
                        int startX = (int)(data[0, 0, i, 3] * frame.Width);
                        int startY = (int)(data[0, 0, i, 4] * frame.Height);
                        int endX = (int)(data[0, 0, i, 5] * frame.Width);
                        int endY = (int)(data[0, 0, i, 6] * frame.Height);

                        startX = Math.Max(0, startX);
                        startY = Math.Max(0, startY);
                        endX = Math.Min(frame.Width - 1, endX);
                        endY = Math.Min(frame.Height - 1, endY);

                        faces.Add(new Rectangle(startX, startY, endX - startX, endY - startY));
                    }
                }

                return faces;
            }

            public void Dispose()
            {
                _faceNet?.Dispose();
            }
        }
    }
```

A.4 Лістинг сервісу ідентифікації RecognitionService.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Face;
using Emgu.CV.Util;

namespace SmartHomeFaceID.Algorithms
```

```
{
    public class RecognitionService : IDisposable
    {
        private LBPHFaceRecognizer _faceRecognizer;
        private readonly string _modelPath;
        private bool _isTrained = false;

        public RecognitionService()
        {
            _modelPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Models", "Recognition", "trained_faces.yml");
            _faceRecognizer = new LBPHFaceRecognizer(1, 8, 8, 8, 100);
            LoadModel();
        }

        private void LoadModel()
        {
            if (File.Exists(_modelPath))
            {
                _faceRecognizer.Read(_modelPath);
                _isTrained = true;
                Console.WriteLine("[LBPH] Базу біометричних даних успішно завантажено.");
            }
            else
            {
                Console.WriteLine("[LBPH] База порожня. Система потребує додавання користувачів.");
            }
        }

        public void AddUser(int userId, List<Mat> faceImages)
        {
            if (faceImages == null || faceImages.Count == 0) return;

            VectorOfMat images = new VectorOfMat();
            VectorOfInt labels = new VectorOfInt();

            foreach (var face in faceImages)
            {
                Mat grayFace = new Mat();
                if (face.NumberOfChannels == 3)
                    CvInvoke.CvtColor(face, grayFace, ColorConversion.Bgr2Gray);
                else
                    face.CopyTo(grayFace);

                Mat resizedFace = new Mat();
                CvInvoke.Resize(grayFace, resizedFace, new System.Drawing.Size(100, 100));

                images.Push(resizedFace);
                labels.Push(new[] { userId });
```

```
            }

            if (_isTrained)
            {
                _faceRecognizer.Update(images, labels);
            }
            else
            {
                _faceRecognizer.Train(images, labels);
                _isTrained = true;
            }
            string? directory = Path.GetDirectoryName(_modelPath);
            if (!string.IsNullOrEmpty(directory))
            {
                Directory.CreateDirectory(directory);
            }
            _faceRecognizer.Write(_modelPath);

            Console.WriteLine($"[LBPH] Користувача з ID {userId} успішно додано.
Файл оновлено.");
        }
        public (int Id, double Distance) Recognize(Mat faceImage)
        {
            if (!_isTrained) return (-1, 0);

            Mat grayFace = new Mat();
            if (faceImage.NumberOfChannels == 3)
                CvInvoke.CvtColor(faceImage, grayFace, ColorConversion.Bgr2Gray);
            else
                faceImage.CopyTo(grayFace);

            Mat resizedFace = new Mat();
            CvInvoke.Resize(grayFace, resizedFace, new System.Drawing.Size(100, 100));

            FaceRecognizer.PredictionResult result = _faceRecognizer.Predict(resizedFace);

            return (result.Label, result.Distance);
        }

        public void Dispose()
        {
            _faceRecognizer?.Dispose();
        }
    }
}
```

А.5 Лістинг модуля захисту від спуфінгу AntiSpoofingService.cs

```
using System;
using System.Drawing;
using Emgu.CV;
```

```
using Emgu.CV.Structure;

namespace SmartHomeFaceID.Algorithms
{
  public class AntiSpoofingService : IDisposable
  {
    private CascadeClassifier? _eyeCascade;

    private bool _eyesWereClosed = false;
    private DateTime _timeEyesClosed;

    private readonly TimeSpan _maxBlinkDuration = TimeSpan.FromMilliseconds(500);

    private DateTime _verifiedUntil = DateTime.MinValue;
    private readonly TimeSpan _verificationHoldTime = TimeSpan.FromSeconds(10);

    public AntiSpoofingService(string eyeCascadePath)
    {
      try
      {
        _eyeCascade = new CascadeClassifier(eyeCascadePath);
        Console.WriteLine("[Liveness] Модуль захисту від спуфінгу (Моргання) завантажено.");
      }
      catch (Exception ex)
      {
        Console.WriteLine($"[Помилка] Не вдалося завантажити каскад очей: {ex.Message}");
      }
    }

    public bool CheckLiveness(Mat faceImage)
    {
      if (faceImage == null || faceImage.IsEmpty || _eyeCascade == null)
        return false;

      Mat grayFace = new Mat();
      if (faceImage.NumberOfChannels == 3)
        CvInvoke.CvtColor(faceImage, grayFace, Emgu.CV.CvEnum.ColorConversion.Bgr2Gray);
      else
        faceImage.CopyTo(grayFace);

      int searchHeight = (int)(grayFace.Height * 0.6);
      Rectangle eyeRegion = new Rectangle(0, 0, grayFace.Width, searchHeight);
      using Mat upperFace = new Mat(grayFace, eyeRegion);

      Rectangle[] eyes = _eyeCascade.DetectMultiScale(upperFace, 1.1, 5, new Size(15, 15));

      if (eyes.Length == 0)
```

```
                {
                    if (!_eyesWereClosed)
                    {
                        _eyesWereClosed = true;
                        _timeEyesClosed = DateTime.Now;
                    }
                    return false;
                }
                else
                {
                    if (_eyesWereClosed)
                    {
                        TimeSpan closedDuration = DateTime.Now - _timeEyesClosed;
                        _eyesWereClosed = false;

                        if (closedDuration.TotalMilliseconds > 50 && closedDuration <= _maxBlinkDuration)
                        {
                            Console.WriteLine($"[Liveness] Зафіксовано моргання! (Тривалість: {closedDuration.TotalMilliseconds:F0} мс)");
                            _verifiedUntil = DateTime.Now + _verificationHoldTime;
                        }
                    }
                }

                return DateTime.Now < _verifiedUntil;
            }

            public void Dispose()
            {
                _eyeCascade?.Dispose();
            }
        }
    }
```

А.6 Лістинг сервісу роботи з камерою CameraService.cs

```
using System;
using Emgu.CV;

namespace SmartHomeFaceID.Services
{
    public class CameraService : IDisposable
    {
        private VideoCapture? _capture;
        private bool _isRunning;

        public event EventHandler<Mat>? OnFrameCaptured;

        public void StartCamera(int cameraIndex = 0)
        {
```

```
            if (_capture == null)
            {
                _capture = new VideoCapture(cameraIndex, VideoCapture.API.Any);
            }

            if (!_capture.IsOpened)
            {
                throw new Exception("Помилка: Не вдалося підключитися до камери. Перевірте підключення.");
            }

            _isRunning = true;

            _capture.ImageGrabbed += ProcessFrame;

            _capture.Start();
            Console.WriteLine($"Камеру {cameraIndex} успішно запущено.");
        }

        private void ProcessFrame(object? sender, EventArgs e)
        {
            if (!_isRunning) return;

            Mat frame = new Mat();
            _capture?.Retrieve(frame);

            if (!frame.IsEmpty)
            {
                OnFrameCaptured?.Invoke(this, frame);
            }
        }

        public void StopCamera()
        {
            _isRunning = false;
            if (_capture != null)
            {
                _capture.ImageGrabbed -= ProcessFrame;
                _capture.Pause();
                _capture.Stop();
                _capture.Dispose();
                _capture = null;
            }
            Console.WriteLine("Камеру зупинено та ресурси звільнено.");
        }

        public void Dispose()
        {
            StopCamera();
        }
    }
}
```

А.7 Лістинг сервісу реєстрації RegistrationService.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Threading;
using Emgu.CV;
using Emgu.CV.CvEnum;
using SmartHomeFaceID.Algorithms;

namespace SmartHomeFaceID.Services
{
    public enum RegistrationStep
    {
        None,
        LookStraight,
        TurnLeft,
        TurnRight,
        LookUp,
        LookDown,
        Complete
    }

    public class RegistrationService : IDisposable
    {
        private readonly RecognitionService _recognitionService;
        private readonly FaceDetectionService _faceService;

        public bool IsActive { get; private set; } = false;
        public RegistrationStep CurrentStep { get; private set; } = RegistrationStep.None;
        public int CurrentUserId { get; private set; }
        public string CurrentInstruction { get; private set; } = "";
        public string CurrentProgress { get; private set; } = "";

        private List<Mat> _capturedFaces = new List<Mat>();
        private const int ImagesPerStep = 15;
        private int _capturedInCurrentStep = 0;

        public RegistrationService(FaceDetectionService faceService, RecognitionService recognitionService)
        {
            _faceService = faceService;
            _recognitionService = recognitionService;
        }

        public void Dispose()
        {
            foreach (var face in _capturedFaces) face.Dispose();
            _capturedFaces.Clear();
        }

        public void StartRegistration(int userId)
```

```
        {
            CurrentUserId = userId;
            _capturedFaces.Clear();
            _capturedInCurrentStep = 0;
            CurrentStep = RegistrationStep.LookStraight;
            IsActive = true;
            Console.WriteLine($"[Registration] Почато реєстрацію для користувача {userId}");
        }

        public void ProcessFrame(Mat frame)
        {
            if (!IsActive) return;

            var faces = _faceService.DetectFaces(frame);

            CurrentInstruction = GetInstructionText();
            CurrentProgress = $"Прогрес: {_capturedInCurrentStep}/{ImagesPerStep}";

            if (faces.Count > 0)
            {
                var faceRect = faces[0];
                CvInvoke.Rectangle(frame, faceRect, new Emgu.CV.Structure.MCvScalar(0, 255, 255), 2);

                if (DateTime.Now.Millisecond % 3 == 0)
                {
                    Mat faceMat = new Mat(frame, faceRect);
                    _capturedFaces.Add(faceMat.Clone());
                    _capturedInCurrentStep++;

                    if (_capturedInCurrentStep >= ImagesPerStep)
                    {
                        NextStep();
                    }
                }
            }
        }

        private void NextStep()
        {
            _capturedInCurrentStep = 0;
            switch (CurrentStep)
            {
                case RegistrationStep.LookStraight: CurrentStep = RegistrationStep.TurnLeft; break;
                case RegistrationStep.TurnLeft: CurrentStep = RegistrationStep.TurnRight; break;
                case RegistrationStep.TurnRight: CurrentStep = RegistrationStep.LookUp; break;
                case RegistrationStep.LookUp: CurrentStep = RegistrationStep.LookDown; break;
```

```
                case RegistrationStep.LookDown:
                    CompleteRegistration();
                    break;
            }
        }

        private void CompleteRegistration()
        {
            _recognitionService.AddUser(CurrentUserId, _capturedFaces);
            IsActive = false;
            CurrentStep = RegistrationStep.None;
            Console.WriteLine($"[Registration] Навчання завершено! Зібрано {_capturedFaces.Count} зразків.");

            foreach (var face in _capturedFaces) face.Dispose();
            _capturedFaces.Clear();
        }

        private string GetInstructionText()
        {
            return CurrentStep switch
            {
                RegistrationStep.LookStraight => "Дивіться ПРЯМО в камеру",
                RegistrationStep.TurnLeft => "Повільно поверніть голову ВЛІВО",
                RegistrationStep.TurnRight => "Повільно поверніть голову ВПРАВО",
                RegistrationStep.LookUp => "Повільно подивіться ВГОРУ",
                RegistrationStep.LookDown => "Повільно подивіться ВНИЗ",
                _ => ""
            };
        }
    }
}
```

А.8 Лістинг сервісу мережевої взаємодії MqttIntegrationService.cs

```
using System;
using System.Text.Json;
using System.Threading.Tasks;
using MQTTnet;
using MQTTnet.Client;

namespace SmartHomeFaceID.Services
{
    public class MqttIntegrationService : IDisposable
    {
        private IMqttClient _mqttClient;
        private MqttClientOptions _mqttOptions;
        private readonly string _brokerIp;

        public MqttIntegrationService(string brokerIp = "127.0.0.1", int port = 1883)
        {
            _brokerIp = brokerIp;
```

```
            var factory = new MqttFactory();
            _mqttClient = factory.CreateMqttClient();

            _mqttOptions = new MqttClientOptionsBuilder()
                .WithTcpServer(brokerIp, port)
                .WithClientId("FaceIdModule_" + Guid.NewGuid().ToString())
                .Build();
        }

        public async Task ConnectAsync()
        {
            try
            {
                if (!_mqttClient.IsConnected)
                {
                    await _mqttClient.ConnectAsync(_mqttOptions);
                    Console.WriteLine($"[MQTT] Успішно підключено до брокера
{_brokerIp}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"[MQTT Помилка] Не вдалося підключитися:
{ex.Message}");
            }
        }

        public async Task SendAccessGrantedAsync(int userId, double confidence)
        {
            if (!_mqttClient.IsConnected)
            {
                await ConnectAsync();
            }

            var payload = new
            {
                device = "DoorCamera",
                status = "access_granted",
                user_id = userId,
                confidence = Math.Round(confidence, 2),
                is_live = true,
                timestamp = DateTime.UtcNow.ToString("O")
            };

            string jsonPayload = JsonSerializer.Serialize(payload);

            var message = new MqttApplicationMessageBuilder()
                .WithTopic("smarthome/security/faceid")
                .WithPayload(jsonPayload)

.WithQualityOfServiceLevel(MQTTnet.Protocol.MqttQualityOfServiceLevel.AtLeastOnce)
                .Build();
```

```
            if (_mqttClient.IsConnected)
            {
                await _mqttClient.PublishAsync(message);
                Console.WriteLine($"[MQTT] Відправлено команду: {jsonPayload}");
            }
        }

        public async Task DisconnectAsync()
        {
            if (_mqttClient != null && _mqttClient.IsConnected)
            {
                await _mqttClient.DisconnectAsync();
                Console.WriteLine("[MQTT] Відключено від брокера.");
            }
        }

        public void Dispose()
        {
            _mqttClient?.Dispose();
        }
    }
}
```

ДОДАТОК Б

Слайди презентації

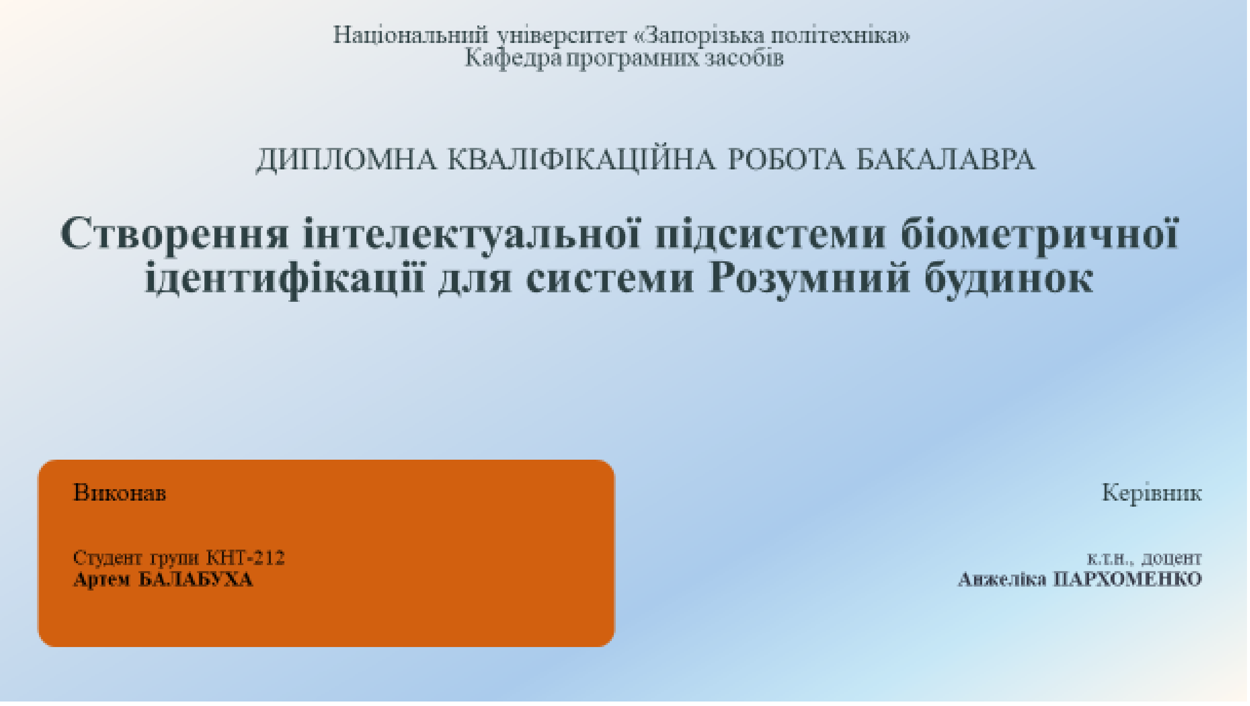

Рисунок Б.1 – Слайд 1

Рисунок Б.2 – Слайд 2

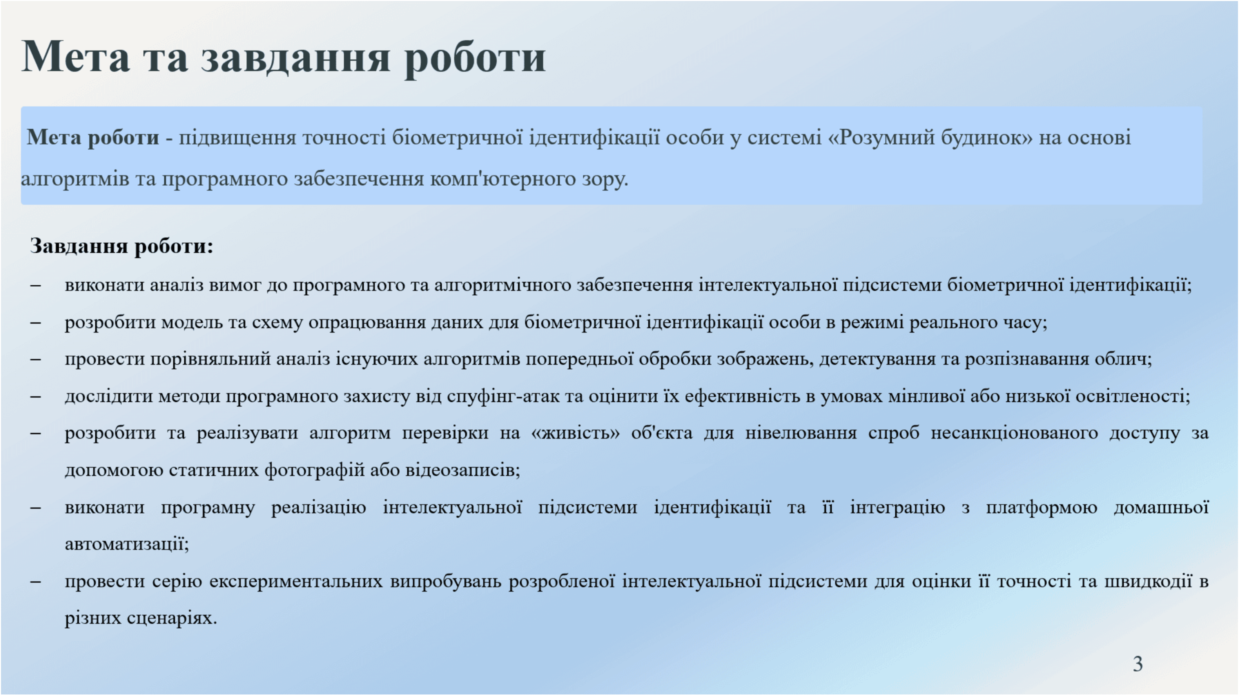

Рисунок Б.3 – Слайд 3

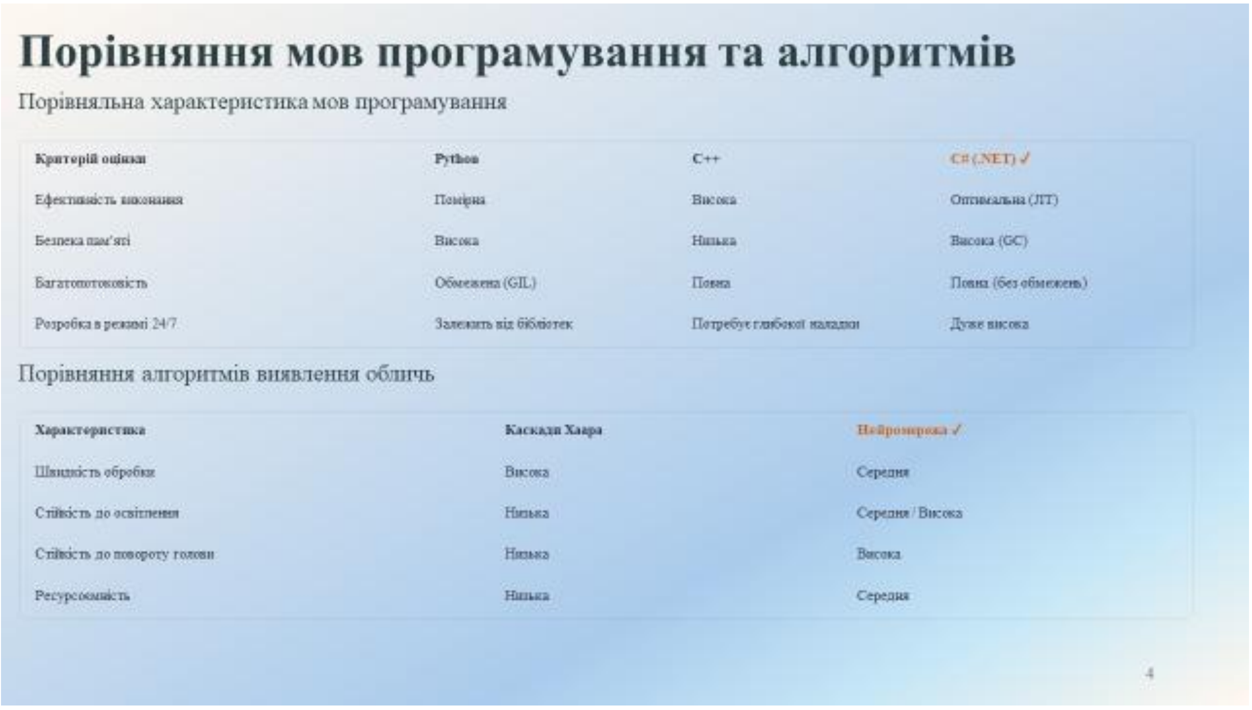

Рисунок Б.4 – Слайд 4

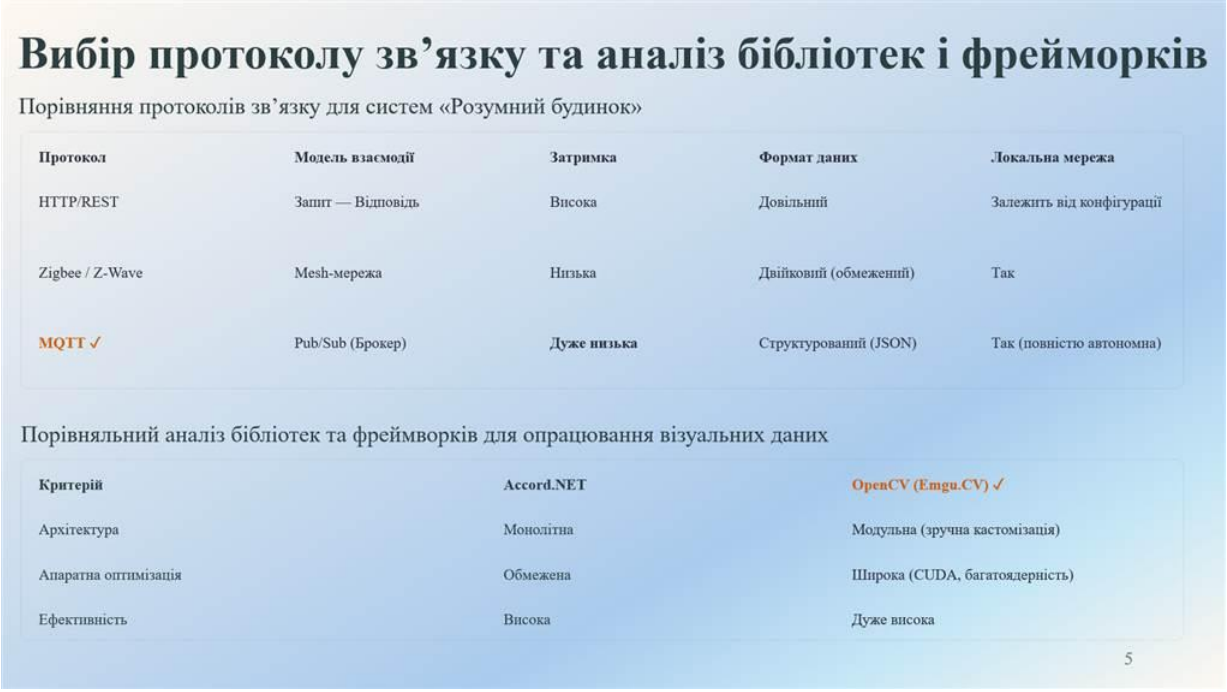

Рисунок Б.5 – Слайд 5

Рисунок Б.6 – Слайд 6

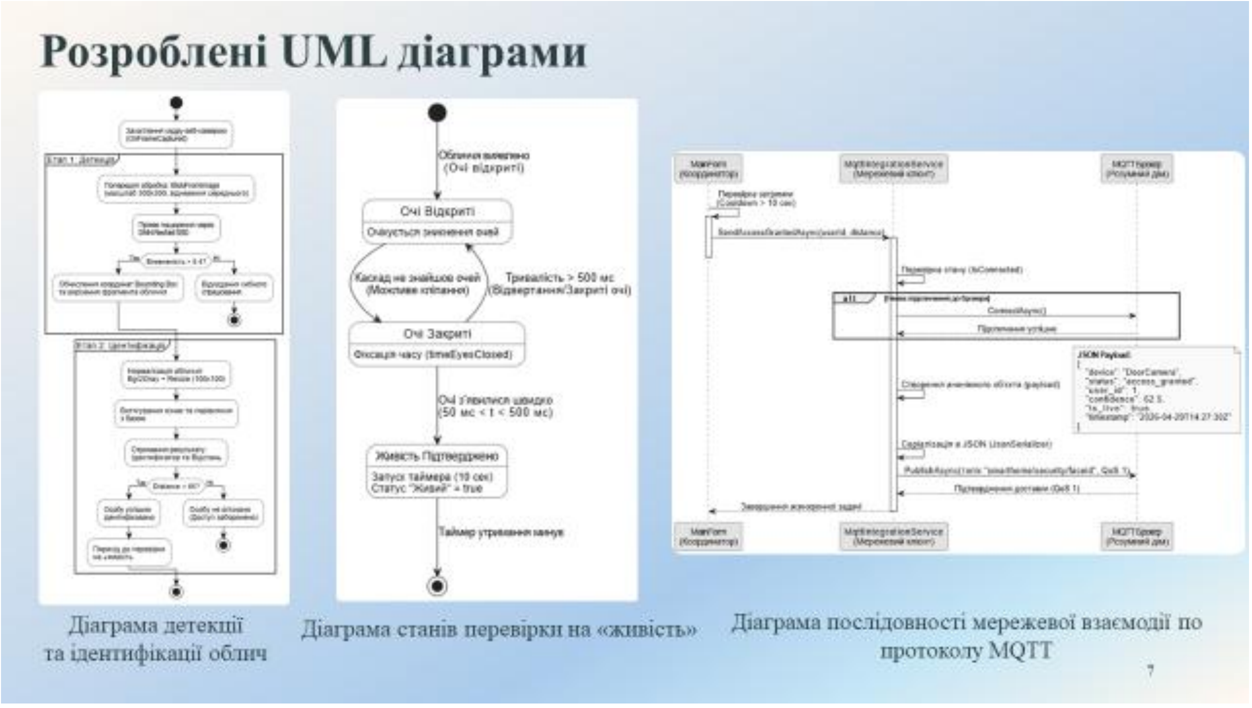

Рисунок Б.7 – Слайд 7

Рисунок Б.8 – Слайд 8

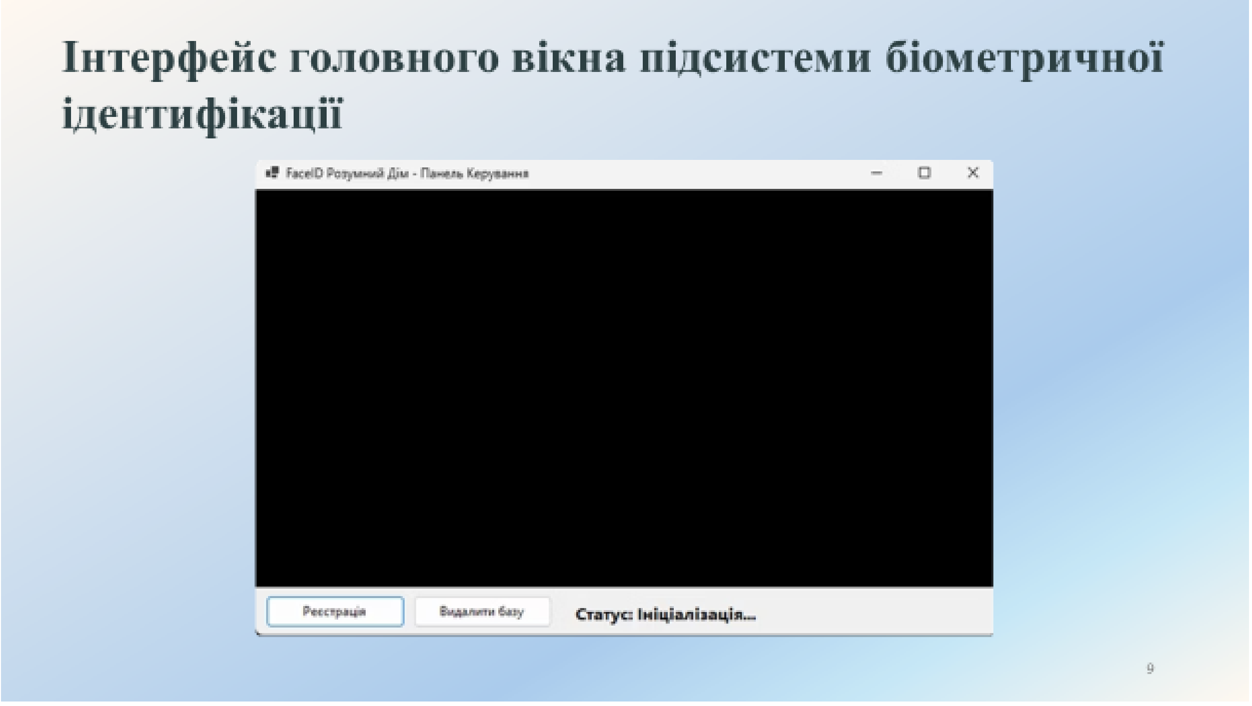

Рисунок Б.9 – Слайд 9

Рисунок Б.10 – Слайд 10

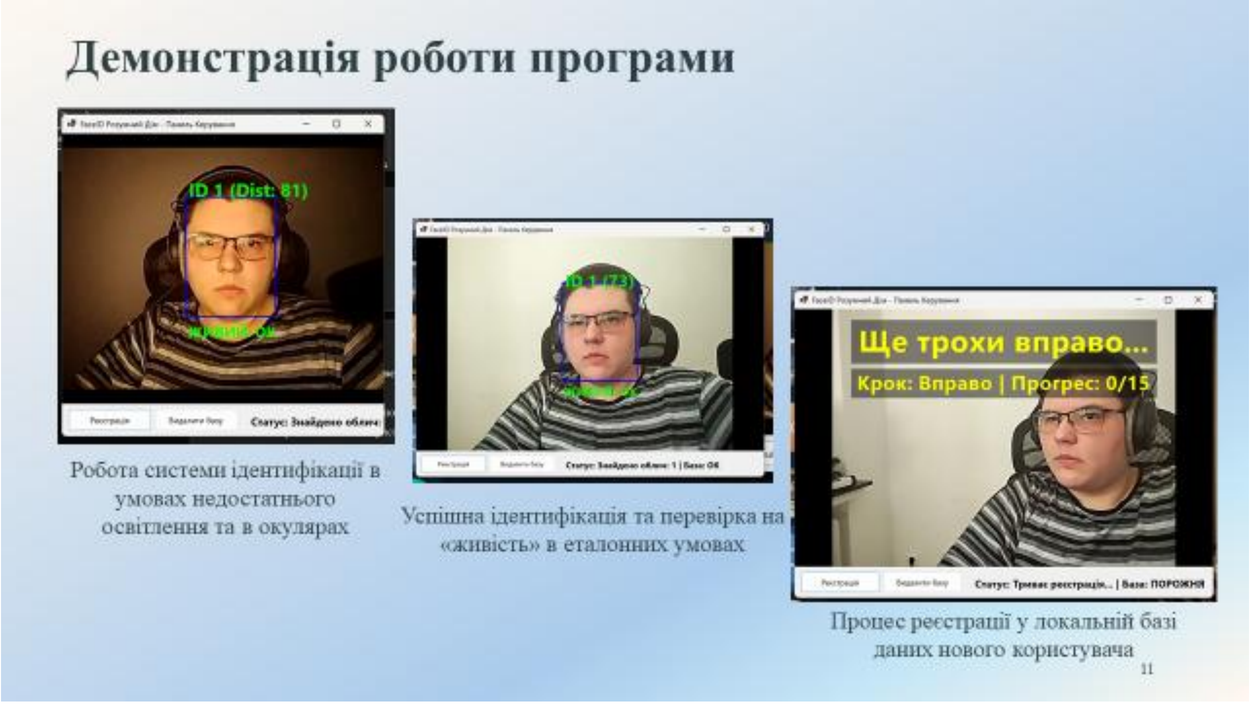

Рисунок Б.11 – Слайд 11

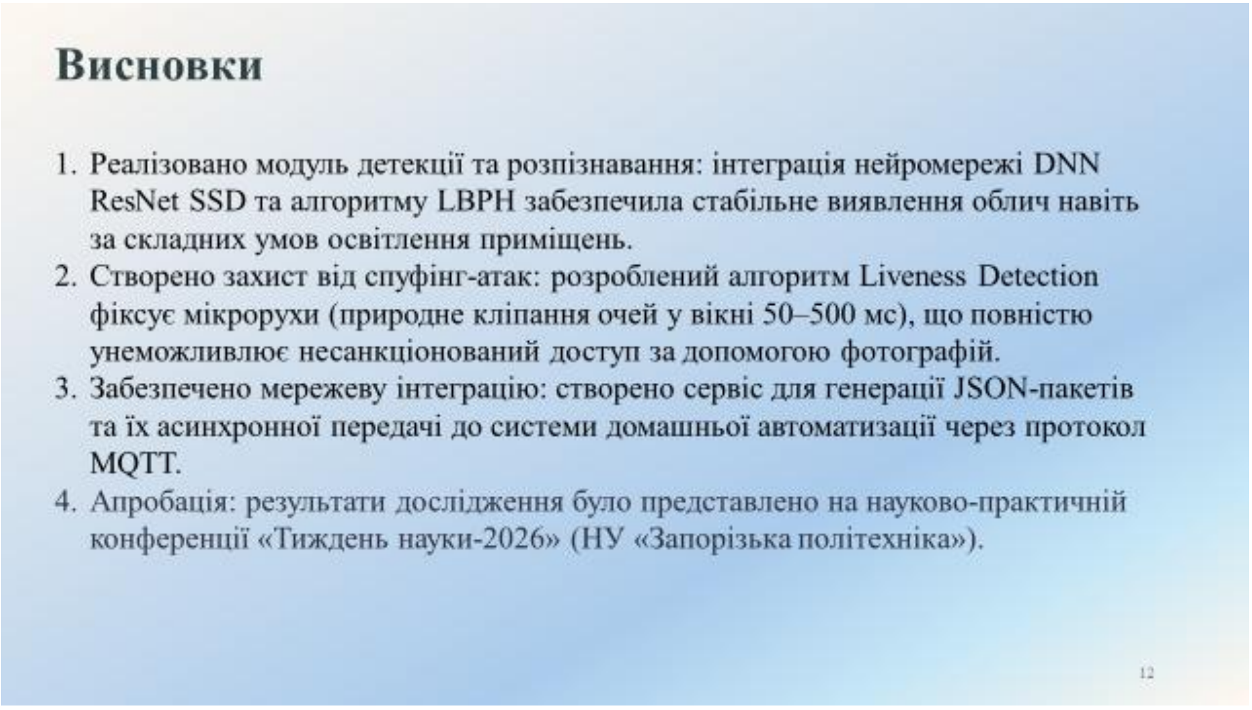

Рисунок Б.12 – Слайд 12