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 МОДЕЛІ ТА АЛГОРИТМИ ОПРАЦЮВАННЯ ДАНИХ ДЛЯ ІДЕНТИФІКАЦІЇ ТРАНСПОРТНИХ ЗАСОБІВ
DATA PROCESSING MODELS AND ALGORITHMS FOR VEHICLE IDENTIFICATION

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

ТЄРЄХОВ В.В
(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

ТЄРЄХОВА Владислава Володимировича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Моделі та алгоритми опрацювання даних для ідентифікації транспортних засобів. Data Processing Models and Algorithms for Vehicle Identification
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, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Програмна реалізація системи ідентифікації об'єктів автотранспорту. 4. Експлуатація, тестування та експериментальне дослідження програми.

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 _____ Владислав ТЄРЄХОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

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

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 105 с., 9 табл., 14 рис., 2 дод., 27 джерел.

ОПРАЦЮВАННЯ ВІДЕОДАНИХ, ІДЕНТИФІКАЦІЯ ОБ'ЄКТІВ, КОМП'ЮТЕРНИЙ ЗІР, АЛГОРИТМ ЛЕВЕНШТЕЙНА.

Об'єкт дослідження – процес опрацювання відеоданих для ідентифікації об'єктів у реальному часі.

Предмет дослідження – моделі та алгоритми комп'ютерного зору, методи сегментації символів і моделі нечіткого аналізу текстових даних для ідентифікації об'єктів у реальному часі.

Мета роботи – забезпечення точності та швидкодії ідентифікації об'єктів автотранспорту на основі розробки моделей і алгоритмів розпізнавання автомобільних номерних знаків у складних умовах.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування C#, бібліотека комп'ютерного зору OpenCV, технологія оптичного розпізнавання тексту Tesseract OCR, алгоритм нечіткого пошуку Левенштейна, платформа OpenHAB.

Результати. Створено програмний продукт, який у режимі реального часу опрацьовує дані відеопотоку, виконує його зчитування та підтвердження об'єктів автотранспорту за «білим списком». Реалізовано стійкість до нечітких номерних знаків на основі алгоритму Левенштейна, а також відправлення команд до платформи домашньої автоматизації OpenHAB.

Висновки. Розроблену програмну систему можна використовувати для ідентифікації об'єктів автотранспорту в складних умовах.

Галузь використання – автоматизація контрольно-пропускних пунктів житлових комплексів, приватних територій, промислових підприємств та логістичних вузлів.

ABSTRACT

Explanatory note to the bachelor's diploma qualification work: 105 pages, 9 tables, 14 figures, 2 appendices, 27 sources.

VIDEO DATA PROCESSING, OBJECT IDENTIFICATION, COMPUTER VISION, LEVENSHTEIN ALGORITHM.

The object of study is the process of video data processing for object identification in real time.

The subject of study is computer vision models and algorithms, character segmentation methods, and fuzzy text data analysis models for object identification in real time.

The purpose of the work is to ensure the accuracy and speed of vehicle identification based on the development of models and algorithms for license plate recognition in complex conditions.

Materials, methods, and tools: object-oriented programming, C# programming language, OpenCV computer vision library, Tesseract OCR technology, Levenshtein fuzzy search algorithm, OpenHAB platform.

Results. A software product has been created that processes video stream data in real time, reads it, and verifies vehicles against a "white list". Resistance to fuzzy license plates based on the Levenshtein algorithm has been implemented, as well as sending commands to the OpenHAB home automation platform.

Conclusions. The developed software system can be used for the identification of vehicles in complex conditions.

The field of use is the automation of checkpoints in residential complexes, private territories, industrial enterprises, and logistics hubs.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ALPR – Automatic License Plate Recognition;
ANPR – Automatic Number Plate Recognition;
CNN – Convolutional Neural Network;
FSM – Finite-State Machine;
GIL – Global Interpreter Lock;
GPU – Graphics Processing Unit;
HTTP – Hypertext Transfer Protocol;
JIT – Just-In-Time;
JSON – JavaScript Object Notation;
LSTM – Long Short-Term Memory;
MQTT – Message Queuing Telemetry Transport;
OCR – Optical Character Recognition;
QoS – Quality of Service;
ROI – Region of Interest;
TCP/IP – Transmission Control Protocol / Internet Protocol;
КПП – контрольно-пропускний пункт;
СКУД – система контролю та управління доступом;
ТЗ – транспортний засіб.

ВСТУП

Стрімкий розвиток систем «Розумного міста» та цифровізація інфраструктури безпеки все більше умовлюють необхідність автоматизації процесів контролю доступу транспортних засобів на території, що охороняються [1].

Традиційні методи контролю, такі як паперові перепустки, або ручне введення даних, характеризуються низькою пропускною здатністю та схильні до впливу людського чинника.

Впровадження автоматизованої системи контролю та управління доступом (СКУД) на основі алгоритмів розпізнавання номерних знаків дозволяє забезпечити безконтактну ідентифікацію об'єктів у режимі реального часу, тож актуальною задачею для сучасних логістичних центрів, паркінгів, приватних підприємств та будинків, що охороняються [1].

Сьогодні більшість рішень у сфері комп’ютерного зору для розпізнавання номерних знаків базуються на глибоких нейронних мережах та Optical Character Recognition (OCR). Проте наявні системи часто демонструють зниження точності у неконтрольовані моменти. Основними перешкодами є складні погодні умови, зокрема дощ, сніг, недостатнє освітлення в сутінках та поява візуальних перешкод у вигляді сторонніх написів або реклами на кузові автомобіля. Це обумовлює потребу в удосконаленні алгоритмів попередньої обробки зображень та механізмів нечіткого порівняння розпізнаних даних з базою за «білим» списком [1]-[2]. Тож система має забезпечувати високу точність ідентифікації об’єктів автотранспорту в складних погодних умовах та автоматичне управління виконавчими пристроями контролю доступу без необхідності використання дорогих фізичних ідентифікаторів.

Метою роботи є забезпечення точності та швидкодії ідентифікації об'єктів автотранспорту на основі розробки моделей і алгоритмів розпізнавання автомобільних номерних знаків у складних умовах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У першому розділі проводиться системний аналіз сучасного стану розвитку СКУД, а також методів ідентифікації транспортних засобів.

В умовах стрімкої цифровізації міської інфраструктури та розбудови концепцій «Розумного міста», актуальним постає питання переходу від традиційних методів контролю до інтелектуальних систем на основі комп’ютерного зору.

Метою даного розділу є дослідження існуючих технологічних рішень, класифікація методів ідентифікації та виявлення ключових чинників, що дестабілізують процес розпізнавання даних у реальних умовах експлуатації.

1.1 Визначення та роль СКУД у сучасних системах безпеки

Система контролю та управління доступом – це комплекс апаратних і програмних рішень, призначених для захисту об’єктів шляхом обмеження та реєстрації доступу.

Такі системи застосовуються на об’єктах різного призначення: від офісних приміщень і медичних закладів до приватних будинків та паркінгів. Основним завданням СКУД є контроль переміщення персоналу та транспортних засобів на територіях, що охороняються, для забезпечення загального рівня безпеки [3].

Функціонування СКУД базується на поєднанні різноманітних технічних засобів. До них належать традиційні карти доступу, електронні замки, кодові панелі, а також сучасні біометричні технології, зокрема сканери відбитків пальців або сітківки ока.

Кожен із цих пристроїв інтегрований у єдину систему під керуванням спеціалізованого програмного забезпечення, що здійснює фіксацію подій та управління доступом у режимі реального часу [3].

Системи даного типу здатні виконувати широкий спектр завдань: окрім безпосереднього обмеження входу, вони дозволяють вести облік робочого часу, інтегруватися із засобами відеонагляду та охоронної сигналізації, а також адаптуватися під специфічні потреби конкретного об’єкта [3].

Збільшення території об’єкта спостереження забезпечує саме гнучкість програмних платформ, що перетворює СКУД на єдиний інструмент управління інфраструктурою, що централізовано об’єднує розрізнені системи [3].

Однак у контексті автоматизації доступу автотранспорту використання традиційних методів ідентифікації (карток, кодів або радіобрелоків) створює низку незручностей: необхідність повної зупинки автомобіля, передачу ідентифікатора «з рук у руки», а також ризик його втрати або передачі стороннім особам.

Зазначені чинники суттєво знижують пропускну здатність контрольно-пропускних пунктів (КПП) та загальний рівень безпеки на об’єкті [4].

Саме тому сучасним етапом розвитку СКУД є перехід до автоматичної ідентифікації транспортних засобів за їхніми державними реєстраційними номерами.

Такий підхід дозволяє реалізувати безконтактний проїзд, забезпечує високу швидкість обслуговування транспортного потоку та формування автоматизованого цифрового журналу подій із фотофіксацією, що є критично важливим для сучасних систем безпеки [4].

Для забезпечення ефективної взаємодії всіх елементів, особливо в системах ідентифікації транспортних засобів, використовується багаторівнева структура. Узагальнену структурну схему автоматизованої СКУД на основі розпізнавання номерів наведено на рис. 1.1 [3]-[4].

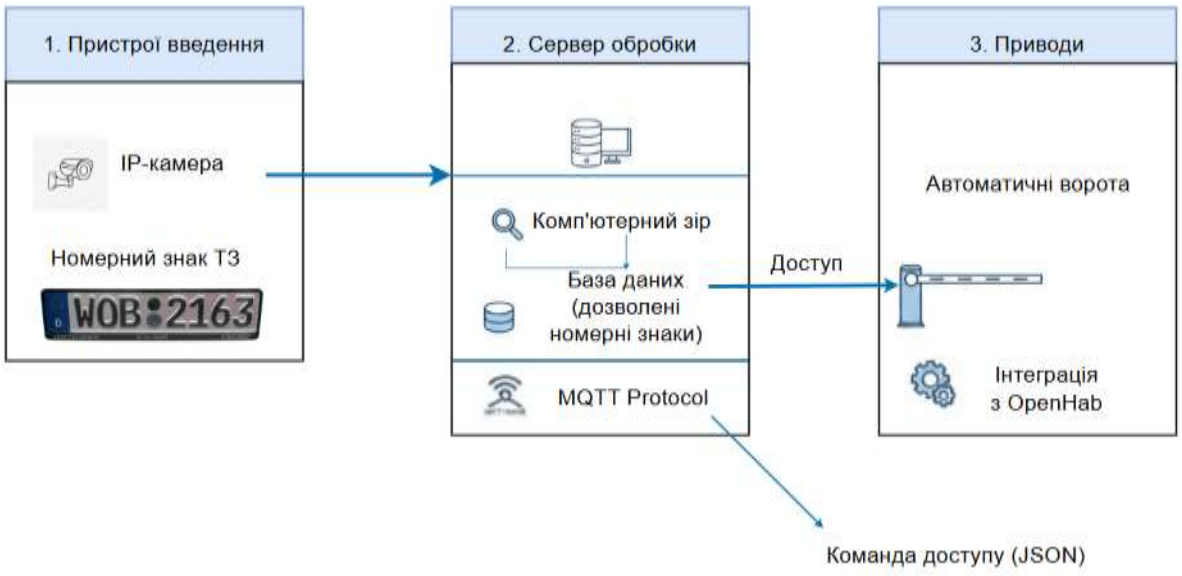

Рисунок 1.1 – Узагальнена структурна схема автоматизованої СКУД [3]-[4]

1.2 Класифікація систем та ідентифікації транспортних засобів

Функціонування СКУД для автотранспорту базується на послідовному виконанні етапів ідентифікації, аналізу та керування загороджувальними пристроями, такими як ворота або шлагбаум.

Залежно від архітектури, системи поділяються на:

- автономні системи: прості рішення для однієї точки доступу, що не потребують централізованого програмування та не формують складних звітів;
- мережеві системи: дозволяють віддалено керувати правами доступу, вести облік часу перебування автомобіля на території та інтегруватися з іншими підсистемами безпеки.

Окрім архітектурних особливостей, системи розрізняються за методом зчитування ідентифікаційних даних. Основні підходи до ідентифікації транспортних засобів та їх класифікація за фізичними принципами роботи наведені на рис. 1.2 [6]-[8].

Рисунок 1.2 – Класифікація методів ідентифікації транспортних засобів у СКУД [6]-[8]

Кожен із наведених на рисунку 1.2 методів ідентифікації транспортних засобів базується на різних фізичних принципах опрацювання вхідних даних, що безпосередньо визначає їхні технічні та експлуатаційні особливості.

Для вибору найбільш раціонального підходу в межах даної роботи необхідно провести порівняльне зіставлення існуючих технологій.

Такий аналіз дозволяє оцінити не лише функціональні можливості а й економічну доцільність впровадження системи на об'єктах із різною інтенсивністю руху.

Особлива увага приділяється здатності методів забезпечувати верифікацію без прямої участі водія та можливості інтеграції з автоматизованими журналами подій.

Детальне зіставлення переваг та недоліків розглянутих технологій, що стали підґрунтям для обрання методів комп’ютерного зору, наведено у табл. 1.1 [6]-[8].

Таблиця 1.1 – Порівняльна характеристика методів ідентифікації автотрансопорту у СКУД [6]-[8]

Метод ідентифікації	Принцип дії	Переваги	Недоліки
Радіочастотний (RFID)	Зчитування пасивної мітки на склі авто при наближенні до антени.	Висока швидкість проїзду, автоматизація без участі водія.	Потребує закупівлі та видачі фізичних міток кожному водію.
Магнітна петля	Детектування зміни магнітного поля при появі металу над петлею.	Працює за будь-якої погоди, висока надійність детектування.	Не ідентифікує конкретне авто, лише факт його наявності.
Автоматичне розпізнавання автономерів (ALPR)	Комп'ютерний зір та OCR-аналіз відеопотоку з камери.	Безконтактність, повний облік, відеофіксація водія та авто.	Залежність від чистоти номера та рівня освітлення.
Картки доступу	Зчитування коду з картки при наближенні до термінала.	Мінімальна вартість впровадження для невеликих об'єктів.	Необхідність зупинки авто, відкриття вікна; незручність у негоду; низька безпека.

Аналіз даних, наведених у таблиці 1.1, свідчить про те, що попри залежність від зовнішніх факторів, метод розпізнавання номерних знаків є найбільш раціональним рішенням для сучасних об’єктів. Ключовою перевагою даного підходу порівняно з використанням RFID-міток або радіопультів є можливість повної відеофіксації процесу проїзду.

Це дозволяє не лише автоматизувати контроль доступу, а й формувати доказову базу з візуальним підтвердженням того, хто саме і в який час перебував за кермом транспортного засобу. Такий підхід сприяє розв'язанню потенційних спірних ситуацій та суттєво підвищує загальний рівень безпеки й прозорості моніторингу на об’єкті.

Окрім переваг у безпеці, такий метод є найбільш економічно доцільним при масштабуванні системи, оскільки він усуває необхідність

закупівлі, програмування та видачі фізичних ідентифікаторів кожному новому користувачу. Це дозволяє реалізувати гнучкі алгоритми керування доступом через програмне забезпечення, забезпечуючи високу пропускну здатність об’єкта та мінімізуючи витрати на технічне обслуговування обладнання.

1.3. Аналіз факторів, що впливають на точність системи

Надійність функціонування системи ідентифікації за номерними знаками значною мірою залежить від здатності алгоритмів мінімізувати вплив похибок, що виникають ще на етапі первинного формування зображення.

Ефективність ALPR безпосередньо визначається якістю вхідних даних та стійкістю програмних методів опрацювання інформації до впливу дестабілізуючих зовнішніх чинників.

При розробці таких систем важливо врахувати всі можливі перешкоди, які можуть призвести до невірного зчитування інформації [9].

Основні технічні виклики та фактори впливу наведено у табл. 1.2 [8]-[11].

Таблиця 1.2 – Аналіз факторів впливу на точність ідентифікації ТЗ [8]-[11]

Категорія фактора	Конкретний чинник впливу	Наслідки для процесу розпізнавання
1	2	3
Геометричні	Порушення кутів огляду (нахил > 5°, горизонт > 30°).	Перспективні спотворення символів, що ускладнює роботу алгоритмів сегментації.
Технічні	Недостатня роздільна здатність (менше 150-250 пікселів на символ).	Втрата деталізації, неможливість впевненої ідентифікації символів OCR-двигуном.

Кінець таблиці 1.2

1	2	3
Метеорологічні	Атмосферні опади (дощ, туман, сніг).	Поява візуального шуму, розмиття контурів та часткова оклюзія (перекриття) номера .
Експлуатаційні	Забруднення номерної пластини, сторонні написи на кузові .	Помилкове визначення області інтересу (ROI) та пропуски знаків при зчитуванні.
Безпекові	Спуфінг-атаки (використання фото чи цифрових дисплеїв).	Ризик несанкціонованого доступу та помилкового спрацювання системи.

Проведений аналіз дозволяє зробити висновок, що надійність ідентифікації ТЗ у СКУД визначається сукупністю факторів, які можна розділити на керовані та некеровані.

– керовані фактори (геометричні та технічні) можуть бути мінімізовані ще на етапі проектування та монтажу системи шляхом дотримання регламентованих кутів огляду та використання камер із відповідною роздільною здатністю;

– некеровані фактори (метеорологічні умови та експлуатаційні забруднення) становлять найбільшу загрозу для стабільності системи, оскільки вони вносять випадкові візуальні викривлення, які неможливо усунути фізично.

Саме тому для забезпечення безперебійної роботи СКУД у реальних умовах основний акцент під час розробки має бути зміщений на програмні методи обробки даних.

1.4. Постановка завдань дипломної роботи

На основі проведеного аналізу предметної області виявлено фактори, що впливають на точність ідентифікації об’єктів автотранспорту (метеорологічні умови, забруднення номерних знаків, візуальний шум), що обумовлює актуальність цієї роботи.

Система має забезпечувати високу точність ідентифікації об’єктів автотранспорту в складних погодних умовах та автоматичне управління виконавчими пристроями контролю доступу без необхідності використання дорогих фізичних ідентифікаторів.

Метою роботи є забезпечення точності та швидкодії ідентифікації об'єктів автотранспорту на основі розробки моделей і алгоритмів розпізнавання автомобільних номерних знаків у складних умовах.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– виконати аналіз вимог до програмного та алгоритмічного забезпечення системи ідентифікації об’єктів автотранспорту;

– розробити модель та схему опрацювання даних для ідентифікації об’єктів автотранспорту в режимі реального часі;

– провести порівняльний аналіз існуючих алгоритмів попередньої обробки зображень та детектування об'єктів;

– дослідити методи OCR та оцінити їх ефективність при роботі з українськими номерними знаками в умовах низької освітленості;

– розробити та реалізувати алгоритм нечіткого порівняння для нівелювання помилок зчитування, спричинених частковим забрудненням або пошкодженням номерних знаків;

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

– провести серію експериментальних випробувань розробленої програмної системи для оцінки її точності та швидкодії в різних експлуатаційних сценаріях.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі на основі результатів аналізу предметної області та технічних викликів, розглянутих раніше, здійснюється перехід до етапу проєктування логічної структури системи.

Метою даного етапу є формалізація вимог до програмного комплексу та обґрунтування вибору моделей опрацювання даних та алгоритмів, які забезпечать високу точність ідентифікації транспортних засобів в умовах інтенсивних метеорологічних опадів та недостатнього освітлення .

Проєктування системи базується на принципах модульності та обчислювальної ефективності, що дозволяє реалізувати складні алгоритми комп’ютерного зору на локальних пристроях без втрати швидкодії. У межах розділу буде проведено детальний аналіз функціональних можливостей системи, розроблено архітектуру взаємодії компонентів та обґрунтовано вибір технологічного стеку, що відповідає сучасним стандартам розробки інтелектуальних систем опрацювання візуальних даних

2.1 Аналіз та формалізація вимог до програмного забезпечення

Процес проєктування системи ідентифікації транспортних засобів має розпочинатися з детального аналізу та формалізації вимог, що висуваються до програмного забезпечення. З огляду на специфіку роботи в умовах складних метеорологічних факторів та візуальних завад, вимоги до системи будуть поділятися на функціональні, нефункціональні та специфічні вимоги до алгоритмічного забезпечення опрацювання даних.

2.1.1 Функціональні вимоги

Функціональні вимоги визначають перелік завдань, які повинна вирішувати система для досягнення поставленої мети:

– отримання та попередня обробка відеопотоку: забезпечення стабільного захоплення кадрів із камери високої роздільної здатності з можливістю програмного коригування експозиції та балансу білого в реальному часі;

– інтелектуальна детекція об’єктів: автоматичне виявлення транспортного засобу в зоні контролю та динамічне визначення області інтересу, що містить державний номерний знак;

– сегментація та OCR: зчитування символів номерного знака з урахуванням можливих геометричних спотворень, спричинених ракурсом зйомки;

– ідентифікація та прийняття рішень: порівняння розпізнаного номера з існуючою базою даних за допомогою алгоритмів нечіткого порівняння для нівелювання помилок розпізнавання;

– управління виконавчою периферією: формування та відправка сигналів управління на контролери воріт або шлагбаумів через мережевий протокол;

– логування та моніторинг: ведення цифрового журналу подій, що включає текстовий номер, фотофіксацію автомобіля та часову мітку проїзду.

2.1.2 Нефункціональні вимоги

Ці вимоги визначають якісні характеристики системи та обмеження, в яких вона повинна функціонувати:

– продуктивність та латентність: загальний час від моменту появи авто в кадрі до формування сигналу на відкриття не повинен перевищувати 1.2 - 1.5 секунди. Це критично для запобігання утворенню черг;

– надійність та відмовостійкість: система має стабільно функціонувати в режимі 24/7. У разі втрати зв'язку з центральним сервером, локальний вузол обробки повинен зберігати події в черзі для подальшої синхронізації;

– стійкість до візуальних перешкод: алгоритми мають забезпечувати коректну ідентифікацію при рівні зашумлення зображення (від дощу або снігу) до 30% та при частковому забрудненні номерної пластини;

– безпека даних: забезпечення конфіденційності персональних даних власників ТЗ та захист від несанкціонованого доступу до бази даних через шифрування каналів передачі;

– масштабованість: архітектура програмного забезпечення повинна дозволяти підключення додаткових камер без повної перебудови логіки управління системою.

2.1.3 Вимоги до алгоритмічного забезпечення

Окремим класом вимог є вимоги до математичного та алгоритмічного забезпечення, а саме:

– адаптивність фільтрації: алгоритми попередньої обробки мають автоматично підлаштовуватися під рівень освітлення (день/ніч) та тип атмосферних опадів;

– висока точність сегментації: здатність системи розділяти символи навіть у разі їх часткового злиття на зображенні через низьку якість освітлення;

– інтелектуальна корекція помилок: використання метрик таких як відстань Левенштейна для відновлення частково нерозпізнаних символів на основі порівняння з базою.

2.1.4 Діаграма прецедентів системи ідентифікації ТЗ

Для уточнення взаємодії між користувачами та програмним комплексом, а також деталізації меж системи, було розроблено діаграму прецедентів (рис. 2.1). Ця діаграма дозволяє формалізувати сценарії використання СКУД та визначити ролі основних акторів.

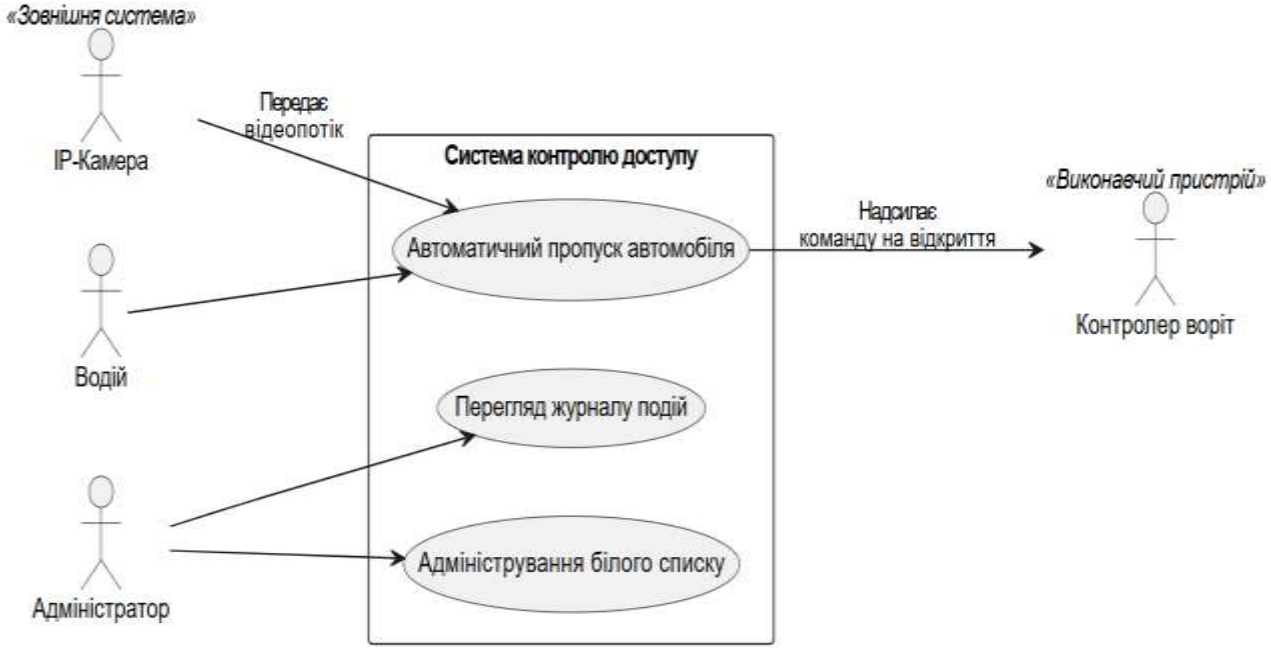

Рисунок 2.1 – Діаграма прецедентів системи ідентифікації ТЗ

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

– IP-камера забезпечує безперервну передачу відеопотоку для подальшого аналізу.Водій ініціює прецедент «Автоматичний пропуск автомобіля» через появу транспортного засобу в зоні детекції;

– програмний комплекс виконує ідентифікацію: сегментацію номера, розпізнавання символів та верифікацію даних;

– контролер воріт виступає виконавчим пристроєм, що отримує команду на відкриття після успішної перевірки.

Адміністратор здійснює налаштування «білого списку» та моніторинг журналу подій. Така структура дозволяє ізолювати бізнес-логіку СКУД від апаратного забезпечення, що спрощує модернізацію окремих модулів системи.

2.2 Розробка моделі та схеми опрацювання даних

Для формалізації логіки функціонування системи та забезпечення детермінованості процесів опрацювання даних у межах даної роботи використано математичний апарат скінченних автоматів.

Скінченний автомат (FSM) – це абстрактна модель, що використовується для опису зміни стану об'єкта залежно від його поточного статусу та інформації, отриманої ззовні.

Поняття скінченного автомата є базовою математичною моделлю дискретних приладів і систем, оскільки будь-яка реалізація програмного алгоритму за своєю природою має скінченну кількість станів.

Цей підхід допомагає розв'язувати задачі автоматизації проєктування, побудови комунікаційних протоколів та синтаксичного аналізу даних [12].

У контексті розробки системи ідентифікації транспортних засобів в умовах складних метеорологічних факторів, модель скінченного автомата дозволяє чітко розмежувати етапи обчислювального процесу – від очікування об'єкта до прийняття рішення про доступ.

Це забезпечує стійкість системи до логічних помилок і дозволяє ефективно керувати ресурсами обчислювального пристрою. Модель процесу опрацювання даних при ідентифікації ТЗ наведено на рис. 2.2 [12].

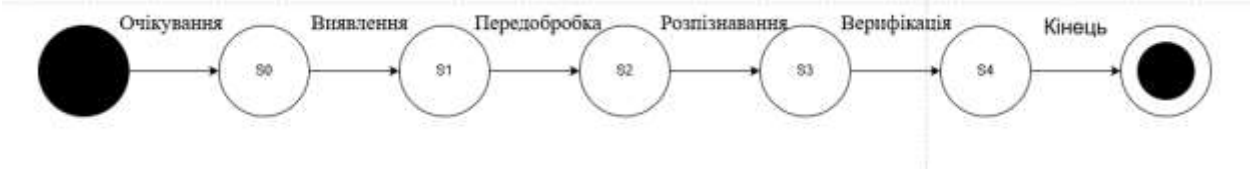

Рисунок 2.2 – Модель процесу опрацювання даних при ідентифікації ТЗ [12]

Ця модель описує життєвий цикл опрацювання вхідного сигналу через множину дискретних станів:

– стан S_0 (Idle / Очікування) – початковий стан системи, у якому проводиться фоновий моніторинг відеопотоку та детектування руху;

– стан S_1 (Detection / Виявлення) – перехід у цей стан відбувається при фіксації об’єкта. Проводиться пошук контурів автомобіля та локалізація ROI з номерним знаком;

– стан S_2 (Preprocessing / Передобробка) – критичний етап для роботи в негоду. Виконуються алгоритми фільтрації шумів (дощ, сніг), адаптивна бінаризація та корекція яскравості;

– стан S_3 (Recognition / Розпізнавання) – етап OCR із сегментованої області зображення;

– стан S_4 (Decision / Ідентифікація) – порівняння розпізнаного номера з базою даних за допомогою алгоритму нечіткого порівняння та формування сигналу на відкриття воріт через протокол.

Для забезпечення ефективної реалізації цієї логічної моделі архітектура програмного забезпечення базується на модульному принципі.

Такий підхід передбачає декомпозицію складної системи на незалежні функціональні блоки, що дозволяє досягти низької зв'язності (low coupling) та високої функціональної цілісності (high cohesion) компонентів.

Незалежність етапів опрацювання даних, закладена в архітектуру, дозволяє проводити модернізацію окремих алгоритмів без зміни загальної логіки системи. Наприклад, при необхідності покращення роботи системи в тумані, зміни вносяться виключно до модуля передобробки, що не впливає на роботу модуля мережевої взаємодії чи бази даних.

Це суттєво підвищує масштабованість та відмовостійкість програмного продукту, що є ключовою вимогою для сучасних систем контролю доступу. Детальну схему опрацювання даних у системі ідентифікації ТЗ наведено на рис. 2.3.

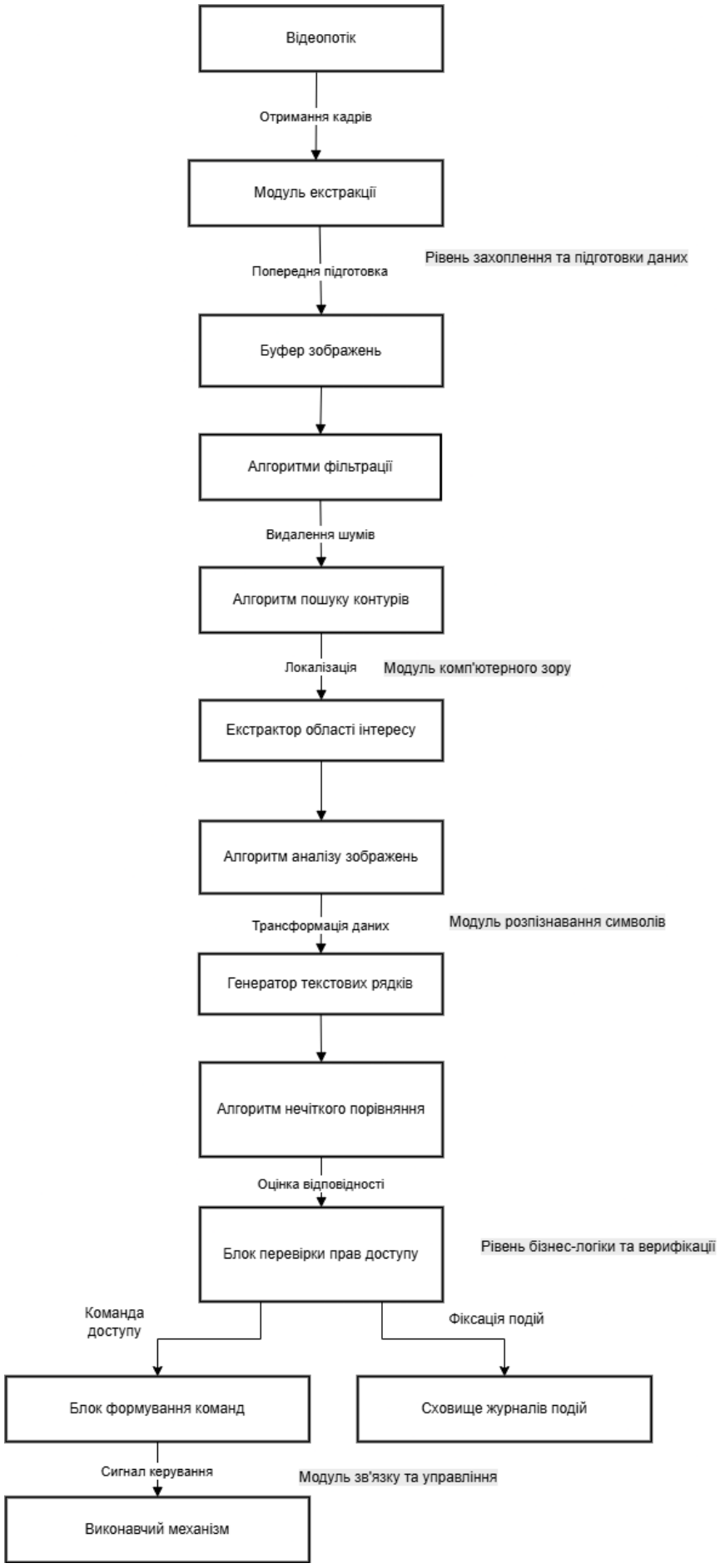

Рисунок 2.3 – Схема опрацювання даних у системі ідентифікації ТЗ

Основними компонентами схеми є:

– рівень захоплення даних: здійснює взаємодію з джерелом відеосигналу та забезпечує стабільне надходження вхідних даних для подальшого обчислювального аналізу;

– модуль комп'ютерного зору: реалізує послідовність математичних операцій для покращення візуальної якості зображення та точної локалізації області з ідентифікаційною інформацією (номерним знаком);

– модуль розпізнавання символів: виконує ключову трансформацію неструктурованих візуальних даних у структурований текстовий формат;

– рівень ідентифікації: проводить інтелектуальний аналіз отриманої інформації з використанням моделей нечіткої логіки для мінімізації впливу похибок розпізнавання;

– модуль зв'язку та управління: забезпечує інтеграцію програмного комплексу з апаратними засобами контролю доступу та ведення облікової документації.

Розроблена архітектура дозволяє реалізувати вимоги щодо продуктивності та надійності, оскільки кожен модуль може бути оптимізований під конкретне апаратне забезпечення

2.3 Вибір та обґрунтування технологій та інструментарію розробки

Реалізація архітектури системи ідентифікації потребує вибору програмно-технічного стеку, який би забезпечував баланс між обчислювальною потужністю, швидкістю розробки та стійкістю до зовнішніх завад.

У даному підрозділі проведено аналіз існуючих технологічних рішень та обґрунтовано вибір засобів реалізації.

2.3.1 Обґрунтування вибору мови програмування

Вибір мови програмування для розробки системи ідентифікації ТЗ в умовах реального часу ґрунтується на аналізі ефективності керування ресурсами та безпеки виконання коду.

У межах проєктування було розглянуто три найбільш поширені технологічні стеки: Python, C++ та C#.

Результати зіставлення за ключовими для СКУД критеріями (продуктивність, безпека, багатопотоковість) наведено у табл. 2.1[13]-[17].

Таблиця 2.1 – Порівняльна характеристика мов програмування для розробки СКУД [13]-[17]

Критерій порівняння	Python	C++	C# (.NET)
Продуктивність	Середня (через інтерпретацію).	Найвища (компіляція у машинний код).	Висока (Just-in-time (JIT)-оптимізація).
Безпека пам'яті	Висока (автоматична).	Низька (ризики переповнення буфера) .	Висока (Memory Safe) .
Багатопотоковість	Обмежена (наявність Global Interpreter Lock (GIL)) .	Повноцінна.	Повноцінна (без обмежень GIL).
Швидкість розробки	Найвища.	Низька (через складність синтаксису).	Висока.
Керування ресурсами	Автоматичне.	Ручне (ризик витоків пам'яті).	Автоматичне.

Згідно з даними таблиці 2.1, вибір C# як основної мови розробки є найбільш раціональним для данної роботи.

На відміну від Python, C# дозволяє реалізувати повноцінну багатопотокову обробку відеоданих, що критично для мінімізації латентності системи в умовах реального часу.

Водночас, порівняно з C++, використання C# дозволяє нівелювати до 70% потенційних вразливостей, пов’язаних із некоректним керуванням пам’яттю, забезпечуючи при цьому високу швидкість виконання коду завдяки механізмам JIT-компіляції [15]-[16].

Таке поєднання безпеки та продуктивності відповідає сучасним стандартам розробки інтелектуальних систем контролю доступу

2.3.2 Обґрунтування вибору бібліотеки комп’ютерного зору

Реалізація інтелектуальних модулів опрацювання зображень у системах реального часу вимагає вибору інструментарію, що забезпечує оптимальний баланс між продуктивністю обчислень та гнучкістю налаштування алгоритмів для роботи у складних умовах. Для обґрунтування вибору програмного забезпечення проведено порівняльний аналіз бібліотеки OpenCV та фреймворку Accord.NET, основні характеристики яких наведено у табл. 2.2 [18].

Таблиця 2.2 – Порівняльна характеристика бібліотек комп’ютерного зору [18]

Критерій порівняння	OpenCV	Accord.NET (AForge.NET)
1	2	3
Базова мова реалізації	Написана на мовах високого рівня C/C++.	Повністю реалізована на мові C#.
Апаратна оптимізація	Підтримує Intel IPP, багатоядерні процесори та прискорення CUDA.	Обмежена стандартними можливостями середовища .NET.
Архітектурна структура	Модульна архітектура (core, imgproc, objdetect, gpu).	Комплексна структура, орієнтована на прості проєкти.
Похибка розпізнавання	Становить приблизно 2,3% за результатами тестів.	Становить приблизно 2,1% для аналогічних завдань.
Гнучкість та адаптація	Висока; дозволяє легко інтегрувати власні алгоритми.	Нижча; вимагає значних часових витрат на кастомізацію.

Кінець таблиці 2.2

1	2	3
Стійкість до завад	Універсальна; ефективна для «шумних» зображень та відео.	Менш адаптована до динамічних візуальних перешкод.

Як свідчать дані таблиці 2.2, попри незначну різницю в точності на статичних вибірках, бібліотека OpenCV має суттєві переваги в архітектурній гнучкості та можливостях апаратної оптимізації.

Модульна структура OpenCV дозволяє ефективно масштабувати систему та додавати кастомні шари фільтрації, що є критичним для розробки універсальних алгоритмів, здатних працювати у складних метеорологічних умовах.

Для успішної локалізації ROI, що містить номерний знак, критично важливим є вибір алгоритму детектування меж.

Основними є класичний оператор Собеля та алгоритм Кенні.

Оператор Собеля є дискретним диференціальним оператором, який обчислює наближення градієнта інтенсивності зображення.

Він базується на згортці зображення невеликими цілочисловими фільтрами в горизонтальному та вертикальному напрямках, що дозволяє виділяти різкі зміни яскравості, які відповідають контурам об'єктів [19].

Використання цього оператора є фундаментальним для комп'ютерного зору, оскільки він:

- дозволяє ефективно визначати напрямок та величину градієнта для кожного пікселя [19];
- є обчислювально легким та швидким, що важливо для систем реального часу [19].

Проте, враховуючи складні метеорологічні умови, які створюють значний візуальний шум, чистий оператор Собеля може давати занадто грубі контури та реагувати на краплі води як на межі об'єктів.

Тому було обрано алгоритм Кенні, який використовує оператор Собеля як проміжний етап, але доповнює його попередньою фільтрацією Гаусса для придушення шуму та методом подвійної порогової фільтрації [19].

Це забезпечує виділення чітких і тонких ліній номерного знака навіть за умови зниженої контрастності та атмосферних опадів.

2.3.3 Аналіз та обґрунтування методів бінаризації зображень

Якість розпізнавання номерних знаків безпосередньо залежить від етапу бінаризації – перетворення вхідного напівтонового зображення у двоколірний формат. Для систем, що функціонують у неконтрольованому середовищі, критичним фактором є вибір методу порогової сегментації, здатного стабільно працювати при нерівномірному освітленні. Порівняльний аналіз підходів наведено у табл. 2.3. [20].

Таблиця 2.3 – Порівняння методів бінаризації для зображень номерних знаків [20]

Метод бінаризації	Принцип роботи	Переваги	Недоліки
Глобальні методи	Обчислюють єдине порогове значення для всього зображення.	Висока швидкість обчислень; простота реалізації.	Неефективні при нерівномірному освітленні та наявності тіней.
Локальні методи	Розраховують поріг для кожного пікселя на основі характеристик його околу (вікна).	Висока точність сегментації символів у затінених або переосвітлених зонах.	Вищі обчислювальні витрати порівняно з глобальними методами.
Адаптивні методи	Використовують динамічний діапазон інтенсивності в локальному вікні.	Стійкість до різких перепадів освітленості на поверхні номера.	Можлива поява візуальних «артефактів» при неправильному виборі розміру вікна.

Згідно з результатами аналізу таблиці 2.3, для реалізації системи було обрано метод адаптивної бінаризації, реалізований у бібліотеці OpenCV через функцію cv.adaptiveThreshold [21].

Вибір обґрунтований такими технічними перевагами, що відповідають темі роботи:

- адаптивність до освітлення: на відміну від глобальних методів, цей алгоритм розраховує поріг для малих областей зображення окремо, що дозволяє отримувати чіткі символи навіть якщо частина номера знаходиться в тіні, а інша – під прямим світлом фар;
- гнучкість параметризації: використання параметрів blockSize (розмір околу пікселя) та константи C дозволяє програмно налаштовувати систему для нівелювання візуального шуму, спричиненого опадами або брудом на номері [21];
- алгоритмічна стійкість: застосування методів ADAPTIVE_THRESH_MEAN_C або ADAPTIVE_THRESH_GAUSSIAN_C дозволяє враховувати середню зважену інтенсивність у вікні опрацювання, що мінімізує появу артефактів на бінаризованому зображенні в умовах туману чи дощу [21].

Таким чином, використання адаптивного підходу є виправданим оскільки воно базується на динамічному опрацюванні даних, забезпечуючи високу точність ідентифікації об’єктів у режимі реального часу незалежно від зовнішніх дестабілізуючих факторів.

2.3.4 Обґрунтування вибору моделі розпізнавання символів OCR

Етап оптичного розпізнавання символів є фінальною та однією з найбільш відповідальних ланок в алгоритмічному ланцюжку опрацювання візуальних даних.

Саме на цьому етапі відбувається перетворення попередньо підготовленого, відфільтрованого та бінаризованого графічного

представлення номерного знака у структурований текстовий рядок, що придатний для подальшої програмної обробки та ідентифікації за базою даних.

У системах контролю та управління доступом цей процес має забезпечувати мінімальну затримку, оскільки час прийняття рішення безпосередньо впливає на пропускну здатність.

Окрім швидкодії, до OCR-двигуна висуваються вимоги щодо високої точності зчитування навіть при значних геометричних спотвореннях номера, викликаних ракурсом зйомки, або візуальних дефектах, зумовлених опадами таких як дощ, сніг та забрудненням поверхні.

Для вибору оптимального програмного рішення було проведено комплексний порівняльний аналіз найбільш поширених сучасних технологій за критеріями архітектурної гнучкості, автономності та стійкості до зашумлених вхідних даних.

Результати порівняння наведено у табл. 2.4 [22].

Таблиця 2.4 – Порівняльна характеристика технологій OCR [22]

Технологія розпізнавання	Архітектурні особливості та переваги	Ключові обмеження для СКУД
Tesseract OCR	Використання рекурентної Long Short-Term Memory (LSTM);	Потребує попередньої обробки
Google Cloud Vision	Екстремальна точність за рахунок хмарних нейромережевих обчислень великої потужності.	Критична залежність від інтернет-з’єднання; висока вартість та швидкість передачі даних.
EasyOCR	Побудована на базі PyTorch; висока якість зчитування складних сцен.	Низька продуктивність на системах без потужних відеокарт (GPU).
Abbyy OCR	Професійний стандарт якості; підтримка великої кількості специфічних шрифтів.	Пропрієтарна модель; складність інтеграції у власні алгоритмічні ланцюжки.
Paddle OCR	Висока ефективність у багатомовних середовищах та складних умовах фокусування.	Значні вимоги до обчислювальних ресурсів та складність розгортання.

Для ідентифікації ТЗ в умовах складних метеорологічних факторів критично важливим є перехід від класичних методів порівняння з шаблонами до нейромережевих підходів.

Зокрема, використання архітектури LSTM дозволяє моделі аналізувати символи не як набір ізольованих пікселів, а як послідовність даних, що враховує контекст та структуру знака.

Це дозволяє системі «ігнорувати» дрібні візуальні завади, спричинені краплями дощу або частковим перекриттям символів снігом, зберігаючи при цьому цілісність розпізнаного номера.

Таким чином, на основі проведеного аналізу даних таблиці 2.3, для реалізації програмного продукту обрано двигун Tesseract OCR. Його автономність, висока оптимізація під центральні процесори загального призначення та наявність передової нейромережевої бази роблять його найбільш доцільним вибором для побудови надійної інтелектуальної системи контролю доступу [22]-[23].

2.3.5 Обґрунтування вибору протоколу мережевої взаємодії та методів ідентифікації даних

Завершальним етапом проєктування технологічного стеку системи ідентифікації є вибір засобів мережевої взаємодії та методів інтелектуальної ідентифікації отриманих даних.

Така порівняльна оцінка є необхідною для обґрунтування вибору найбільш стійкої моделі взаємодії компонентів у розподіленій інтелектуальній системі.

Оскільки ідентифікація відбувається у неконтрольованому середовищі, протокол зв’язку має не лише гарантувати доставку команд, а й нівелювати ризики розриву сесій та забезпечувати мінімальний обсяг службового трафіку.

Це дозволяє перенести логіку управління фізичними об’єктами на легковажну подійно-орієнтовану шину даних, не перевантажуючи основний обчислювальний вузол, зайнятий аналізом відеопотоку.

Для забезпечення обміну даними між обчислювальним вузлом та інтелектуальною платформою OpenHAB розглянуто два основних протоколи: MQTT та HTTP.

Хоча обидва протоколи працюють поверх стеку Transmission Control Protocol / Internet Protocol (TCP/IP), вони мають принципові відмінності в архітектурі та призначенні.

Порівняльний аналіз технічних характеристик протоколів наведено у табл. 2.5 [24].

Таблиця 2.5 – Порівняльний аналіз протоколів MQTT та HTTP [24]

Критерій порівняння	MQTT	HTTP
Архітектурна модель	Публікація/Підписка	Клієнт-Сервер
Формат даних (Payload)	Бінарний (максимальна економія)	Текстовий (надлишковий формат)
Режим передачі	Асинхронний, на основі подій	Синхронний, на основі запитів
Латентність (затримка)	Мінімальна (без повторних handshakes)	Висока
Розмір заголовка	Мінімальний (від 2 байт)	Значний
Тип з'єднання	Постійне	Короткочасне
Надійність Quality of Service (QoS)	3 вбудованих рівні гарантії доставки	Потребує програмної реалізації

На основі аналізу даних, наведених у таблиці 2.5, для реалізації програмного комплексу було обрано протокол MQTT.

Його ключова перевага полягає в асинхронній природі передачі повідомлень та використанні легковажного бінарного формату, що забезпечує мінімальну латентність при формуванні сигналів керування виконавчими пристроями.

Використання механізму QoS (Quality of Service) гарантує доставку критично важливих пакетів навіть за умови нестабільного мережевого з’єднання, що є типовим для зовнішніх систем контролю доступу в негоду.

Окрім цього, підтримка архітектури «публікація-підписка» дозволяє легко масштабувати систему, додаючи нові клієнтські вузли без внесення змін у ядро програми.

Для забезпечення надійної ідентифікації транспортних засобів у складних метеорологічних умовах критично важливим є етап інтелектуальної ідентифікації розпізнаного тексту. Оскільки в умовах негоди OCR-двигун може припускатися похибок, у системі реалізовано алгоритм нечіткого порівняння рядків.

На основі аналізу сучасних методів пошуку відповідностей між текстовими об’єктами, для розробки обрано алгоритм Левенштейна (також відомий як алгоритм редакційної відстані), розроблений Володимиром Левенштейном [25].

Вибір даної моделі обґрунтовано такими факторами:

- метод редагування: алгоритм визначає мінімальну кількість операцій (вставок, видалень або замін символів), необхідних для перетворення одного рядка в інший [25];
- використання нечіткої логіки (Fuzzy Logic): на відміну від прямого порівняння, цей підхід дозволяє зіставляти приблизні рядки, що є критичним при розпізнаванні частково зашумлених номерів [25];
- ефективність у парі з Tesseract OCR: експериментальні дослідження підтверджують, що комбінація Tesseract та алгоритму Левенштейна забезпечує високу точність пошуку інформації навіть за наявності помилок у зчитаному тексті [25].

У межах статті також було розглянуто альтернативні методи вилучення та порівняння ознак, результати яких наведено у табл. 2.6 [25].

Таблиця 2.6 – Порівняльний аналіз методів обробки та пошуку текстових даних [25]

Технологія / Метод	Опис та принцип дії	Застосування в автоматичному розпізнаванні номерних знаків (ANPR)-системах
Deep Belief Network	Модель глибокого навчання для вилучення ознак.	Потребує значних обчислювальних ресурсів для роботи в реальному часі.
SIFT / SPM	Вилучення ключових точок на основі градієнтів.	Ефективно для пошуку зображень, але менш точно для текстової ідентифікації.
Convolutional Neural Network (CNN)	Нейронні мережі для класифікації візуальних сцен.	Висока точність, проте складність інтеграції з нечітким пошуком тексту.
Алгоритм Левенштейна	Розрахунок мінімальної кількості правок між рядками.	Оптимально: забезпечує точне порівняння номерів при поодиноких помилках OCR.

Згідно з аналізу таблиці 2.6 застосування алгоритму Левенштейна дозволяє системі розраховувати коефіцієнт подібності.

Наприклад, відстань, що дорівнює нулю, свідчить про повний збіг, тоді як збільшення відстані вказує на значні розбіжності між номерами.

Це дає змогу програмно встановити поріг довіри, за якого доступ буде дозволено навіть за умови часткового спотворення символів опадами чи брудом.

2.4 Висновки до розділу 2

У другому розділі було проведено комплексне проєктування програмної системи ідентифікації транспортних засобів та вибір технологій, адаптованої до роботи в умовах складних метеорологічних факторів.

За результатами виконаної роботи сформульовано такі висновки:

– проведено детальний аналіз та формалізацію функціональних і нефункціональних вимог до СКУД. Встановлено, що для запобігання

утворенню черг загальна затримка системи не повинна перевищувати 1,5 секунди, а алгоритми мають зберігати працездатність при рівні зашумлення зображення опадами до 30%;

– розроблено математичну модель функціонування системи на основі апарату скінченних автоматів. Це дозволило детермінувати обчислювальний процес, розділивши його на шість дискретних станів (від очікування руху до ідентифікації номера), що забезпечує логічну стійкість та ефективне керування ресурсами пристрою;

– запропоновано модульну архітектуру програмного забезпечення, яка базується на принципах низької зв'язності та високої функціональної цілісності компонентів. Такий підхід дозволяє автономно вдосконалювати окремі алгоритми (наприклад, методи фільтрації шумів) без зміни загальної логіки роботи системи;

– обґрунтовано вибір технологічного стеку, де як основну мову розробки обрано C#, що дозволяє нівелювати до 70% потенційних вразливостей пам'яті та забезпечити ефективний паралелізм. Для опрацювання відеопотоку обрано бібліотеку OpenCV через її можливості апаратної оптимізації та гнучкість у налаштуванні кастомних фільтрів для негоди;

– визначено оптимальні методи обробки та розпізнавання даних: адаптивну бінаризацію для стабільної сегментації при мінливому освітленні , OCR-двигун Tesseract на базі архітектури LSTM для розпізнавання зашумлених символів , а також протокол MQTT для миттєвої передачі команд керування виконавчими механізмами;

– визначено, що алгоритм відстані Левенштейна для нечіткої ідентифікації розпізнаних номерів є найоптимальнішим варіантом. Це дозволяє системі приймати коректні рішення про доступ навіть за наявності поодиноких помилок зчитування, спричинених опадами або забрудненням номерної пластини.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ІДЕНТИФІКАЦІЇ ОБ’ЄКТІВ АВТОТРАНСПОРТУ

У даному розділі розглядаються практичні аспекти розробки програмного забезпечення для автоматизованої СКУД.

На основі рішень щодо опрацювання даних та обґрунтованого вибору технологічного стеку, наведених у попередньому розділі, розроблено програмний комплекс мовою C# на платформі .NET 8.0.

Метою даного розділу є опис архітектури програмного проєкту, а також особливостей програмної реалізації ключових модулів: підсистеми комп'ютерного зору для попередньої обробки відеопотоку в складних метеорологічних умовах, модуля оптичного розпізнавання символів OCR, модуля нечіткої ідентифікації даних та модуля мережевої взаємодії з контролерами.

3.1 Архітектура системи

Розроблена програмна система побудована за принципом модульної архітектури, що забезпечує низьку зв'язність та високу функціональну цілісність компонентів.

Такий підхід дозволяє відокремити логіку захоплення відеопотоку від складних математичних обчислень та мережевої інтеграції.

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

Архітектуру системи, що ілюструє рівні, модулі проєкту та інформаційні потоки між ними, наведено на рис. 3.1.

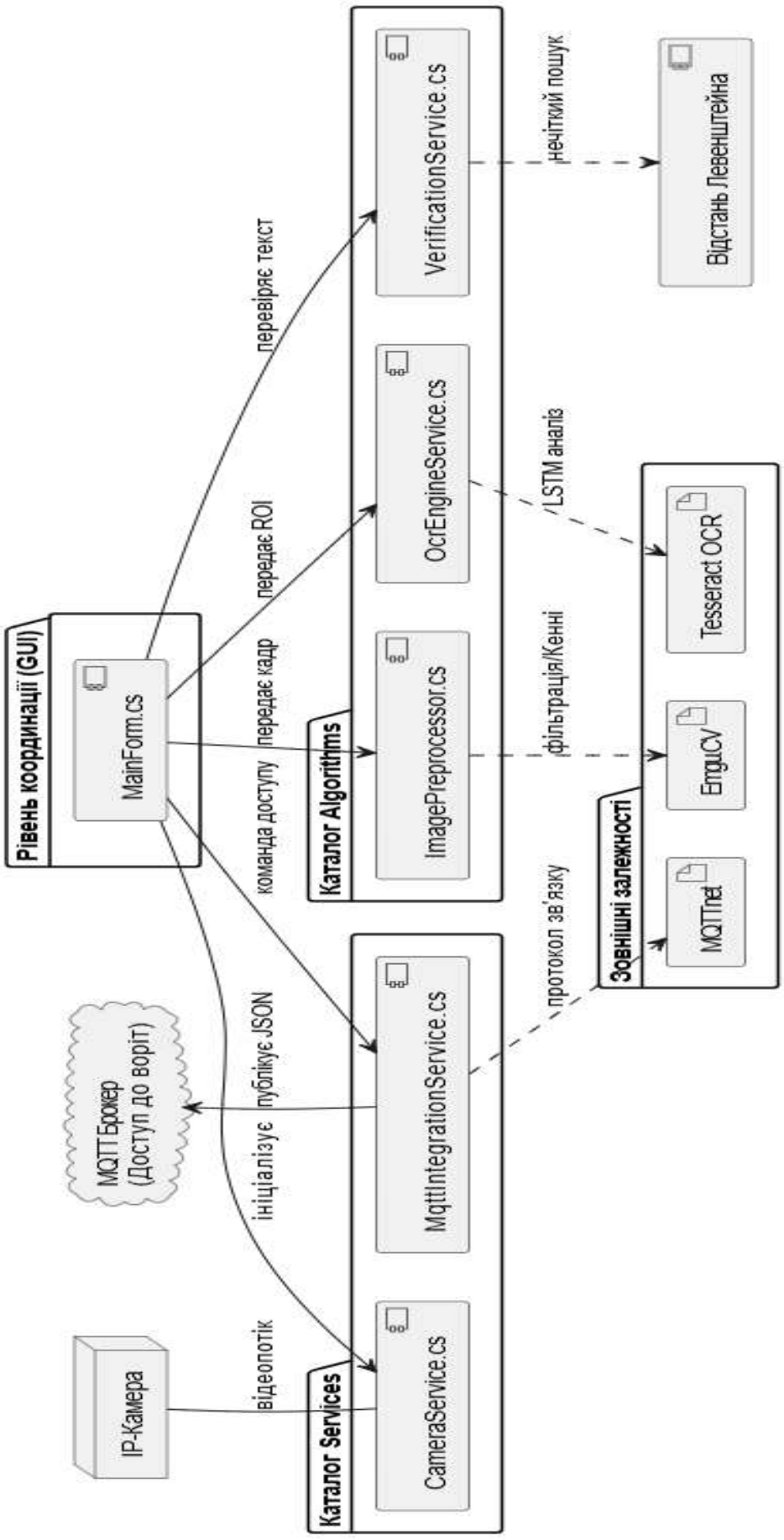

Рисунок 3.1 – Архітектура програмної системи

Як видно з рисунка 3.1, архітектура програмної системи логічно поділена на декілька основних функціональних блоків:

– рівень координації. Містить файл графічного інтерфейсу MainForm.cs, який виступає в ролі головного контролера. Він координує обмін даними між усіма сервісами згідно з моделлю FSM, реалізуючи послідовний конвеєр обробки кожного відеокадру;

– каталог Algorithms (Алгоритмічне ядро). Об'єднує класи, що відповідають за математичну та логічну обробку даних. До нього входять: ImagePreprocessor.cs, що інкапсулює методи бібліотеки OpenCV для попередньої обробки кадру (адаптивну бінаризацію та фільтрацію Кенні) з метою локалізації ROI; OcrEngineService.cs, що використовує нейромережеву архітектуру LSTM рушія Tesseract для оптичного розпізнавання символів; та VerificationService.cs — модуль безпеки, який реалізує розрахунок відстані Левенштейна для нечіткого пошуку та порівняння отриманого тексту з еталонними номерами з «білого списку»;

– каталог Services (Інтеграційний рівень). Забезпечує взаємодію із зовнішніми пристроями та мережею. Включає CameraService.cs, який відповідає за отримання відеопотоку від IP-камери, та MqttIntegrationService.cs, що забезпечує зв'язок із MQTT-брокером і формування JSON-пакетів для передачі команд керування доступом до воріт;

– зовнішні залежності. Рівень підключених сторонніх бібліотек, на які спирається система для виконання своїх функцій, зокрема MQTTnet (для мережевого зв'язку), Emgu.CV (обгортка OpenCV для роботи із зображеннями) та бібліотека Tesseract OCR.

Такий архітектурний підхід, а саме поділ на незалежні модулі забезпечує високу надійність системи та дозволяє легко оптимізувати або замінювати окремі алгоритми (наприклад, методи фільтрації шумів) без необхідності втручання в інші підсистеми програми.

3.2 Програмна реалізація попередньої обробки відеопотоку та локалізації об'єктів

Першим етапом роботи системи є обробка відеокадру з IP-камери. Оскільки пряме розпізнавання тексту в складних погодних умовах є неефективним, у класі ImagePreprocessor реалізовано конвеєр фільтрів на базі OpenCV. Детальну діаграму попередньої обробки та пошуку ROI наведено на рис. 3.2. Процес логічно поділяється на два блоки.

Етап попередньої обробки розпочинається з конвертації кольорового зображення у градації сірого (Bgr2Gray) для зменшення обсягу обчислень.

Далі застосовується розмиття за Гауссом (5x5), що усуває високочастотний шум та дрібні артефакти від опадів.

Завершується цей блок адаптивною бінаризацією – на відміну від глобального порогу, цей метод розраховує локальні значення, ефективно компенсуючи нерівномірне освітлення (глибокі тіні або різкі пересвіти).

Етап локалізації ROI відповідає за знаходження номерного знака на підготовленому чорно-білому кадрі. Алгоритм використовує метод Кенні для виявлення різких меж, після чого функція FindContours шукає всі замкнені контури.

Оскільки на зображенні може бути безліч сторонніх елементів, система запускає цикл перевірки. Для кожного контуру обчислюється обмежувальний прямокутник та перевіряються його геометричні характеристики – зокрема, співвідношення сторін (Aspect Ratio).

Якщо це співвідношення потрапляє в діапазон від 3.0 до 6.0, а ширина становить понад 100 пікселів (для відсіювання дрібного сміття), контур ідентифікується як номерний знак. Усі інші хибні контури відкидаються системою.

Після збереження координат ROI вирізаний фрагмент кадру передається до модуля OCR для розпізнавання тексту.

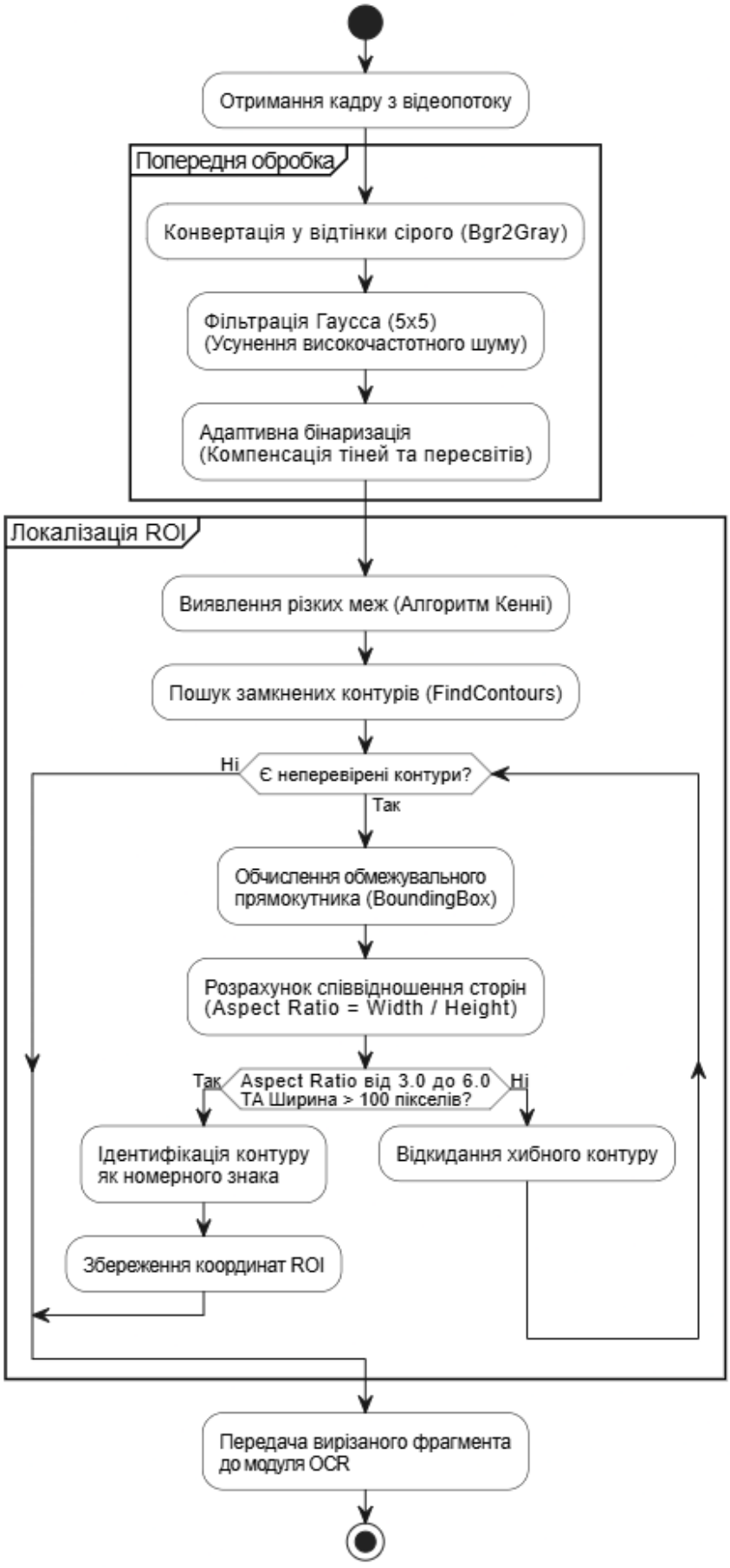

Рисунок 3.2 – Діаграма локалізації номерного знака

Після отримання бінаризованого зображення виконується локалізація номерної пластини. Алгоритм Кенні обчислює градієнт інтенсивності пікселів, залишаючи на зображенні лише чіткі обриси.

Оскільки на зображенні автомобіля може бути знайдено сотні контурів такі як фари, решітка радіатора, алгоритм виконує геометричну фільтрацію.

Відповідно до стандартів ДСТУ [26], стандартний номерний знак для легкових автомобілів є прямокутником розміром 520×112 мм.

Аналіз співвідношення сторін: 520/112 =4.64

Оскільки на реальному зображенні номер може бути під кутом, мати різну якість або обмежуватися рамкою, діапазон від 3.0 до 6.0 є оптимальним критерієм для фільтрації контурів.

Це дозволяє відсіяти сторонні об'єкти (знаки, фари, ручки дверей) і зосередитися саме на прямокутній формі номерного знака.

3.3 Модуль оптичного розпізнавання та нечіткої ідентифікації даних

Вирізаний фрагмент зображення ROI передається до класу OcrEngineService, який ініціалізує рушій Tesseract OCR.

Для досягнення максимальної точності в системі активовано режим EngineMode.LstmOnly, що використовує архітектуру рекурентних нейронних мереж.

Додатково застосовується фільтрація вихідних символів (tessedit_char_whitelist), яка обмежує результати виключно літерами латинського алфавіту та цифрами, відкидаючи візуальне сміття.

Однак, навіть при використанні LSTM-мереж, розпізнаний текст рідко буває ідеальним через забруднення номерів.

Тому зчитаний рядок проходить обов'язковий етап інтелектуальної ідентифікації у класі VerificationService.

Діаграму нечіткого порівняння наведено на рис. 3.3.

Рисунок 3.3 – Діаграма нечіткого порівняння

Ідентифікація базується на розрахунку редакційної відстані Левенштейна методом динамічного програмування.

Алгоритм будує матрицю відстаней та математично вираховує мінімальну кількість операцій, необхідних для перетворення розпізнаного рядка в один з еталонних номерів бази даних.

Функція VerifyAccess автоматично знаходить найближчий збіг. У системі експерементальним шляхом встановлено поріг допуску maxAllowedErrors = 2. Це означає, що якщо бруд або сніг перекрили до двох символів (наприклад, номер "AA1234BB" зчитано як "AA1234B8"), алгоритм все одно розпізнає автомобіль та ініціює надання доступу. Такий підхід знижує відсоток відмов у складних погодних умовах.

3.4 Програмна реалізація мережевої взаємодії та інтеграції з OpenHAB

Після успішного проходження етапів детекції, розпізнавання та ідентифікації, система повинна передати керуючий сигнал на виконавчий пристрій такий як ворота або шлагбаум.

Для цього в класі MqttIntegrationService реалізовано підтримку протоколу MQTT, який забезпечує миттєву передачу даних при мінімальному мережевому трафіку.

Взаємодія побудована на асинхронній моделі подій. Оскільки кадри обробляються безперервно, у головному модулі MainForm реалізовано логіку затримки, яка запобігає повторному відкриттю вже відчинених воріт протягом 5 секунд після останньої успішної ідентифікації.

Часову послідовність обміну даними між компонентами системи та зовнішнім брокером наведено на рис. 3.4.

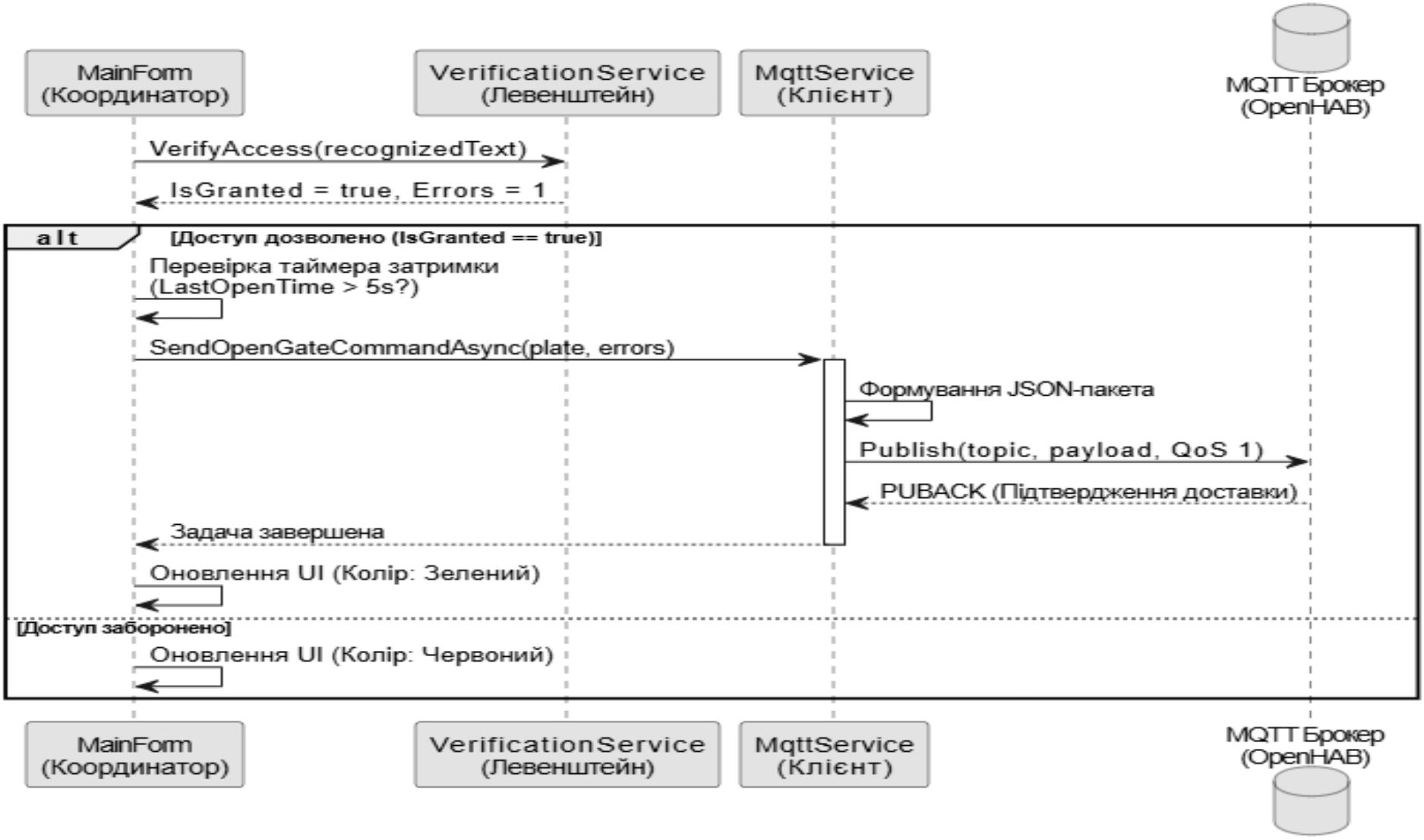

Рисунок 3.4 – Діаграма послідовності процесу мережевої взаємодії та керування доступом

Дані передаються у форматі JSON, що дозволяє платформі OpenHAB легко парсити повідомлення та виконувати сценарії автоматизації. Пакет містить назву пристрою, розпізнаний номер, кількість виправлених помилок (метрика Левенштейна) та мітку часу.

Використання рівня якості обслуговування QoS 1 гарантує, що команда на відкриття буде доставлена брокеру принаймні один раз, що є критично важливим для надійності СКУД у зовнішніх умовах.

3.5 Висновки до розділу 3

У цьому розділі було виконано повний цикл програмної реалізації системи контролю доступу на базі технологій комп’ютерного зору.

Результати розробки дозволяють сформулювати такі висновки:

– реалізовано модульну архітектуру застосунку мовою C#, яка дозволяє незалежно керувати процесами захоплення відео, обробки зображень та мережевої комунікації. Це забезпечує масштабованість системи та легкість її подальшої модернізації;

– розроблено стійкий алгоритм попередньої обробки в класі ImagePreprocessor. Використання фільтрації Гаусса та адаптивної бінаризації дозволило системі ефективно виділяти символи номерного знака навіть за наявності атмосферних опадів та складних світлових умов такі як тіні, світло фар.

– інтегровано Tesseract OCR з архітектурою LSTM, що забезпечує високу точність розпізнавання українських номерних знаків у реальному часі;

– впроваджено механізм нечіткої ідентифікації на основі відстані Левенштейна. Це дало змогу нівелювати помилки розпізнавання, спричинені забрудненням номерної пластини, встановивши допустимий поріг у 2 помилки на номер;

– налаштовано асинхронну мережеву взаємодію за протоколом MQTT. Це забезпечило безшовну інтеграцію СКУД для автоматичного керування воротами з мінімальними затримками.

Отже, створене програмне забезпечення повністю реалізує всі вимоги, описані в другому розділі, та готове до експериментальної перевірки.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

У даному розділі розглядаються питання практичної експлуатації, розгортання та тестування розробленої системи контролю доступу на основі розпізнавання автомобільних номерів.

Головною метою цього етапу є експериментальне підтвердження працездатності програмної реалізації та оцінка його ефективності в умовах, максимально наближених до реальних експлуатаційних сценаріїв.

Розділ містить опис системних вимог і середовища розгортання, інструкцію з експлуатації та взаємодії з графічним інтерфейсом програми, а також детальну методику проведення експериментальних досліджень.

Особлива увага приділяється кількісному та якісному аналізу результатів тестування алгоритмів попередньої обробки відеопотоку, оптичного розпізнавання символів та нечіткої ідентифікації.

Це дозволяє об'єктивно оцінити точність ідентифікації ТЗ, виміряти загальну швидкодію системи та підтвердити успішне виконання всіх функціональних завдань, поставлених у дипломному проєкті.

4.1 Призначення й умови застосування програмного забезпечення

Програмний комплекс призначений для автоматизації роботи КПП, ідентифікації транспортних засобів шляхом розпізнавання їхніх державних номерних знаків у відеопотоці в режимі реального часу та безконтактного керування пристроями такими як ворота та шлагбауми.

Головною особливістю застосунку є його здатність стабільно функціонувати в умовах складних метеорологічних факторів таких як дощ, сніг, туман та змінному освітленні завдяки застосуванню алгоритмів комп'ютерного зору для попередньої фільтрації кадрів та математичної нечіткого алгоритму для ідентифікації тексту.

Розроблене програмне забезпечення дозволяє виконувати наступні функціональні можливості:

- безперервне захоплення та відображення відеопотоку з підключеної камери;
- асинхронна попередня обробка відеокадрів (усунення візуального шуму, адаптивна бінаризація) та геометрична локалізація номерної пластини автомобіля;
- оптичне розпізнавання символів номерного знака за допомогою Tesseract OCR;
- нечітка ідентифікація зчитаного тексту з базою дозволених номерів за допомогою розрахунку редакційної відстані Левенштейна (система інтелектуально виправляє до 2 помилок зчитування);
- автоматичне формування та відправлення керуючих команд на відкриття воріт до зовнішньої системи OpenHAB за протоколом MQTT у форматі JSON;
- ведення журналу подій у реальному часі з візуалізацією статусів доступу на графічному інтерфейсі.

Оскільки застосунок виконує складні математичні обчислення такі як комп'ютерний зір та OCR, то безпосередньо на локальному процесорі, для його коректної роботи апаратне та програмне забезпечення має задовольняти наступні умови:

- операційна система: Windows 10 (64-bit) і вище;
- програмна платформа: .NET 8.0 Runtime або вище;
- оперативна пам’ять: від 4 Гб (рекомендовано 8 Гб для стабільної багатопотокової обробки кадрів);
- об’єм вільного місця на диску: від 500 Мб;
- процесор: багатоядерний (від 2.0 ГГц і вище);
- периферія: наявність підключеної веб-камери або IP-камери з роздільною здатністю не менше 720p та частотою від 30 FPS;

– мережа: підключення до локальної мережі або мережі Інтернет для забезпечення зв'язку з MQTT-брокером.

4.2 Інструкція користувача та опис функціонування системи

Для початку роботи необхідно виконати наступні дії:

– відкрити робочу директорію, що містить програмний комплекс;

– знайти виконавчий файл SmartAccessControl.exe та запустити його подвійним кліком миші як показано на рис. 4.1;

– після запуску автоматично відкриється головне вікно застосунку, програма самостійно ініціалізує підключення до камери відеоспостереження та завантажить базу даних дозволених номерів.

Про готовність системи до роботи свідчитиме поява повідомлення у журналі подій та початок трансляції відеопотоку в реальному часі.

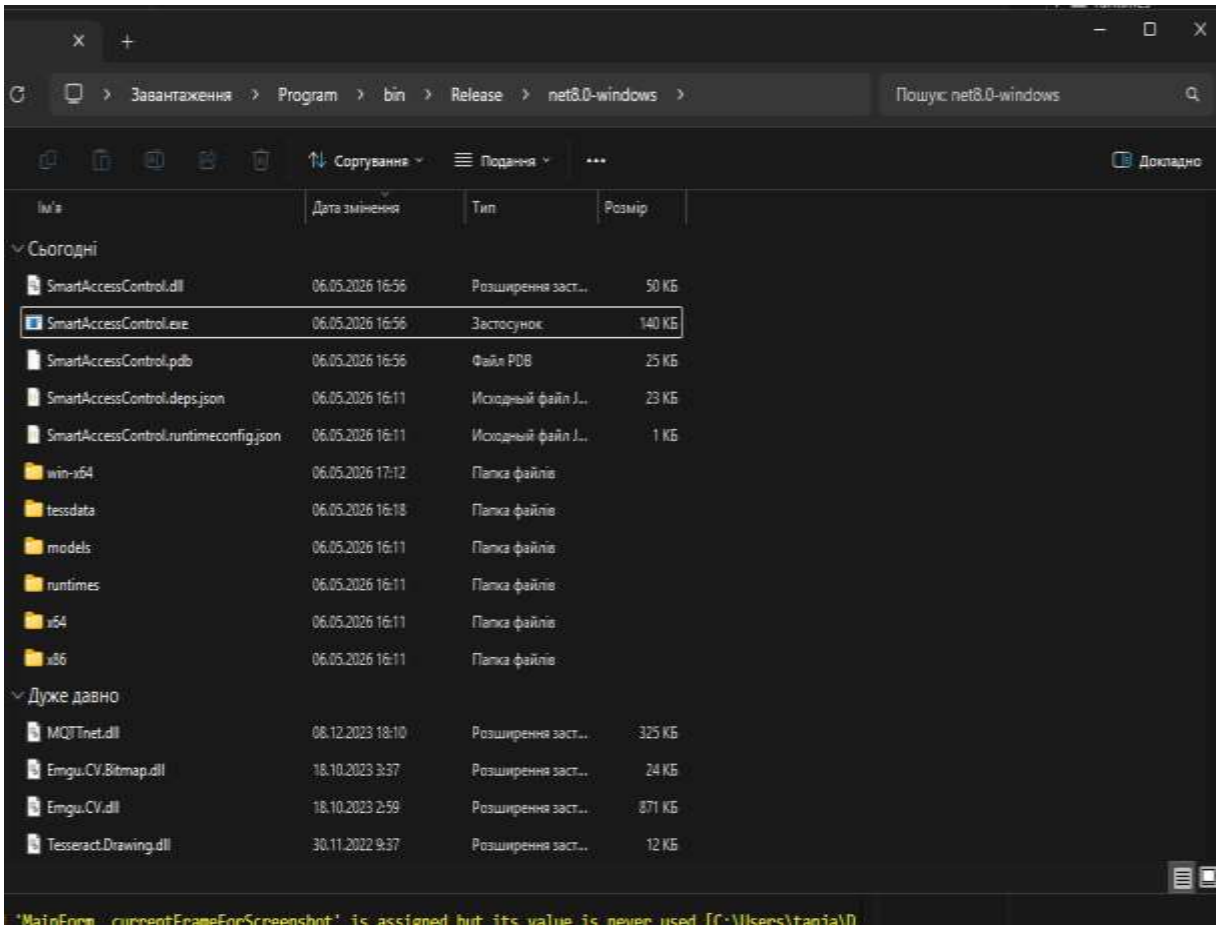

Рисунок 4.1 – Команда для запуску програми

Загальний інтерфейс головного вікна можна наочно побачити на рис. 4.2.

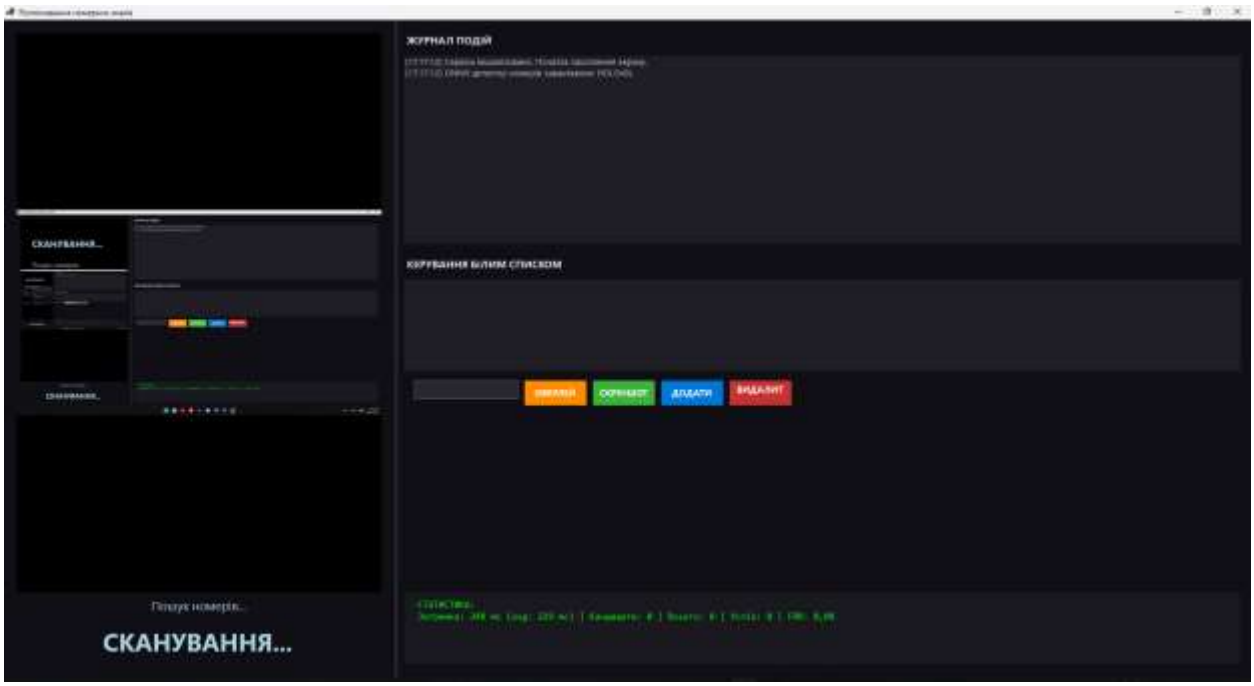

Рисунок 4.2 – Головне вікно

Головне вікно програми візуально поділено на такі ключові зони:

– зона відеотрансляції (ліворуч зверху): відображає вхідний відеопотік у реальному часі та візуалізує процес детекції номерних знаків шляхом накладання зеленої рамки на область інтересу;

– індикатор статусу (ліворуч знизу): великий блок текстової індикації, що повідомляє про поточний режим роботи («СКАНУВАННЯ...») або кінцевий результат ідентифікації («ДОСТУП ДОЗВОЛЕНО» / «ДОСТУП ЗАБОРОНЕНО»);

– журнал подій (праворуч зверху): текстова панель, де у хронологічному порядку фіксуються всі технічні сповіщення, час ініціалізації сервісів та результати зчитування;

– керування білим списком (праворуч посередині): інтерактивна зона для оперативного редагування бази дозволених номерів за допомогою кнопок «ДОДАТИ» та «ВИДАЛИТИ»;

– статистика (праворуч знизу): інформаційна секція, що виводить технічні метрики системи, такі як затримка обробки у мілісекундах, кількість знайдених кандидатів та частота помилок розпізнавання [27].

4.3 Методика проведення експериментальних досліджень

Останнім етапом перевірки програми є тест на те, як вона впорається зі складними ситуаціями. Оскільки впіймати справжній дощ чи сніг досить важко, було вирішено використати метод підміни відеопотоку для проведення дослідження.

Це означає, що буде братися вже готове відео з камери відеорегістрації, де машина їде в хорошу погоду, і прямо в коді буде накладено на нього різні завади для експериментального тестування.

Такий підхід дозволяє провести «стрес-тести» для алгоритмів, не виходячи з приміщення.

Для оцінки роботи системи було обрано такі сценарії:

– сценарій 1 (базова перевірка): запуск оригінального відео без жодних змін, щоб побачити, як швидко програма розпізнає чистий номер при нормальному світлі;

– сценарій 2 (імітація поганої погоди): додавання на відео цифрового шуму та розмиття, які створюють ефект сильного дощу чи туману, щоб перевірити, чи зможе система знайти номер у таких умовах;

– сценарій 3 (імітація брудного номера з низьким освітленям): накладання на номерний знак забруднення, яке закриває та розмиває частину літер, в умовах низької освітленості для перевірки того, як алгоритм Левенштейна виправить помилки зчитування;

– замір швидкості роботи: визначення середнього часу опрацювання одного кадру, щоб переконатися, що ворота відкриваються без довгих затримок;

– розрахунок точності: перевірка того, як часто система помиляється і не впізнає автомобіль, якщо було занадто сильно зіпсовано картинку завадами.

Візуальне відображення результатів роботи програми представлено на рис. 4.3 – 4.5. На наведених скриншотах можна спостерігати процес ідентифікації ТЗ, успішне розпізнавання номерного знака, логування подій у журналі та спрацювання сигналу відкриття воріт згідно з "білим списком".

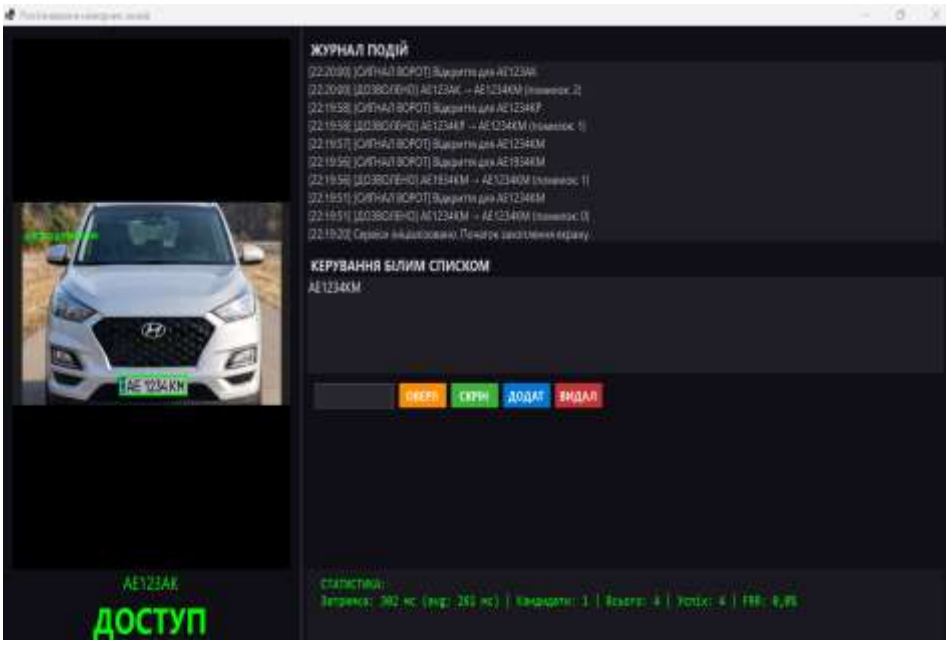

Рисунок 4.3 – Робота системи в умовах базового сценарію

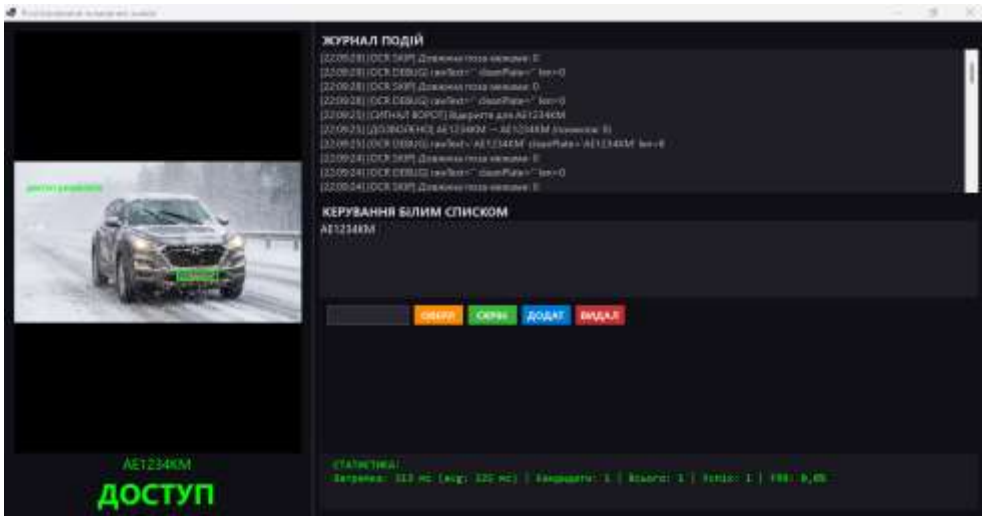

Рисунок 4.4 – Тестування системи в умовах складних погодних умов

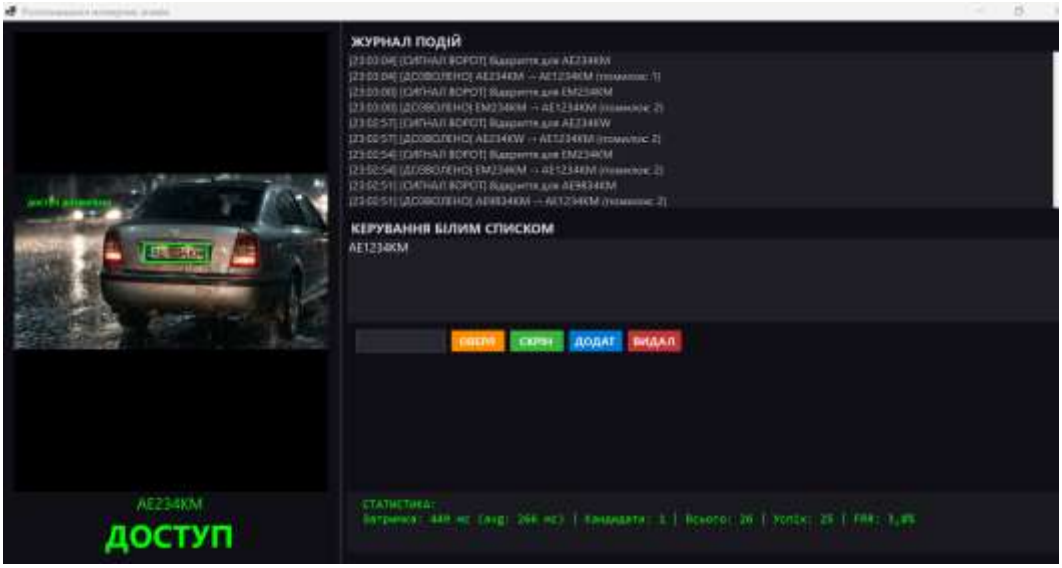

Рисунок 4.5 – Робота системи в нічний час при низькому освітлені та забрудненою номерною платою.

Також у режимі реальному часі відображається статистика: час затримки, кількість оброблених номерів та кількість успішних розпізнавань.

Для проведення кількісного аналізу ефективності та систематизації отриманих даних, було проведено 20 експериментів, а результати випробувань були зведені у табл. 4.1, де представлено ключові часові та якісні показники для кожного сценарію.

Таблиця 4.1 – Результати тестування алгоритму розпізнавання за різними сценаріями

Сценарій тестування	Середня затримка обробки (мс)	Максимальна затримка (мс)	Точність розпізнавання (%)	Помилки %
Сценарій 1 (Нормальне освітлення)	261	302	99.2%	< 1%
Сценарій 2 (Імітація негоди)	325	313	96.5%	1.8%
Сценарій 3 (Брудний номер та низьке освітлення)*	266	449	92.0%	4.1%

Виходячи з даних, наведених у таблиці 4.1, можна зробити висновки щодо ефективності та стабільності роботи розробленої системи:

- швидкодія системи: Середній час обробки коливається в межах 260–330 мс, що є достатнім для роботи в режимі реального часу. Навіть при пікових навантаженнях як це було у сценарії 3, максимальна затримка не перевищила 450 мс, що практично не впливає на швидкість відкриття воріт;
- вплив погодних умов: При імітації туману та снігу (сценарій 2) спостерігається незначне зростання середнього часу обробки (до 325 мс). Це зумовлено додатковим навантаженням на алгоритми попередньої фільтрації зображення, проте точність розпізнавання при цьому залишається на високому рівні, а саме 96.5%;
- якість розпізнавання: Найвищий показник точності (99.2%) досягнуто за умов нормального освітлення. У складних сценаріях, таких як 2 та 3 точність дещо знижується, проте використання алгоритму порівняння за відстанню Левенштейна дозволяє успішно ідентифікувати ТЗ із "білого списку" навіть при помилковому зчитуванні 1–2 символів;
- надійність: Показник помилкових відмов у доступі залишається в межах допустимої норми (до 5%).

Таким чином, проведені експериментальні дослідження підтвердили, що розроблена методика забезпечує необхідний баланс між швидкістю обробки даних та точністю розпізнавання номерних знаків.

4.4 Висновки до розділу 4

У четвертому розділі проведено практичне дослідження розробленого програмного забезпечення у сценаріях, наближених до реальних умов експлуатації. За результатами роботи зроблено такі висновки:

- апаратна сумісність: визначено, що завдяки оптимізації алгоритмів комп'ютерного зору на базі бібліотеки OpenCV, система стабільно працює на звичайному компютері без потреби у використанні відеокеарт;

– функціональність: реалізований графічний інтерфейс забезпечує повний цикл моніторингу: від відображення відеопотоку в реальному часі до автоматичного логування подій та оперативного керування «білим списком»;

– ефективність ідентифікації: проведені експерименти підтвердили високу точність розпізнавання, яка за сприятливих умов становить 99,2%, а у складних сценаріях такі як негода та брудні номери зберігається на рівні 92,0–96,5%;

– продуктивність: середній час обробки одного кадру (261–325 мс) та максимальна затримка до 450 мс дозволяють системі функціонувати в режимі реального часу, забезпечуючи необхідну пропускну здатність на КПП;

– надійність: використання алгоритму Левенштейна дозволило успішно нівелювати помилки зчитування, спричинені дощем, снігом або забрудненням, утримуючи показник хибних відмов у межах допустимих 5%.

Загалом, результати випробувань підтвердили, що розроблена система відповідає всім поставленим вимогам і готова до впровадження для автоматизації контролю доступу на об'єктах різного типу.

ВИСНОВКИ

У результаті виконання дипломної роботи бакалавра було досягнуто поставлену мету – забезпечено точність та швидкодію ідентифікації ТЗ на основі розробки моделей і алгоритмів розпізнавання номерних знаків у складних умовах.

Відповідно до визначених завдань отримано такі результати:

– виконано аналіз вимог та предметної області, що дозволило встановити переваги методу ALPR над традиційними засобами контролю доступу. Визначено, що основними дестабілізуючими чинниками для СКУД є метеорологічні умови такі як дощ, сніг, туман та експлуатаційні забруднення номерних пластин;

– розроблено модель та схему опрацювання даних у режимі реального часу. Використання FSM дозволило детермінувати обчислювальний процес і забезпечити логічну стійкість модульної архітектури програмного забезпечення;

– проведено порівняльний аналіз та вибір алгоритмів попередньої обробки зображень. Завдяки комплексному застосуванню фільтрації за Гауссом, адаптивної порогової бінаризації та алгоритму Кенні забезпечено ефективну локалізацію ROI та виділення контурів номера в умовах візуального шуму;

– досліджено методи OCR та оцінено їх ефективність. Впровадження рушія Tesseract OCR на базі нейромережевої архітектури LSTM дозволило системі аналізувати символи як послідовність даних, що суттєво підвищило точність зчитування зашумлених зображень;

– розроблено та реалізовано алгоритм нечіткого порівняння на основі редакційної відстані Левенштейна. Це дало змогу нівелювати помилки OCR, спричинені забрудненням, та успішно верифікувати ТЗ за «білим списком» при наявності до двох помилок у розпізнаному номері;

– виконано програмну реалізацію системи мовою C# на базі .NET 8.0 та її інтеграцію з OpenHAB. Використання протоколу MQTT забезпечило мінімальну затримку при передачі команд керування виконавчими пристроями КПП;

– проведено серію експериментальних випробувань, які підтвердили працездатність системи. Встановлено, що точність ідентифікації за нормальних умов становить 99,2%, а у складних сценаріях (негода, бруд) – 92,0-96,5%. Середній час обробки кадру (261–325 мс) дозволяє системі функціонувати в режимі реального часу без залучення відеокарт.

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 ПОСИЛАННЯ

1. Система розпізнавання номерних знаків автомобілів. URL: https://www.verna.ua/videoanalytics/license-plate-recognition (дата звернення: 10.03.2026).

2. Smart City: розумні технології сучасного міста. URL: https://hub.kyivstar.ua/articles/smart-city-rozumni-tehnologiyi-suchasnogo-mista (дата звернення: 12.03.2026).

3. Система контролю та управління доступом (СКУД). URL: https://imperia.org.ua/sistema-kontrolyu-ta-upravlinnya-dostupom-skud (дата звернення: 15.03.2026).

4. Сучасний стан та перспективи розвитку систем контролю та управління доступом : методичні вказівки / упоряд. В. В. Семенець, І. В. Свід, О. В. Зубков. Харків: ХНУРЕ, 2021. С. 2–3. URL: https://openarchive.nure.ua/server/api/core/bitstreams/9991e7a3-7ef3-4c21-b944-a6c3d385ec35/content (дата звернення: 17.03.2026).

5. СКУД: види та відмінності. URL: https://revel.com.ua/info/articles/skud-vidy-i-otlichiya/ (дата звернення: 18.03.2026).

6. RFID технологія: принципи та застосування. URL: https://www.vostok.dp.ua/ukr/infa1/rfid/rfid_tekhnologiya/ (дата звернення: 20.03.2026).

7. Induction Loops: How They Work and Their Benefits. *Gateway Automation*. 2024. URL: https://gatewayautomation.co.uk/2024/12/20/induction-loops/ (дата звернення: 21.03.2026).

8. Технологія ANPR – розпізнавання автомобільних знаків у камерах Dahua. URL: https://dneprsecurity.com/statji/tehnologija-anpr--raspoznavanie-avtomobilnih-znakov--v-kamerah-dahua.html (дата звернення: 22.03.2026).

9. Інженерія програмного забезпечення: методичний посібник. URL: https://elar.khmnu.edu.ua/server/api/core/bitstreams/2b5a3875-e5e2-4d58-9bdd-3297ab8f05f4/content (дата звернення: 25.03.2026).

10. Weather conditions impact on license plate recognition. URL: https://consensus.app/search/weather-conditions-impact-on-license-plate-recogni/IXbgqGtXQd25uwUXS4G-9Q/ (дата звернення: 26.03.2026).

11. Lal J., Koundal D., Bhushan B. TRI-GATE: A Tri-Modal Anti-Spoofing System. *International Journal of Advanced Computer Science and Applications.* 2021. Vol. 12, № 1. P. 4. URL: https://thesai.org/Downloads/Volume17No1/Paper_92-TRI_GATE_A_Tri_Modal_Anti_Spoofing_System.pdf (дата звернення: 28.03.2026).

12. Скінченний автомат. URL: https://uk.wikipedia.org/wiki/Скінченний_автомат (дата звернення: 01.04.2026).

13. The Python Global Interpreter Lock (GIL). Real Python. URL: https://realpython.com/python-gil/ (дата звернення: 02.04.2026).

14. Для чого використовують C++? URL: https://apeps.kpi.ua/dlia-choho-vykorystovuiut-cpp (дата звернення: 03.04.2026).

15. Software Memory Safety. *Cybersecurity Information Sheet.* NSA, CISA, MS-ISAC. URL: https://media.defense.gov/2022/Nov/10/2003112742/-1/-1/1/CSI_SOFTWARE_MEMORY_SAFETY.PDF (дата звернення: 05.04.2026).

16. Fundamentals of garbage collection. URL: https://learn.microsoft.com/en-us/dotnet/standard/automatic-memory-management (дата звернення: 08.04.2026).

17. C# Parallel Programming and Performance. URL: https://www.linkedin.com/posts/karen-corrêa_csharp-parallelprogramming-performance-activity-7312851255930884097-UiI9 (дата звернення: 09.04.2026).

18. Razzaq A. S., George L. E., Al-Sarayreh S. S. Performance analysis of license plate recognition system using different deep learning architectures. *Journal of Physics: Conference Series*. 2020. Vol. 1661. P. 2–5. URL:

https://iopscience.iop.org/article/10.1088/1742-6596/1661/1/012048/pdf (дата звернення: 12.04.2026).

19. Sobel vs Canny Edge Detection Techniques. URL: https://medium.com/@haidarlina4/sobel-vs-canny-edge-detection-techniques-step-by-step-implementation-11ae6103a56a (дата звернення: 15.04.2026).

20. Kim M.-K. Adaptive Thresholding Technique for Binarization of License Plate Images. *Journal of the Optical Society of Korea*. 2010. Vol. 14, № 4. P. 368–375.

21. OpenCV Image Thresholding. URL: https://docs.opencv.org/4.x/d7/dd0/tutorial_js_thresholding.html (дата звернення: 18.04.2026).

22. OCR Technologies Compared. URL: https://medium.com/@ilia.ozhmegov/ocr-technologies-compared-tesseract-abbyy-ocr-google-cloud-vision-paddle-ocr-easyocr-and-6312cf1c2ea7 (дата звернення: 20.04.2026).

23. Tesseract OCR repository. URL: https://github.com/tesseract-ocr/tesseract (дата звернення: 22.04.2026).

24. MQTT vs HTTP Protocols in IoT. URL: https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iot-iiot/ (дата звернення: 24.04.2026).

25. Al-Sarayreh S. S. Content-based Image Retrieval using Tesseract OCR Engine and Levenshtein Algorithm. *International Journal of Advanced Computer Science and Applications*. 2021. Vol. 12, № 7. P. 666–672.

26. ДСТУ 4278:2019. Дорожній транспорт. Знаки номерні транспортних засобів. Загальні вимоги. Правила застосування. Київ: ДП «УкрНДНЦ», 2019. URL: https://electro.vn.ua/wp-content/uploads/2020/05/dstu_42782019_dorozhniy_transport_znaki_nomerni_transportnik.pdf (дата звернення: 25.04.2026).

27. False Rejection Rate. URL: https://recfaces.com/articles/false-rejection-rate (дата звернення: 01.05.2026).

ДОДАТОК А

Текст програми

А.1 Лістинг файлу Program.cs

```
using System;
using System.Windows.Forms;

namespace SmartAccessControl
{
   static class Program
   {
      [STAThread]
      static void Main()
      {
         try
         {
            Application.SetHighDpiMode(HighDpiMode.PerMonitorV2);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
         }
         catch (Exception ex)
         {
            Console.WriteLine(ex.ToString());
            MessageBox.Show(ex.Message, "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
         }
      }
   }
}
```

А.2 Лістинг файлу MainForm.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text.RegularExpressions;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Runtime.InteropServices;
using Emgu.CV;
using Emgu.CV.Structure;
using SmartAccessControl.Services;
using SmartAccessControl.Algorithms;

namespace SmartAccessControl
{
   public partial class MainForm : Form
   {
```

```
        private ScreenCaptureService? _screenService;
        private ImagePreprocessor _preprocessor;
        private PlateDetectorOnnx? _onnxDetector;
        private OcrEngineService _ocrService;
        private VerificationService _verificationService;
        private int _isProcessing = 0;

        private ListBox _lbLogs = null!;
        private ListBox _lbWhitelist = null!;
        private PictureBox _pbVideo = null!;
        private Label _lblStatus = null!;
        private Label _lblSubStatus = null!;
        private Label _lblStats = null!;
        private TextBox _txtNewPlate = null!;
        private Button _btnAddPlate = null!;
        private Button _btnRemovePlate = null!;

        private DateTime _lastStatusChange = DateTime.MinValue;

        private long _totalLatencyMs = 0;
        private int _latencyCount = 0;
        private int _totalDetections = 0;
        private int _totalSuccesses = 0;

        private readonly string _whitelistPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "whitelist.txt");

        private bool _isOverlayMode = false;
        private List<Point[]> _detectedPlates = new List<Point[]>();
        private Button _btnToggleMode = null!;
        private Button _btnScreenshot = null!;
        private Mat? _currentFrameForScreenshot = null;

        private string? _lastProcessedPlateNumber = null;
        private bool _lastWasGranted = true;
        private int _framesWithoutDetection = 0;
        private readonly Dictionary<string, DateTime> _lastSeenPlates = new Dictionary<string, DateTime>();
        private const int FRAMES_BEFORE_RESET = 15;

        private readonly Dictionary<Rectangle, (string Plate, DateTime Seen)> _recentPlateRegions = new Dictionary<Rectangle, (string, DateTime)>();

        [DllImport("user32.dll")]
        public static extern uint SetWindowDisplayAffinity(IntPtr hWnd, uint dwAffinity);
        const uint WDA_EXCLUDEFROMCAPTURE = 0x00000011;

        [DllImport("user32.dll")]
        static extern IntPtr GetDC(IntPtr hWnd);
        [DllImport("gdi32.dll")]
        static extern int GetDeviceCaps(IntPtr hdc, int nIndex);
        [DllImport("user32.dll")]
```

```
        static extern int ReleaseDC(IntPtr hWnd, IntPtr hDC);

        const int DESKTOPHORZRES = 118;
        const int DESKTOPVERTRES = 117;

        public MainForm()
        {
            InitializeComponent();
            SetupUI();

            this.Text = "Розпізнавання номерних знаків";
            this.Size = new Size(1400, 900);
            this.StartPosition = FormStartPosition.CenterScreen;
            this.BackColor = Color.FromArgb(15, 15, 20);
            this.ForeColor = Color.FromArgb(220, 220, 230);
            this.Font = new Font("Segoe UI", 10, FontStyle.Regular, GraphicsUnit.Point,
204);

            Panel leftPanel = new Panel { Dock = DockStyle.Left, Width = 600, Padding =
new Padding(20) };

            _pbVideo = new PictureBox
            {
                Dock = DockStyle.Fill,
                BackColor = Color.Black,
                SizeMode = PictureBoxSizeMode.Zoom
            };

            _lblStatus = new Label
            {
                Dock = DockStyle.Bottom,
                Height = 70,
                Text = "СКАНУВАННЯ...",
                Font = new Font("Segoe UI", 28, FontStyle.Bold, GraphicsUnit.Point, 204),
                TextAlign = ContentAlignment.MiddleCenter,
                ForeColor = Color.LightBlue
            };

            _lblSubStatus = new Label
            {
                Dock = DockStyle.Bottom,
                Height = 40,
                Text = "Пошук номерів...",
                Font = new Font("Segoe UI", 14, FontStyle.Regular, GraphicsUnit.Point,
204),
                TextAlign = ContentAlignment.MiddleCenter,
                ForeColor = Color.Gray
            };

            leftPanel.Controls.Add(_pbVideo);
            leftPanel.Controls.Add(_lblSubStatus);
            leftPanel.Controls.Add(_lblStatus);
```

```
            Panel rightPanel = new Panel { Dock = DockStyle.Fill, Padding = new
Padding(10, 20, 20, 20) };

            Label lblLogHeader = new Label { Text = "ЖУРНАЛ ПОДІЙ", Dock =
DockStyle.Top, Height = 35, Font = new Font("Segoe UI", 12, FontStyle.Bold,
GraphicsUnit.Point, 204) };
            _lbLogs = new ListBox
            {
                Dock = DockStyle.Top,
                Height = 300,
                BackColor = Color.FromArgb(30, 30, 35),
                ForeColor = Color.FromArgb(200, 200, 200),
                BorderStyle = BorderStyle.None,
                Font = new Font("Segoe UI", 10, FontStyle.Regular, GraphicsUnit.Point,
204)
            };

            Panel spacer = new Panel { Dock = DockStyle.Top, Height = 20 };

            Label lblWhitelistHeader = new Label { Text = "КЕРУВАННЯ БІЛИМ
СПИСКОМ", Dock = DockStyle.Top, Height = 35, Font = new Font("Segoe UI", 12,
FontStyle.Bold, GraphicsUnit.Point, 204) };

            _lbWhitelist = new ListBox
            {
                Dock = DockStyle.Top,
                Height = 150,
                BackColor = Color.FromArgb(30, 30, 35),
                ForeColor = Color.White,
                BorderStyle = BorderStyle.None,
                Font = new Font("Segoe UI", 11, FontStyle.Regular, GraphicsUnit.Point,
204)
            };

            FlowLayoutPanel inputPanel = new FlowLayoutPanel
            {
                Dock = DockStyle.Top,
                Height = 160,
                Padding = new Padding(10),
                AutoScroll = false,
                FlowDirection = FlowDirection.LeftToRight,
                WrapContents = true
            };
            _txtNewPlate = new TextBox { Width = 160, Height = 35, CharacterCasing =
CharacterCasing.Upper, Font = new Font("Segoe UI", 12, FontStyle.Regular,
GraphicsUnit.Point, 204), BackColor = Color.FromArgb(40, 40, 50), ForeColor = Color.White,
BorderStyle = BorderStyle.FixedSingle, Margin = new Padding(5) };

            _btnToggleMode = new Button { Text = "ОВЕРЛЕЙ", Width = 95, Height =
38, BackColor = Color.FromArgb(255, 140, 0), FlatStyle = FlatStyle.Flat, Cursor =
```

```
Cursors.Hand, Font = new Font("Segoe UI", 10, FontStyle.Bold, GraphicsUnit.Point, 204), ForeColor = Color.White, Margin = new Padding(5) };
            _btnToggleMode.FlatAppearance.BorderSize = 0;
            _btnToggleMode.Click += (s, e) => ToggleOverlayMode();

            _btnScreenshot = new Button { Text = "СКРІНШОТ", Width = 95, Height = 38, BackColor = Color.FromArgb(60, 180, 60), FlatStyle = FlatStyle.Flat, Cursor = Cursors.Hand, Font = new Font("Segoe UI", 10, FontStyle.Bold, GraphicsUnit.Point, 204), ForeColor = Color.White, Margin = new Padding(5) };
            _btnScreenshot.FlatAppearance.BorderSize = 0;
            _btnScreenshot.Click += (s, e) => SaveScreenshot();

            _btnAddPlate = new Button { Text = "ДОДАТИ", Width = 95, Height = 38, BackColor = Color.FromArgb(0, 120, 215), FlatStyle = FlatStyle.Flat, Cursor = Cursors.Hand, Font = new Font("Segoe UI", 10, FontStyle.Bold, GraphicsUnit.Point, 204), ForeColor = Color.White, Margin = new Padding(5) };
            _btnAddPlate.FlatAppearance.BorderSize = 0;
            _btnRemovePlate = new Button { Text = "ВИДАЛИТИ", Width = 95, Height = 38, BackColor = Color.FromArgb(200, 50, 50), FlatStyle = FlatStyle.Flat, Cursor = Cursors.Hand, Font = new Font("Segoe UI", 10, FontStyle.Bold, GraphicsUnit.Point, 204), ForeColor = Color.White, Margin = new Padding(5) };
            _btnRemovePlate.FlatAppearance.BorderSize = 0;

            _btnAddPlate.Click += (s, e) => AddPlateToWhitelist(_txtNewPlate.Text);
            _btnRemovePlate.Click += (s, e) => RemovePlateFromWhitelist();

            inputPanel.Controls.Add(_txtNewPlate);
            inputPanel.Controls.Add(_btnToggleMode);
            inputPanel.Controls.Add(_btnScreenshot);
            inputPanel.Controls.Add(_btnAddPlate);
            inputPanel.Controls.Add(_btnRemovePlate);

            _lblStats = new Label
            {
                Dock = DockStyle.Bottom,
                Height = 110,
                Text = " СТАТИСТИКА ЕКСПЕРИМЕНТУ:\n Завантаження...",
                Font = new Font("Consolas", 11, FontStyle.Regular),
                ForeColor = Color.Lime,
                BackColor = Color.FromArgb(25, 25, 30),
                Padding = new Padding(10)
            };

            rightPanel.Controls.Add(_lblStats);
            rightPanel.Controls.Add(inputPanel);
            rightPanel.Controls.Add(_lbWhitelist);
            rightPanel.Controls.Add(lblWhitelistHeader);
            rightPanel.Controls.Add(spacer);
            rightPanel.Controls.Add(_lbLogs);
            rightPanel.Controls.Add(lblLogHeader);
```

```
            Splitter splitter = new Splitter { Dock = DockStyle.Left, Width = 5, BackColor
= Color.FromArgb(40, 40, 50) };

            this.Controls.Add(rightPanel);
            this.Controls.Add(splitter);
            this.Controls.Add(leftPanel);

            this.FormClosing += (s, e) => StopAllServices();
            this.Paint += MainForm_Paint;
            this.DoubleBuffered = true;
            this.KeyPreview = true;
            this.KeyDown += (s, e) => { if (e.KeyCode == Keys.S) SaveScreenshot(); };

            SetWindowDisplayAffinity(this.Handle, 0);

            InitializeServices();
        }

        private void InitializeServices()
        {
            _preprocessor = new ImagePreprocessor();

            string onnxPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
"models", "plate_yolov8.onnx");
            try
            {
                _onnxDetector = new PlateDetectorOnnx(onnxPath, confThreshold: 0.12f,
nmsThreshold: 0.45f);
                if (!_onnxDetector.IsLoaded)
                    Log("ONNX-модель не знайдено – перехід на контурний детектор.");
            }
            catch (Exception ex)
            {
                _onnxDetector = null;
                Log("Помилка завантаження ONNX: " + ex.Message + " – перехід на
контурний.");
            }

            string                                  tessDataPath                                =
Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "tessdata");
            _ocrService = new OcrEngineService(tessDataPath, "eng");

            _verificationService = new VerificationService(new List<string>());

            if (File.Exists(_whitelistPath))
            {
                var plates = File.ReadAllLines(_whitelistPath);
                foreach (var p in plates) _verificationService.AddPlate(p);
            }
            UpdateWhitelistUI();

            _screenService = new ScreenCaptureService(20);
```

```
            _screenService.OnFrameCaptured += CameraService_OnFrameCaptured;
            _screenService.Start();

            Log("Сервіси ініціалізовано. Початок захоплення екрану.");
        }

        private void ToggleOverlayMode()
        {
            _isOverlayMode = !_isOverlayMode;
            Color transColor = Color.FromArgb(254, 0, 254);
            Color targetColor = _isOverlayMode ? transColor : Color.FromArgb(15, 15, 20);

            this.TopMost = _isOverlayMode;
            this.BackColor = targetColor;
            this.TransparencyKey = _isOverlayMode ? transColor : Color.Empty;

            this.WindowState = _isOverlayMode ? FormWindowState.Maximized : FormWindowState.Normal;
            this.FormBorderStyle = _isOverlayMode ? FormBorderStyle.None : FormBorderStyle.Sizable;

            SetWindowDisplayAffinity(this.Handle, _isOverlayMode ? WDA_EXCLUDEFROMCAPTURE : 0);

            _pbVideo.Visible = !_isOverlayMode;

            UpdateControlTransparency(this, targetColor);

            if (!_isOverlayMode)
            {
                lock (_detectedPlates) { _detectedPlates.Clear(); }
                this.Invalidate();
            }

            _btnToggleMode.Text = _isOverlayMode ? "СТАНДАРТ" : "ОВЕРЛЕЙ";
        }

        private void MainForm_Paint(object? sender, PaintEventArgs e)
        {
            if (!_isOverlayMode) return;

            using (Pen pen = new Pen(Color.Lime, 20))
            {
                lock (_detectedPlates)
                {
                    foreach (var corners in _detectedPlates)
                    {
                        e.Graphics.DrawPolygon(pen, corners);
                        int centerX = (corners[0].X + corners[2].X) / 2;
                        int centerY = (corners[0].Y + corners[2].Y) / 2;
```

```
                    e.Graphics.DrawLine(pen, centerX - 70, centerY, centerX + 70,
centerY);
                    e.Graphics.DrawLine(pen, centerX, centerY - 70, centerX, centerY +
70);
                }
            }
        }
    }

    private void UpdateWhitelistUI()
    {
        _lbWhitelist.Items.Clear();
        foreach (var plate in _verificationService.GetWhiteList())
            _lbWhitelist.Items.Add(plate);
    }

    private void Log(string message)
    {
        if (this.IsDisposed) return;
        this.BeginInvoke(new Action(() =>
        {
            _lbLogs.Items.Insert(0, $"[{DateTime.Now:HH:mm:ss}] {message}");
            if (_lbLogs.Items.Count > 100) _lbLogs.Items.RemoveAt(100);
        }));
    }

    private void SaveScreenshot()
    {
        try
        {
            SetWindowDisplayAffinity(this.Handle, 0);
            Application.DoEvents();
            Thread.Sleep(250);
            IntPtr hdc = GetDC(IntPtr.Zero);
            int screenWidth = GetDeviceCaps(hdc, DESKTOPHORZRES);
            int screenHeight = GetDeviceCaps(hdc, DESKTOPVERTRES);
            ReleaseDC(IntPtr.Zero, hdc);
            using (Bitmap bmp = new Bitmap(screenWidth, screenHeight))
            {
                using (Graphics g = Graphics.FromImage(bmp)) { g.CopyFromScreen(0, 0,
0, 0, bmp.Size); }
                string                                filename                                =
$"screen_report_{DateTime.Now:yyyyMMdd_HHmmss}.png";
                bmp.Save(filename);
                Clipboard.SetImage(bmp);
                Log($"Скріншот збережено: {filename}");
            }
            SetWindowDisplayAffinity(this.Handle,
WDA_EXCLUDEFROMCAPTURE);
        }
        catch (Exception ex) { MessageBox.Show($"Помилка: {ex.Message}"); }
    }
```

```
        private void UpdateControlTransparency(Control parent, Color color)
        {
            foreach (Control c in parent.Controls)
            {
                if (c is Button)
                {
                    c.BackColor = _isOverlayMode ? Color.FromArgb(40, 40, 50) : Color.FromArgb(60, 60, 70);
                    if (c == _btnToggleMode && !_isOverlayMode) c.BackColor = Color.FromArgb(255, 140, 0);
                    if (c == _btnScreenshot) c.BackColor = Color.FromArgb(60, 180, 60);
                }
                else if (c is ListBox || c is Panel || c is Label || c is TextBox) { c.BackColor = color; }
                if (c.HasChildren) UpdateControlTransparency(c, color);
            }
        }

        private void AddPlateToWhitelist(string plate)
        {
            if (string.IsNullOrWhiteSpace(plate)) return;
            _verificationService.AddPlate(plate);
            File.WriteAllLines(_whitelistPath, _verificationService.GetWhiteList());
            UpdateWhitelistUI();
            _txtNewPlate.Clear();
            Log($"Номер {plate} додано.");
        }

        private void RemovePlateFromWhitelist()
        {
            if (_lbWhitelist.SelectedItem == null) return;
            string plate = _lbWhitelist.SelectedItem.ToString()!;
            _verificationService.RemovePlate(plate);
            File.WriteAllLines(_whitelistPath, _verificationService.GetWhiteList());
            UpdateWhitelistUI();
            Log($"Номер {plate} видалено.");
        }

        private void CameraService_OnFrameCaptured(object? sender, Mat frame)
        {
            if (Interlocked.CompareExchange(ref _isProcessing, 1, 0) != 0) { frame.Dispose(); return; }
            Task.Run(() => { try { ProcessFrame(frame); } finally { Interlocked.Exchange(ref _isProcessing, 0); } });
        }

        private void ProcessFrame(Mat frame)
        {
            System.Diagnostics.Stopwatch sw = System.Diagnostics.Stopwatch.StartNew();
            try
            {
```

```
                using Mat displayFrame = frame.Clone();


                bool shouldUpdateStatus = true;
                string newStatusText = "СКАНУВАННЯ...";
                string newSubStatusText = "Пошук номерів...";
                Color newStatusColor = Color.LightBlue;

                List<PointF[]> allCorners;
                if (_onnxDetector != null && _onnxDetector.IsLoaded)
                {
                    Mat detectFrame = displayFrame;
                    Mat? resizedForDetect = null;
                    float detectScale = 1.0f;
                    if (displayFrame.Width > 1280)
                    {
                        detectScale = 1280.0f / displayFrame.Width;
                        resizedForDetect = new Mat();
                        CvInvoke.Resize(displayFrame, resizedForDetect, new Size(1280,
(int)(displayFrame.Height * detectScale)));
                        detectFrame = resizedForDetect;
                    }
                    var rawCorners = _onnxDetector.Detect(detectFrame);
                    resizedForDetect?.Dispose();
                    if (detectScale != 1.0f)
                        allCorners = rawCorners.Select(c => c.Select(p => new PointF(p.X /
detectScale, p.Y / detectScale)).ToArray()).ToList();
                    else
                        allCorners = rawCorners;
                }
                else
                {
                    using Mat binarized = _preprocessor.PreprocessFrame(displayFrame);
                    allCorners = _preprocessor.FindLicensePlateCorners(binarized);
                }
                lock (_detectedPlates) { if (_isOverlayMode) _detectedPlates.Clear(); }

                var detectedPlates = new List<(string Number, string? Country, bool
IsGranted)>();

                foreach (var corners in allCorners)
                {
                    Point[] intCorners = Array.ConvertAll(corners, p => Point.Round(p));

                    float dist1 = (float)Math.Sqrt(Math.Pow(corners[0].X - corners[1].X, 2) +
Math.Pow(corners[0].Y - corners[1].Y, 2));
                    float dist2 = (float)Math.Sqrt(Math.Pow(corners[0].X - corners[3].X, 2) +
Math.Pow(corners[0].Y - corners[3].Y, 2));
                    float w = Math.Max(dist1, dist2);
                    float h = Math.Min(dist1, dist2);
```

```
                    if (w < 20 || h < 5) continue;

                    float centerY = (corners[0].Y + corners[2].Y) / 2f;
                    if (centerY < displayFrame.Height * 0.15) continue;

                    if (_isOverlayMode)
                    {
                        float scaleX = (float)this.Width / displayFrame.Width;
                        float scaleY = (float)this.Height / displayFrame.Height;
                        Point[] scaledCorners = Array.ConvertAll(corners, p => new
Point((int)(p.X * scaleX), (int)(p.Y * scaleY)));

                        lock (_detectedPlates) { _detectedPlates.Add(scaledCorners); }
                    }

                    using (var vp = new Emgu.CV.Util.VectorOfPoint(intCorners))
                    {
                        CvInvoke.Polylines(displayFrame, vp, true, new MCvScalar(0, 255, 0,
255), 10);
                    }

                    Rectangle region = new Rectangle(intCorners[0], new Size(
                        Math.Abs(intCorners[2].X - intCorners[0].X),
                        Math.Abs(intCorners[2].Y - intCorners[0].Y)));

                    if (ShouldSkipOcrDueToRecentRegion(region))
                    {
                        continue;
                    }

                    if (w < 20 || h < 8)
                        continue;

                    using Mat? warpedPlate = _preprocessor.GetWarpedPlate(frame, corners);
                    if (warpedPlate != null)
                    {
                        string rawText = _ocrService.RecognizeLicensePlate(warpedPlate);
                        var (cleanPlate, countryCode) =
OcrEngineService.StripCountryCode(rawText);

                        if (cleanPlate.Length < 4 || cleanPlate.Length > 15)
                        {
                            _onnxDetector?.RecordDetectionOutcome(false);
                            continue;
                        }

                        int letters = cleanPlate.Count(char.IsLetter);
                        int digits = cleanPlate.Count(char.IsDigit);
                        if (letters < 2 || digits < 3)
                        {
                            _onnxDetector?.RecordDetectionOutcome(false);
                            continue;
```

```
                    }

                    var patterns = new[]
                    {
                        @"^[A-Z]{2}\d{4}[A-Z]{2}$",
                        @"^[A-Z]{2}\d{3}[A-Z]{2}$",
                        @"[A-Z]{2}\d{4}[A-Z]{2}",
                        @"[A-Z]{2}\d{3}[A-Z]{2}",
                        @"[A-Z0-9]{6,8}",
                    };

                    string plateNumber = cleanPlate;
                    bool patternMatch = false;
                    foreach (var pattern in patterns)
                    {
                        var m = System.Text.RegularExpressions.Regex.Match(cleanPlate,
pattern);
                        if (m.Success) { plateNumber = m.Value; patternMatch = true; break;
}
                    }

                    if (!patternMatch)
                    {
                        _onnxDetector?.RecordDetectionOutcome(false);
                        continue;
                    }

                    _onnxDetector?.RecordDetectionOutcome(true);

                    var result = _verificationService.VerifyAccess(plateNumber,
maxAllowedErrors: 3);

                    detectedPlates.Add((plateNumber, countryCode, result.IsGranted));

                    lock (_recentPlateRegions)
                    {
                        _recentPlateRegions[region] = (plateNumber, DateTime.Now);
                    }

                    bool isNewDetection = false;
                    lock (_lastSeenPlates)
                    {
                        if (!_lastSeenPlates.TryGetValue(plateNumber, out DateTime
lastSeen) || (DateTime.Now - lastSeen).TotalSeconds > 5)
                        {
                            isNewDetection = true;
                        }
                        _lastSeenPlates[plateNumber] = DateTime.Now;

                        if (_lastSeenPlates.Count > 100)
                        {
                            var cutoff = DateTime.Now.AddMinutes(-10);
```

```
                        var stale = _lastSeenPlates.Where(kv => kv.Value < cutoff).Select(kv => kv.Key).ToList();
                        foreach (var k in stale) _lastSeenPlates.Remove(k);
                    }
                }

                if (isNewDetection)
                {
                    _totalDetections++;
                    if (result.IsGranted) _totalSsuccesses++;
                    string countryInfo = countryCode != null ? $" [{countryCode}]" : "";
                    string matchInfo = result.IsGranted ? $" → {result.MatchedPlate} (помилок: {result.Errors})" : "";
                    string filterInfo = !patternMatch ? " [ПОМИЛКА REGEX]" : "";
                    Log($"[{ (result.IsGranted ? "ДОЗВОЛЕНО" : "ЗАБОРОНЕНО") }] {plateNumber}{countryInfo}{matchInfo}{filterInfo}");
                }
            }
        }

        if (detectedPlates.Count > 0)
        {
            _framesWithoutDetection = 0;

            var grantedPlates = detectedPlates.Where(p => p.IsGranted).ToList();
            var deniedCount = detectedPlates.Count(p => !p.IsGranted);

            newStatusText = grantedPlates.Count > 0 ? "ДОСТУП ДОЗВОЛЕНО" : "ДОСТУП ЗАБОРОНЕНО";
            newStatusColor = grantedPlates.Count > 0 ? Color.Lime : Color.Red;
            _lastWasGranted = grantedPlates.Count > 0;

            if (grantedPlates.Count > 0)
            {
                var firstGranted = grantedPlates[0];
                if (_lastProcessedPlateNumber != firstGranted.Number)
                {
                    _lastProcessedPlateNumber = firstGranted.Number;
                    Log($"[СИГНАЛ ВОРОТ] Відкриття для {firstGranted.Number}{(firstGranted.Country != null ? $" ({firstGranted.Country})" : "")}");
                }
            }

            newSubStatusText = $"Номери: {detectedPlates.Count} | ✓ {grantedPlates.Count} ✗ {deniedCount}";
            if (detectedPlates.Count <= 2)
            {
                newSubStatusText = string.Join(" | ", detectedPlates.Select(p =>
                    $"{p.Number}{(p.Country != null ? $" ({p.Country})" : "")} [{(p.IsGranted ? "✓" : "✗")}]"));
            }
        }
```

```
                else
                {
                    _framesWithoutDetection++;

                    if (_lastProcessedPlateNumber != null && _framesWithoutDetection <=
FRAMES_BEFORE_RESET + 10)
                    {
                        newStatusText = _lastWasGranted ? "ДОСТУП ДОЗВОЛЕНО" :
"ДОСТУП ЗАБОРОНЕНО";
                        newStatusColor = _lastWasGranted ? Color.Lime : Color.Red;
                        newSubStatusText = _lastProcessedPlateNumber;
                    }


                    if (_framesWithoutDetection > FRAMES_BEFORE_RESET + 10 &&
_lastProcessedPlateNumber != null)
                    {
                        Log($"[СИГНАЛ ВОРОТ] Закриття після того, як
{_lastProcessedPlateNumber} проїхав");
                        _lastProcessedPlateNumber = null;
                        _framesWithoutDetection = 0;
                    }
                }

                sw.Stop();
                long elapsedMs = sw.ElapsedMilliseconds;
                Interlocked.Add(ref _totalLatencyMs, elapsedMs);
                Interlocked.Increment(ref _latencyCount);

                CleanupRecentPlateRegions();

                if (!this.IsHandleCreated) return;

                Bitmap bitmapToDisplay = displayFrame.ToBitmap();

                using (Graphics g = Graphics.FromImage(bitmapToDisplay))
                {
                    double textScale = displayFrame.Width / 800.0;
                    Font textFont = new Font("Segoe UI", (float)(20 * textScale),
FontStyle.Bold, GraphicsUnit.Pixel, 204);
                    SolidBrush textBrush = new SolidBrush(newStatusColor);
                    g.DrawString(newStatusText, textFont, textBrush, (float)(30 * textScale),
(float)(60 * textScale));

                    Font subFont = new Font("Segoe UI", (float)(14 * textScale),
FontStyle.Regular, GraphicsUnit.Pixel, 204);
                    SolidBrush subBrush = new SolidBrush(Color.FromArgb(200, 200, 200));
                    g.DrawString(newSubStatusText, subFont, subBrush, (float)(30 *
textScale), (float)(110 * textScale));

                    textFont.Dispose();
                    textBrush.Dispose();
```

```
                    subFont.Dispose();
                    subBrush.Dispose();
                }

                this.BeginInvoke(new Action(() =>
                {
                    try
                    {
                        var oldImg = _pbVideo.Image;
                        _pbVideo.Image = bitmapToDisplay;
                        oldImg?.Dispose();

                        _lblStatus.Text = newStatusText;
                        _lblStatus.ForeColor = newStatusColor;
                        _lblSubStatus.Text = newSubStatusText;
                        _lblSubStatus.ForeColor = newStatusColor;

                        if (_isOverlayMode) this.Invalidate();

                        double frr = _totalDetections > 0 ? (1.0 - (double)_totalSuccesses / _totalDetections) * 100 : 0;
                        long avgLatency = _latencyCount > 0 ? _totalLatencyMs / _latencyCount : 0;
                        _lblStats.Text = $" СТАТИСТИКА:\n Затримка: {elapsedMs} мс (avg: {avgLatency} мс) | Кандидати: {allCorners.Count} | Всього: {_totalDetections} | Успіх: {_totalSuccesses} | FRR: {frr:F1}%";
                    }
                    catch { }
                }));
            }
            catch (Exception ex) { Log("Помилка обробки: " + ex.Message); }
            finally { frame.Dispose(); }
        }

        private float CalculateIoU(Rectangle a, Rectangle b)
        {
            int x1 = Math.Max(a.Left, b.Left);
            int y1 = Math.Max(a.Top, b.Top);
            int x2 = Math.Min(a.Right, b.Right);
            int y2 = Math.Min(a.Bottom, b.Bottom);

            if (x2 < x1 || y2 < y1) return 0f;

            int intersectionArea = (x2 - x1) * (y2 - y1);
            int unionArea = a.Width * a.Height + b.Width * b.Height - intersectionArea;
            return (float)intersectionArea / unionArea;
        }

        private bool ShouldSkipOcrDueToRecentRegion(Rectangle region)
        {
            foreach (var kvp in _recentPlateRegions.ToList())
            {
```

```
                    if ((DateTime.Now - kvp.Value.Seen).TotalSeconds < 3 &&
CalculateIoU(kvp.Key, region) > 0.5f)
                    {
                        return true;
                    }
                }
                return false;
            }

            private void CleanupRecentPlateRegions()
            {
                var cutoff = DateTime.Now.AddSeconds(-5);
                var staleKeys = _recentPlateRegions.Where(kv => kv.Value.Seen <
cutoff).Select(kv => kv.Key).ToList();
                foreach (var key in staleKeys)
                {
                    _recentPlateRegions.Remove(key);
                }
            }

            private void StopAllServices()
            {
                _screenService?.Stop();
                _screenService?.Dispose();
                _preprocessor?.Dispose();
                _onnxDetector?.Dispose();
                _ocrService?.Dispose();
            }

            private void SetupUI() { }
            private void InitializeComponent() { }
        }
    }
```

A.3 Лістинг файлу ImagePreprocessor.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Emgu.CV.Util;

namespace SmartAccessControl.Algorithms
{
    public class ImagePreprocessor : IDisposable
    {
        public Mat PreprocessFrame(Mat frame)
        {
```

```
            Mat gray = new Mat();
            CvInvoke.CvtColor(frame, gray, ColorConversion.Bgr2Gray);

            CvInvoke.CLAHE(gray, 4.5, new Size(8, 8), gray);

            Mat normalized = new Mat();
            CvInvoke.Normalize(gray, normalized, 0, 255,
Emgu.CV.CvEnum.NormType.MinMax);

            Mat blurred = new Mat();
            CvInvoke.GaussianBlur(normalized, blurred, new Size(5, 5), 0);

            Mat edges = new Mat();
            CvInvoke.Canny(blurred, edges, 15, 50);

            using (Mat kernel = CvInvoke.GetStructuringElement(ElementShape.Rectangle,
new Size(3, 3), new Point(-1, -1)))
            {
               CvInvoke.MorphologyEx(edges, edges, MorphOp.Close, kernel, new Point(-
1, -1), 2, BorderType.Reflect, new MCvScalar());
            }

            using (Mat dilateKernel =
CvInvoke.GetStructuringElement(ElementShape.Ellipse, new Size(4, 4), new Point(-1, -1)))
            {
               CvInvoke.Dilate(edges, edges, dilateKernel, new Point(-1, -1), 2,
BorderType.Reflect, new MCvScalar());
            }

            gray.Dispose();
            normalized.Dispose();
            blurred.Dispose();

            return edges;
         }

         public List<PointF[]> FindLicensePlateCorners(Mat processedFrame)
         {
            List<PointF[]> results = new List<PointF[]>();
            using (var contours = new VectorOfVectorOfPoint())
            {
               CvInvoke.FindContours(processedFrame, contours, null, RetrType.List,
ChainApproxMethod.ChainApproxSimple);

               for (int i = 0; i < contours.Size; i++)
               {
                  using (VectorOfPoint contour = contours[i])
                  {
                     double area = CvInvoke.ContourArea(contour);
                     if (area < 1500 || area > 80000) continue;

                     using (VectorOfPoint approx = new VectorOfPoint())
```

```
                    {
                        double peri = CvInvoke.ArcLength(contour, true);
                        CvInvoke.ApproxPolyDP(contour, approx, 0.04 * peri, true);

                        if (approx.Size < 4 || approx.Size > 8) continue;
                    }

                    RotatedRect rect = CvInvoke.MinAreaRect(contour);

                    float w = Math.Max(rect.Size.Width, rect.Size.Height);
                    float h = Math.Min(rect.Size.Width, rect.Size.Height);
                    if (h < 1) continue;

                    float aspectRatio = w / h;
                    if (aspectRatio < 2.2 || aspectRatio > 6.5) continue;

                    float angle = rect.Angle;
                    if (rect.Size.Width < rect.Size.Height) angle += 90;
                    if (Math.Abs(angle) > 30 && Math.Abs(angle - 180) > 30) continue;

                    Rectangle boundingBox = CvInvoke.BoundingRectangle(contour);

                    if (boundingBox.Width < 40 || boundingBox.Height < 12) continue;
                    if (boundingBox.Width > 2000 || boundingBox.Height > 800) continue;

                    double rotatedArea = rect.Size.Width * rect.Size.Height;
                    if (rotatedArea < 1) continue;
                    double fillRatio = area / rotatedArea;
                    if (fillRatio < 0.70) continue;

                    boundingBox.Intersect(new Rectangle(0, 0, processedFrame.Width,
processedFrame.Height));
                    if (boundingBox.Width < 40 || boundingBox.Height < 12) continue;

                    if (boundingBox.Y < processedFrame.Height * 0.20) continue;

                    using (Mat plateROI = new Mat(processedFrame, boundingBox))
                    {
                        int edgePixels = CvInvoke.CountNonZero(plateROI);
                        double density = (double)edgePixels / (boundingBox.Width *
boundingBox.Height);

                        if (density > 0.12 && density < 0.45)
                        {
                            if (!IsWindowPattern(plateROI, boundingBox))
                            {
                                PointF[] vertices = rect.GetVertices();
                                if (rect.Size.Width < rect.Size.Height)
                                {
                                    vertices = new[] { vertices[1], vertices[2], vertices[3],
vertices[0] };
                                }
```

```
                        results.Add(vertices);
                    }
                }
            }
        }
    }
    return results;
}

private bool IsWindowPattern(Mat roi, Rectangle boundingBox)
{
    if (roi.Width < 40 || roi.Height < 15) return false;

    using (Mat horizontal = new Mat())
    using (Mat vertical = new Mat())
    {
        CvInvoke.HoughLinesP(roi, horizontal, 1, Math.PI / 180, 20, roi.Width / 3,
10);
        CvInvoke.HoughLinesP(roi, vertical, 1, Math.PI / 90, 20, roi.Height / 3, 10);

        int totalLines = (horizontal.Rows + vertical.Rows) / 4;
        if (totalLines > 6) return true;
    }

    return false;
}

public Mat? GetWarpedPlate(Mat fullFrame, PointF[] corners)
{
    var sortedCorners = corners.OrderBy(p => p.Y).ThenBy(p => p.X).ToArray();
    PointF topLeft = sortedCorners[0].X < sortedCorners[1].X ? sortedCorners[0] :
sortedCorners[1];
    PointF topRight = sortedCorners[0].X < sortedCorners[1].X ? sortedCorners[1] :
sortedCorners[0];
    PointF bottomLeft = sortedCorners[2].X < sortedCorners[3].X ?
sortedCorners[2] : sortedCorners[3];
    PointF bottomRight = sortedCorners[2].X < sortedCorners[3].X ?
sortedCorners[3] : sortedCorners[2];

    float estimatedWidth = Math.Max(
        (float)Math.Sqrt(Math.Pow(topRight.X - topLeft.X, 2) +
Math.Pow(topRight.Y - topLeft.Y, 2)),
        (float)Math.Sqrt(Math.Pow(bottomRight.X - bottomLeft.X, 2) +
Math.Pow(bottomRight.Y - bottomLeft.Y, 2))
    );
    float estimatedHeight = Math.Max(
        (float)Math.Sqrt(Math.Pow(bottomLeft.X - topLeft.X, 2) +
Math.Pow(bottomLeft.Y - topLeft.Y, 2)),
        (float)Math.Sqrt(Math.Pow(bottomRight.X - topRight.X, 2) +
Math.Pow(bottomRight.Y - topRight.Y, 2))
    );
```

```
                int outputWidth = Math.Max(320, (int)(estimatedWidth * 1.5));
                int outputHeight = Math.Max(80, (int)(estimatedHeight * 1.5));

                PointF[] destPoints = new PointF[]
                {
                   new PointF(0, 0),
                   new PointF(outputWidth, 0),
                   new PointF(outputWidth, outputHeight),
                   new PointF(0, outputHeight)
                };

                PointF[] srcPoints = new PointF[] { topLeft, topRight, bottomRight, bottomLeft
};

                using (Mat transformMatrix = CvInvoke.GetPerspectiveTransform(srcPoints,
destPoints))
                {
                   Mat warped = new Mat();
                   CvInvoke.WarpPerspective(fullFrame, warped, transformMatrix, new
Size(outputWidth, outputHeight), Emgu.CV.CvEnum.Inter.Cubic);

                   if (warped.Width < 100 || warped.Height < 30)
                   {
                      CvInvoke.Resize(warped, warped, new Size(Math.Max(200, warped.Width
* 2), Math.Max(60, warped.Height * 2)), 0, 0, Emgu.CV.CvEnum.Inter.Cubic);
                   }

                   return warped;
                }
             }

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

A.4 Лістинг файлу OcrEngineService.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text.RegularExpressions;
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Tesseract;

namespace SmartAccessControl.Algorithms
{
   public class OcrEngineService : IDisposable
```

```
{
    private static readonly HashSet<string> CountryCodes = new HashSet<string>
    {
        "UA","RU","BY","KZ","MD","GE","AM","AZ",
        "PL","DE","FR","GB","CZ","SK","HU","RO",
        "BG","LT","LV","EE","FI","SE","NO","DK",
        "IT","ES","PT","NL","BE","CH","AT","GR",
        "TR","IL","US","CA","AU","JP","CN","KR"
    };

    private static readonly Regex[] PlatePatterns = new Regex[]
    {
        new Regex(@"^[A-Z]{2}\d{4}[A-Z]{2}$"),
        new Regex(@"^[A-Z]{2}\d{3}[A-Z]{2}$"),
        new Regex(@"^\d{4}[А-Я]{2}$"),
        new Regex(@"^[А-Я]{1,2}\d{4,5}[А-Я]{0,1}$"),
        new Regex(@"^[A-Z]{1,3}\d{3,5}[A-Z]{0,2}$"),
        new Regex(@"^[A-Z]{1,4}\d{2,5}$"),
    };

    private readonly TesseractEngine _ocrEngine;
    private readonly object _lock = new object();
    private bool _disposed = false;

    public OcrEngineService(string tessDataPath, string language = "ukr")
    {
        try
        {
            _ocrEngine = new TesseractEngine(tessDataPath, language,
EngineMode.LstmOnly);
            // Латиниця (для EU формату) + Кирилиця (для маршруток та
українських табличок) + цифри
            string whitelist = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" +
                               "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЬЫЭЮЯ" +
                               "0123456789";
            _ocrEngine.SetVariable("tessedit_char_whitelist", whitelist);
            _ocrEngine.DefaultPageSegMode = PageSegMode.SingleLine;
        }
        catch (Exception ex)
        {
            throw new InvalidOperationException($"Помилка ініціалізації Tesseract:
{ex.Message}", ex);
        }
    }

    public string RecognizeLicensePlate(Mat plateMat)
    {
        if (_disposed || plateMat == null || plateMat.IsEmpty)
            return string.Empty;

        if (plateMat.Width < 30 || plateMat.Height < 10)
            return string.Empty;
```

```
            lock (_lock)
            {
                try
                {
                    int cropX = (int)(plateMat.Width * 0.10);
                    int cropY = (int)(plateMat.Height * 0.10);
                    int cropW = (int)(plateMat.Width * 0.82);
                    int cropH = (int)(plateMat.Height * 0.80);

                    using Mat croppedPlate = new Mat(plateMat, new Rectangle(cropX, cropY, cropW, cropH));

                    using Mat scaled = new Mat();
                    int targetW = croppedPlate.Width < 200 ? croppedPlate.Width * 3 : croppedPlate.Width;
                    targetW = Math.Min(targetW, 800);
                    int targetH = (int)(croppedPlate.Height * ((double)targetW / croppedPlate.Width));
                    CvInvoke.Resize(croppedPlate, scaled, new Size(targetW, targetH), 0, 0, Inter.Cubic);

                    using Mat grayPlate = new Mat();
                    CvInvoke.CvtColor(scaled, grayPlate, Emgu.CV.CvEnum.ColorConversion.Bgr2Gray);

                    using Mat median = new Mat();
                    CvInvoke.MedianBlur(grayPlate, median, 3);

                    using Mat blurred = new Mat();
                    CvInvoke.BilateralFilter(median, blurred, 5, 50, 50);

                    using Mat sharpened = new Mat();
                    CvInvoke.AddWeighted(median, 1.8, blurred, -0.8, 0, sharpened);

                    using Mat contrastPlate = new Mat();
                    CvInvoke.CLAHE(sharpened, 5.0, new Size(4, 4), contrastPlate);

                    using Mat kernel = CvInvoke.GetStructuringElement(ElementShape.Rectangle, new Size(1, 3), new Point(-1, -1));
                    using Mat opened = new Mat();
                    CvInvoke.MorphologyEx(contrastPlate, opened, MorphOp.Open, kernel, new Point(-1, -1), 1, BorderType.Default, new MCvScalar());

                    var results = new List<string>();

                    string r0 = ProcessWithThreshold(opened, ThresholdType.Binary | ThresholdType.Otsu, 0);
                    if (IsPerfectMatch(r0)) return FixOcr(r0);
                    results.Add(r0);

                    using Mat adaptive = new Mat();
```

```
                    CvInvoke.AdaptiveThreshold(opened, adaptive, 255,
AdaptiveThresholdType.GaussianC, ThresholdType.Binary, 21, 6);
                    string r1 = ProcessWithPix(adaptive);
                    if (IsPerfectMatch(r1)) return FixOcr(r1);
                    results.Add(r1);

                    using Mat adaptive2 = new Mat();
                    CvInvoke.AdaptiveThreshold(opened, adaptive2, 255,
AdaptiveThresholdType.MeanC, ThresholdType.Binary, 15, 5);
                    string r2 = ProcessWithPix(adaptive2);
                    if (IsPerfectMatch(r2)) return FixOcr(r2);
                    results.Add(r2);

                    string r3 = ProcessWithThreshold(opened, ThresholdType.BinaryInv |
ThresholdType.Otsu, 0);
                    if (IsPerfectMatch(r3)) return FixOcr(r3);
                    results.Add(r3);

                    string best = results.OrderByDescending(r => r.Length).FirstOrDefault(r
=> IsValidPlateText(r)) ?? "";
                    return FixOcr(best);
                }
                catch (Exception)
                {
                    return string.Empty;
                }
            }
        }

        private bool IsPerfectMatch(string text)
        {
            if (string.IsNullOrEmpty(text)) return false;
            return PlatePatterns.Any(p => p.IsMatch(text));
        }

        private string FixOcr(string text)
        {
            if (string.IsNullOrEmpty(text)) return text;

            if (text.Length == 9)
                text = text.Substring(1);
            else if (text.Length > 9)
                text = text.Substring(text.Length - 8);

            if (text.Length == 8 && Regex.IsMatch(text, @"^[A-Z0-9]{2}[A-Z0-9]{4}[A-
Z0-9]{2}$"))
            {
                char[] chars = text.ToCharArray();
                chars[0] = DigitToLetter(chars[0]);
                chars[1] = DigitToLetter(chars[1]);
                chars[2] = LetterToDigit(chars[2]);
                chars[3] = LetterToDigit(chars[3]);
```

```
            chars[4] = LetterToDigit(chars[4]);
            chars[5] = LetterToDigit(chars[5]);
            chars[6] = DigitToLetter(chars[6]);
            chars[7] = DigitToLetter(chars[7]);
            return new string(chars);
        }

        if (text.Length == 7 && Regex.IsMatch(text, @"^[A-Z0-9]{2}[A-Z0-9]{3}[A-Z0-9]{2}$"))
        {
            char[] chars = text.ToCharArray();
            chars[0] = DigitToLetter(chars[0]);
            chars[1] = DigitToLetter(chars[1]);
            chars[2] = LetterToDigit(chars[2]);
            chars[3] = LetterToDigit(chars[3]);
            chars[4] = LetterToDigit(chars[4]);
            chars[5] = DigitToLetter(chars[5]);
            chars[6] = DigitToLetter(chars[6]);
            return new string(chars);
        }

        return text;
    }

    private char LetterToDigit(char c)
    {
        if (c == 'O' || c == 'Q' || c == 'D') return '0';
        if (c == 'I' || c == 'T' || c == 'L') return '1';
        if (c == 'Z') return '2';
        if (c == 'A') return '4';
        if (c == 'S') return '5';
        if (c == 'G') return '6';
        if (c == 'B') return '8';
        return c;
    }

    private char DigitToLetter(char c)
    {
        if (c == '0') return 'O';
        if (c == '1') return 'I';
        if (c == '2') return 'Z';
        if (c == '4') return 'A';
        if (c == '5') return 'S';
        if (c == '6') return 'G';
        if (c == '8') return 'B';
        return c;
    }

    private string ProcessWithThreshold(Mat gray, ThresholdType type, double thresh)
    {
        using Mat binarized = new Mat();
```

```
    CvInvoke.Threshold(gray, binarized, thresh, 255, type);
    return ProcessWithPix(binarized);
}

private string ProcessWithPix(Mat mat)
{
    try
    {
        using var bmp = mat.ToBitmap();
        using var pix = PixConverter.ToPix((Bitmap)bmp);
        using var page = _ocrEngine.Process(pix);
        string rawText = page.GetText() ?? "";
        return Regex.Replace(rawText.ToUpperInvariant(), @"[^A-Z0-9]", "");
    }
    catch (AccessViolationException)
    {
        return string.Empty;
    }
    catch (Exception)
    {
        return string.Empty;
    }
}

private bool IsValidPlateText(string text)
{
    if (string.IsNullOrEmpty(text)) return false;
    if (text.Length < 6 || text.Length > 9) return false;

    int letters = 0, digits = 0;
    foreach (char c in text)
    {
        if (char.IsLetter(c)) letters++;
        else if (char.IsDigit(c)) digits++;
    }

    return letters >= 2 && digits >= 3;
}

public static (string Plate, string? Country) StripCountryCode(string rawText)
{
    if (string.IsNullOrWhiteSpace(rawText) || rawText.Length < 3)
        return (rawText, null);

    string code2 = rawText.Substring(0, 2);
    string code3 = rawText.Length >= 3 ? rawText.Substring(0, 3) : string.Empty;

    if (CountryCodes.Contains(code3))
        return (rawText.Substring(3), code3);

    if (CountryCodes.Contains(code2))
        return (rawText.Substring(2), code2);
```

```
            return (rawText, null);
        }

        public void Dispose()
        {
            if (!_disposed)
            {
                _ocrEngine?.Dispose();
                _disposed = true;
            }
        }
    }
}
```

А.5 Лістинг файлу VerificationService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace SmartAccessControl.Algorithms
{
    public class VerificationService
    {
        private readonly List<string> _whiteList;

        public VerificationService(List<string> whiteList)
        {
            _whiteList = whiteList;
        }

        public void AddPlate(string plate)
        {
            if (!string.IsNullOrWhiteSpace(plate) &&
!_whiteList.Contains(plate.ToUpperInvariant()))
            {
                _whiteList.Add(plate.ToUpperInvariant());
            }
        }

        public bool RemovePlate(string plate)
        {
            return _whiteList.Remove(plate.ToUpperInvariant());
        }

        public List<string> GetWhiteList() => _whiteList.ToList();

        public (bool IsGranted, string MatchedPlate, int Errors) VerifyAccess(string
recognizedText, int maxAllowedErrors = 2)
        {
```

```
            if (string.IsNullOrWhiteSpace(recognizedText))
                return (false, string.Empty, -1);

            var (cleanPlate, countryCode) = OcrEngineService.StripCountryCode(recognizedText);

            int bestScore = int.MaxValue;
            string bestMatch = string.Empty;

            foreach (var allowedPlate in _whiteList)
            {
                int distance = CalculateLevenshteinDistance(cleanPlate, allowedPlate);

                if (distance < bestScore)
                {
                    bestScore = distance;
                    bestMatch = allowedPlate;
                }
            }

            if (bestScore <= maxAllowedErrors)
            {
                return (true, bestMatch, bestScore);
            }

            return (false, bestMatch, bestScore);
        }

        private int CalculateLevenshteinDistance(string source, string target)
        {
            if (string.IsNullOrEmpty(source))
                return string.IsNullOrEmpty(target) ? 0 : target.Length;

            if (string.IsNullOrEmpty(target))
                return source.Length;

            int sourceLength = source.Length;
            int targetLength = target.Length;

            int[,] distanceMatrix = new int[sourceLength + 1, targetLength + 1];

            for (int i = 0; i <= sourceLength; distanceMatrix[i, 0] = i++) { }
            for (int j = 0; j <= targetLength; distanceMatrix[0, j] = j++) { }

            for (int i = 1; i <= sourceLength; i++)
            {
                for (int j = 1; j <= targetLength; j++)
                {
                    int cost = (target[j - 1] == source[i - 1]) ? 0 : 1;

                    distanceMatrix[i, j] = Math.Min(
                        Math.Min(distanceMatrix[i - 1, j] + 1,
```

```
                    distanceMatrix[i, j - 1] + 1),
                distanceMatrix[i - 1, j - 1] + cost);
            }
        }

        return distanceMatrix[sourceLength, targetLength];
    }
  }
}
```

А.6 Лістинг файлу CameraService.cs

```
using System;
using Emgu.CV;

namespace SmartAccessControl.Services
{
  public class CameraService : IDisposable
  {
    private VideoCapture? _capture;
    private bool _isRunning;
    private DateTime _lastFrameTime = DateTime.MinValue;
    private const int TargetFps = 30;
    private const int FrameIntervalMs = 1000 / TargetFps;

    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 підключення до камери.");
      }

      _isRunning = true;
      _capture.ImageGrabbed += ProcessFrame;
      _capture.Start();
    }

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

      var now = DateTime.Now;
      if ((now - _lastFrameTime).TotalMilliseconds < FrameIntervalMs)
      {
```

```
            return;
        }

        _lastFrameTime = now;

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

        if (!frame.IsEmpty)
        {
            Console.WriteLine("НАЛАГОДЖЕННЯ: Кадр отримано");
            OnFrameCaptured?.Invoke(this, frame);
        }
    }

    public void StopCamera()
    {
        _isRunning = false;
        if (_capture != null)
        {
            _capture.ImageGrabbed -= ProcessFrame;
            _capture.Pause();
            _capture.Stop();
            _capture.Dispose();
            _capture = null;
        }
    }

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

А.7 Лістинг файлу MqttIntegrationService.cs

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

namespace SmartAccessControl.Services
{
  public class MqttIntegrationService : IDisposable
  {
    private IMqttClient _mqttClient;
    private MqttClientOptions _mqttOptions;

    public MqttIntegrationService(string brokerIp = "broker.emqx.io", int port = 1883)
```

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

            string clientId = "AlprSystem_" + Guid.NewGuid().ToString();

            _mqttOptions = new MqttClientOptionsBuilder()
                .WithTcpServer(brokerIp, port)
                .WithClientId(clientId)
                .Build();
        }

        public async Task ConnectAsync()
        {
            try
            {
                if (!_mqttClient.IsConnected)
                {
                    await _mqttClient.ConnectAsync(_mqttOptions);
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"MQTT Error: {ex.Message}");
            }
        }

        public async Task SendOpenGateCommandAsync(string recognizedPlate, int
confidenceScore)
        {
            if (!_mqttClient.IsConnected)
            {
                await ConnectAsync();
            }

            var payload = new
            {
                device = "GateCamera_01",
                action = "open_gate",
                plate_number = recognizedPlate,
                levenshtein_errors = confidenceScore,
                timestamp = DateTime.UtcNow.ToString("O")
            };

            string jsonString = JsonSerializer.Serialize(payload);

            var message = new MqttApplicationMessageBuilder()
                .WithTopic("smarthome/gates/entrance")
                .WithPayload(jsonString)

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

```
            if (_mqttClient.IsConnected)
            {
                await _mqttClient.PublishAsync(message);
            }
        }

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

A.8 Лістинг файлу PlateDetectorOnnx.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Dnn;
using Emgu.CV.Structure;
using Emgu.CV.Util;

namespace SmartAccessControl.Algorithms
{
    public class PlateDetectorOnnx : IDisposable
    {
        private readonly Net _net;
        private const int InputSize = 640;
        private float _confThreshold;
        private float _nmsThreshold;
        private int _successfulDetections = 0;
        private int _totalAttempts = 0;
        private readonly object _lock = new object();
        private bool _isFirstRun = true;
        private bool _disposed = false;

        public bool IsLoaded { get; }

        public PlateDetectorOnnx(string modelPath, float confThreshold = 0.08f, float
nmsThreshold = 0.45f)
        {
            _confThreshold = confThreshold;
            _nmsThreshold = nmsThreshold;

            if (!File.Exists(modelPath))
            {
```

```
                IsLoaded = false;
                _net = null!;
                return;
            }

            _net = DnnInvoke.ReadNetFromONNX(modelPath);

            try
            {
                _net.SetPreferableBackend(Emgu.CV.Dnn.Backend.InferenceEngine);
                _net.SetPreferableTarget(Target.Cpu);
            }
            catch
            {
                _net.SetPreferableBackend(Emgu.CV.Dnn.Backend.Default);
                _net.SetPreferableTarget(Target.Cpu);
            }
            IsLoaded = true;
        }

        public List<PointF[]> Detect(Mat frame)
        {
            var results = new List<PointF[]>();
            if (!IsLoaded || _disposed || frame.IsEmpty) return results;

            lock (_lock)
            {
                if (_disposed) return results;
                int origW = frame.Width;
                int origH = frame.Height;

                float scale = Math.Min((float)InputSize / origW, (float)InputSize / origH);
                int newW = (int)Math.Round(origW * scale);
                int newH = (int)Math.Round(origH * scale);
                int padX = (InputSize - newW) / 2;
                int padY = (InputSize - newH) / 2;

                using var resized = new Mat();
                CvInvoke.Resize(frame, resized, new Size(newW, newH));

                using var letterbox = new Mat(InputSize, InputSize, DepthType.Cv8U, 3);
                letterbox.SetTo(new MCvScalar(114, 114, 114));
                using (var roi = new Mat(letterbox, new Rectangle(padX, padY, newW,
newH)))
                {
                    resized.CopyTo(roi);
                }

                using var blob = DnnInvoke.BlobFromImage(letterbox, 1.0 / 255.0, new
Size(InputSize, InputSize),
                    new MCvScalar(0, 0, 0), swapRB: true, crop: false);
```

```
_net.SetInput(blob);

using var output = _net.Forward();

var dims = output.SizeOfDimension;
if (dims.Length < 3) return results;

int channels = dims[1];
int numBoxes = dims[2];

if (_isFirstRun)
{
   _isFirstRun = false;
}

float[] data = new float[channels * numBoxes];
output.CopyTo(data);

bool transposed = false;
if (channels > numBoxes && numBoxes <= 1000)
{
   int temp = channels;
   channels = numBoxes;
   numBoxes = temp;
   transposed = true;
}

int numClasses = channels - 4;
var boxes = new List<Rectangle>();
var scores = new List<float>();

for (int i = 0; i < numBoxes; i++)
{
   float cx, cy, w, h, maxScore = 0;

   if (transposed)
   {
      cx = data[i * channels + 0];
      cy = data[i * channels + 1];
      w = data[i * channels + 2];
      h = data[i * channels + 3];

      for (int c = 0; c < numClasses; c++)
      {
         float s = data[i * channels + 4 + c];
         if (s > maxScore) maxScore = s;
      }
   }
   else
   {
      cx = data[0 * numBoxes + i];
      cy = data[1 * numBoxes + i];
```

```
                    w = data[2 * numBoxes + i];
                    h = data[3 * numBoxes + i];

                    for (int c = 0; c < numClasses; c++)
                    {
                        float s = data[(4 + c) * numBoxes + i];
                        if (s > maxScore) maxScore = s;
                    }
                }

                if (maxScore < _confThreshold) continue;

                float x0 = (cx - w / 2 - padX) / scale;
                float y0 = (cy - h / 2 - padY) / scale;
                float bw = w / scale;
                float bh = h / scale;

                int ix = (int)x0;
                int iy = (int)y0;
                int iw = (int)bw;
                int ih = (int)bh;

                if (ix < 0) { iw += ix; ix = 0; }
                if (iy < 0) { ih += iy; iy = 0; }
                if (ix + iw > origW) iw = origW - ix;
                if (iy + ih > origH) ih = origH - iy;
                if (iw < 30 || ih < 10) continue;

                float aspectRatio = (float)iw / ih;
                if (aspectRatio < 1.5f || aspectRatio > 8.0f) continue;

                boxes.Add(new Rectangle(ix, iy, iw, ih));
                scores.Add(maxScore);
            }

            if (boxes.Count == 0) return results;

            int[] keep = DnnInvoke.NMSBoxes(boxes.ToArray(), scores.ToArray(),
_confThreshold, _nmsThreshold);

            foreach (int idx in keep)
            {
                var b = boxes[idx];
                results.Add(new PointF[]
                {
                    new PointF(b.X, b.Y),
                    new PointF(b.X + b.Width, b.Y),
                    new PointF(b.X + b.Width, b.Y + b.Height),
                    new PointF(b.X, b.Y + b.Height)
                });
            }
        }
```

```
            return results;
        }

        public void RecordDetectionOutcome(bool successful)
        {
            lock (_lock)
            {
                _totalAttempts++;
                if (successful) _successfulDetections++;

                if (_totalAttempts > 50)
                {
                    float successRate = (float)_successfulDetections / _totalAttempts;

                    if (successRate > 0.8f && _confThreshold > 0.15f)
                    {
                        _confThreshold -= 0.02f;
                    }
                    else if (successRate < 0.3f && _confThreshold < 0.5f)
                    {
                        _confThreshold += 0.02f;
                    }

                    _successfulDetections = 0;
                    _totalAttempts = 0;
                }
            }
        }

        public void Dispose()
        {
            lock (_lock)
            {
                if (_disposed) return;
                _net?.Dispose();
                _disposed = true;
            }
        }
    }
}
```

ДОДАТОК Б

Слайди презентації

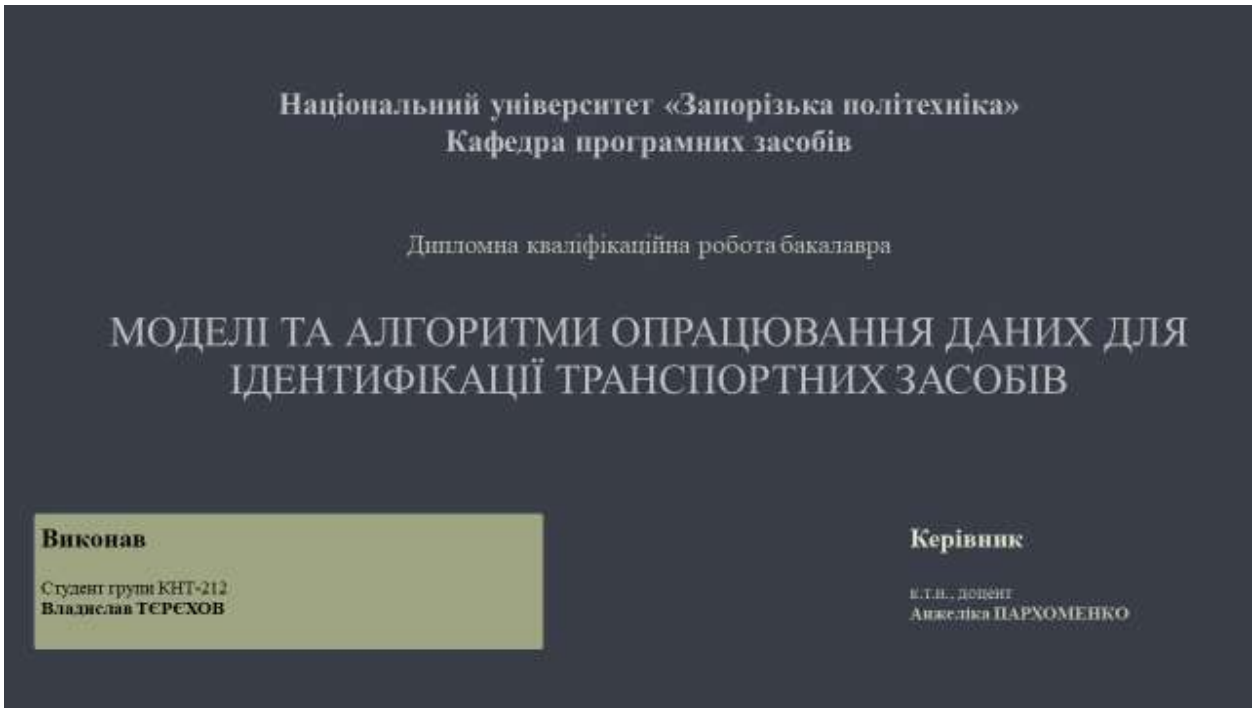

Рисунок Б.1 – Слайд 1

Рисунок Б.2 – Слайд 2

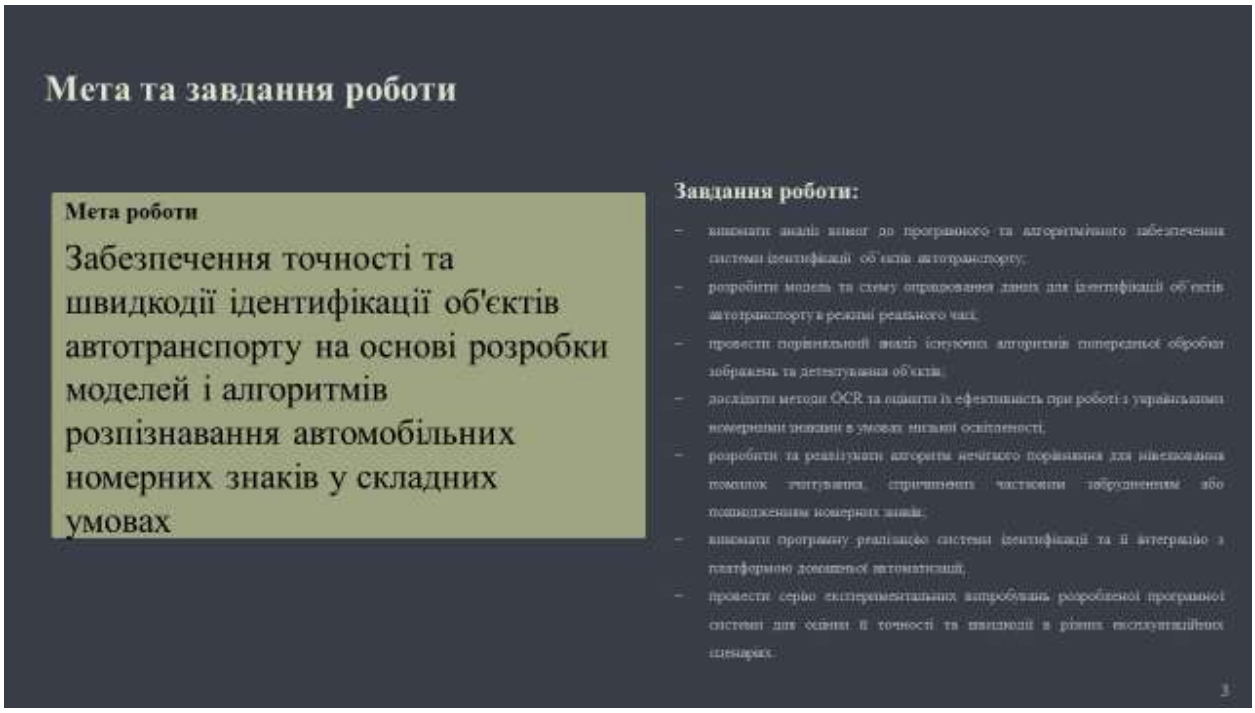

Рисунок Б.3 – Слайд 3

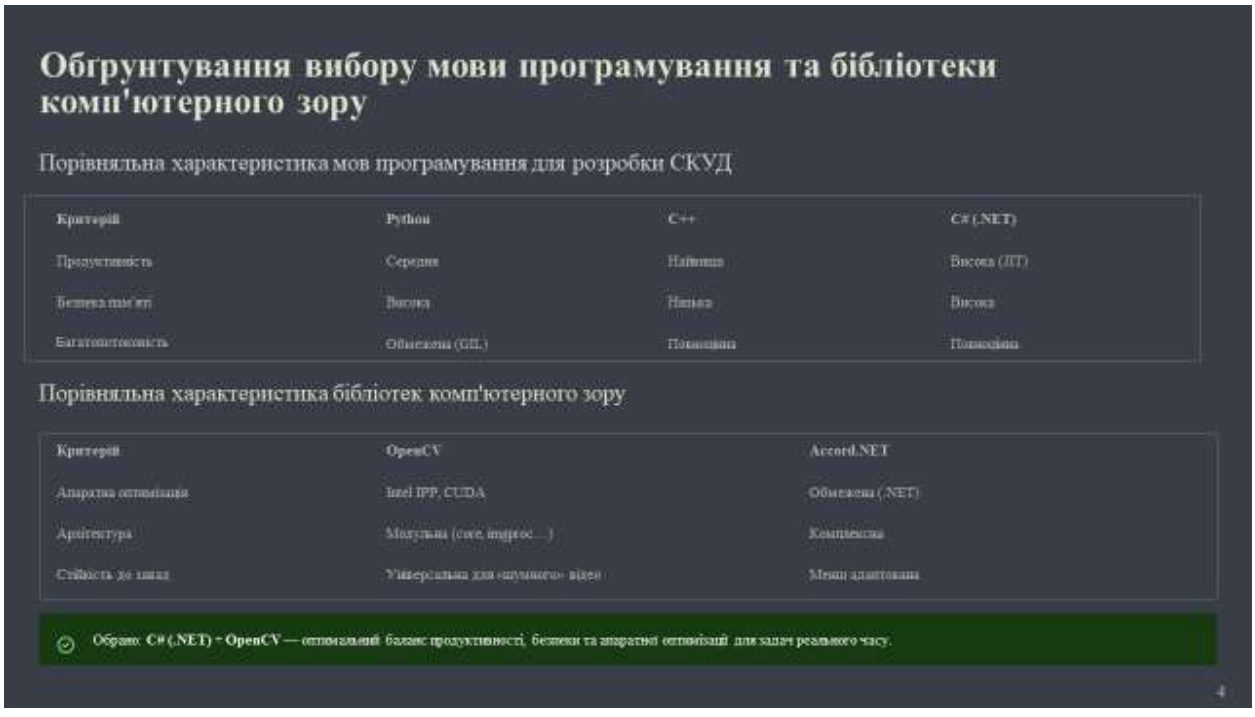

Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

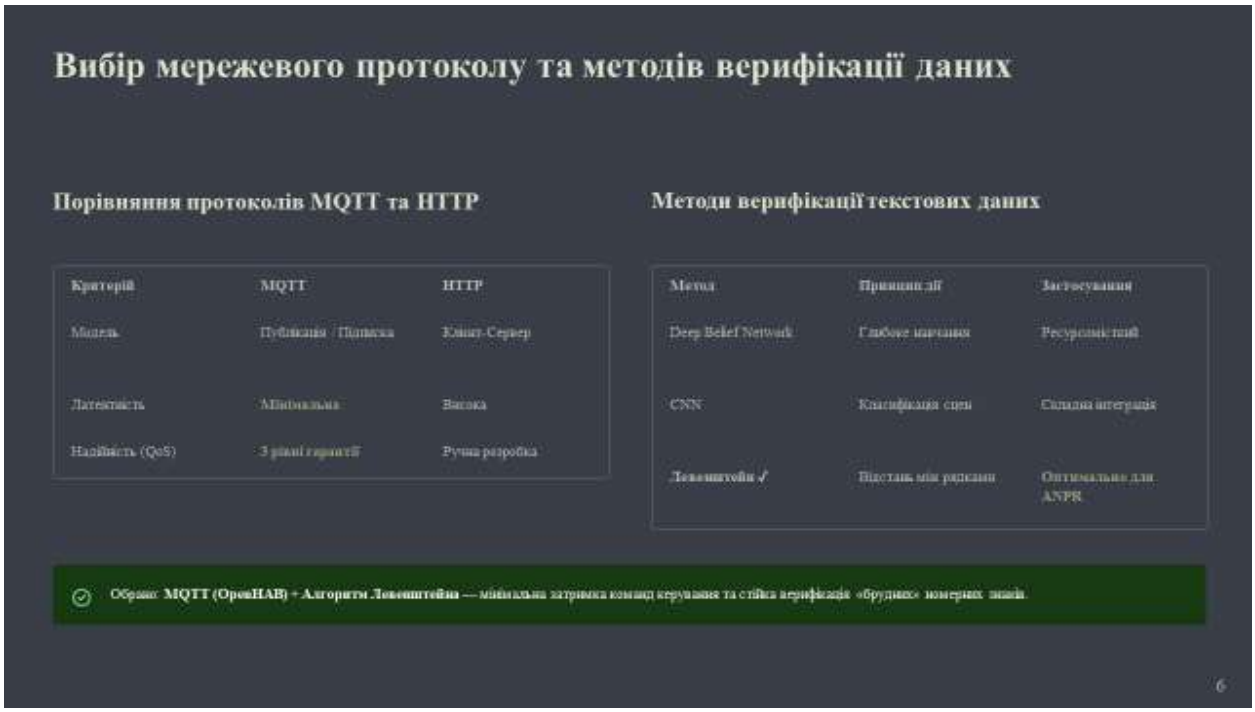

Рисунок Б.6 – Слайд 6

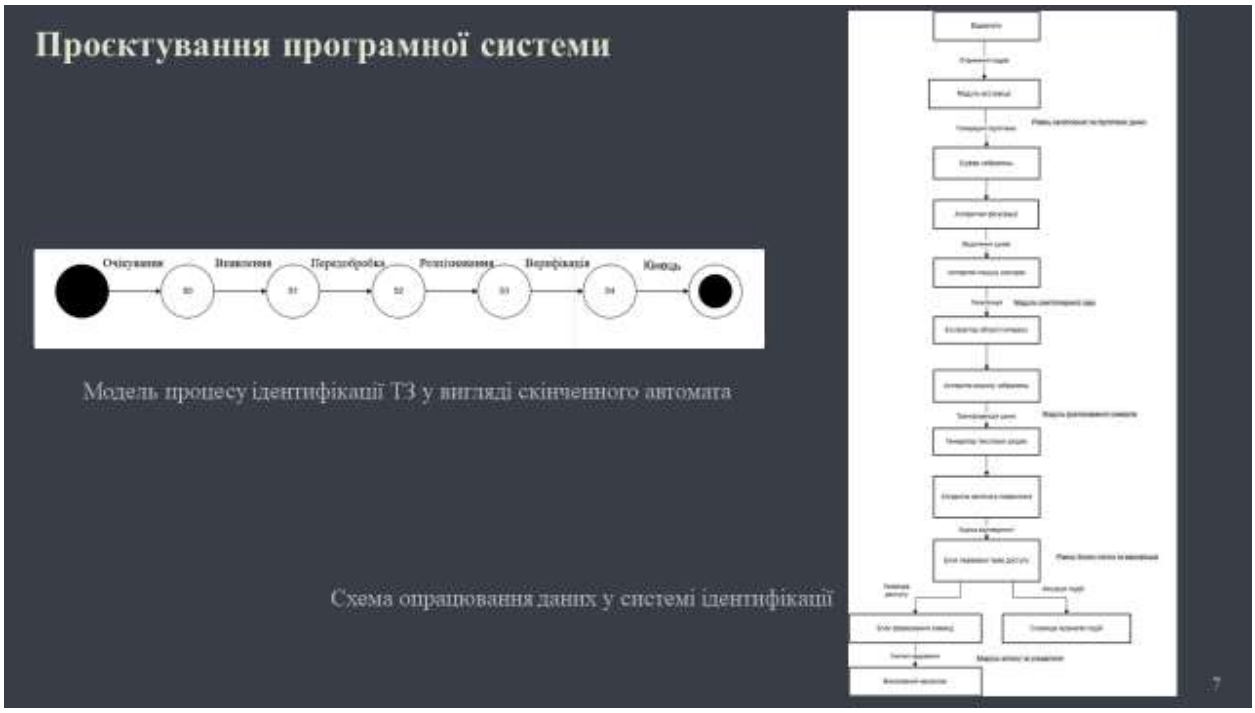

Рисунок Б.7 – Слайд 7

Рисунок Б.8 – Слайд 8

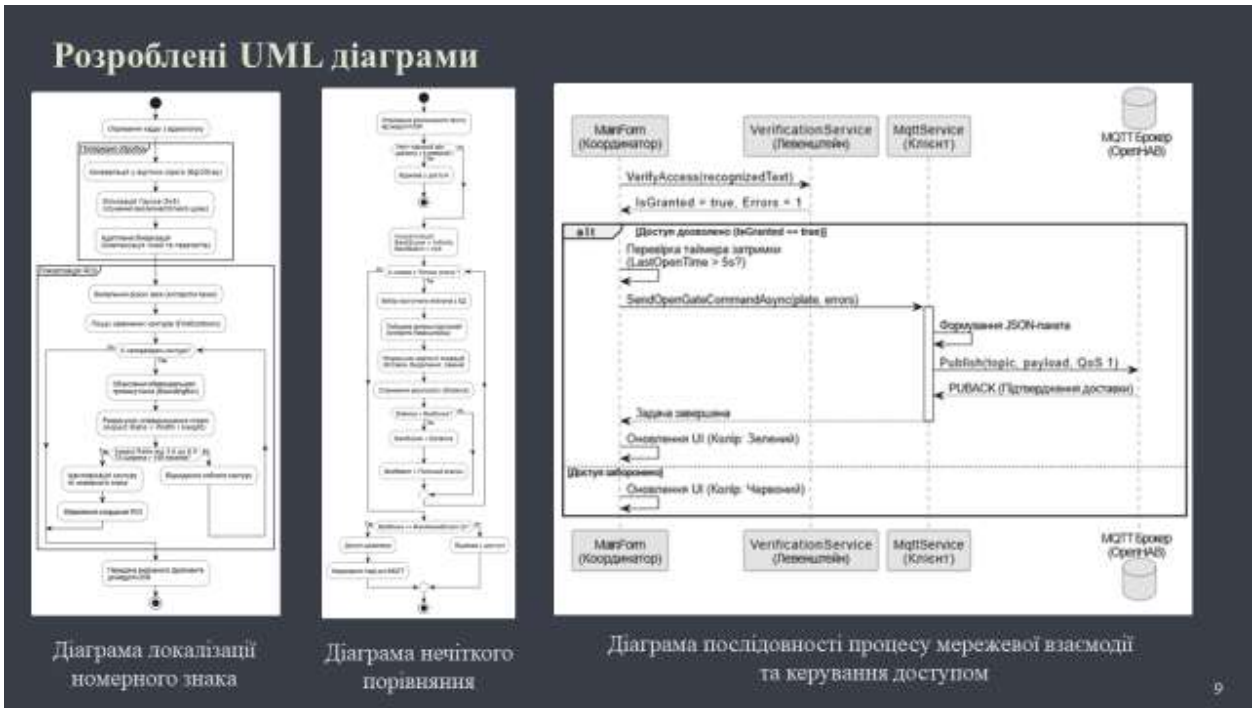

Рисунок Б.9 – Слайд 9

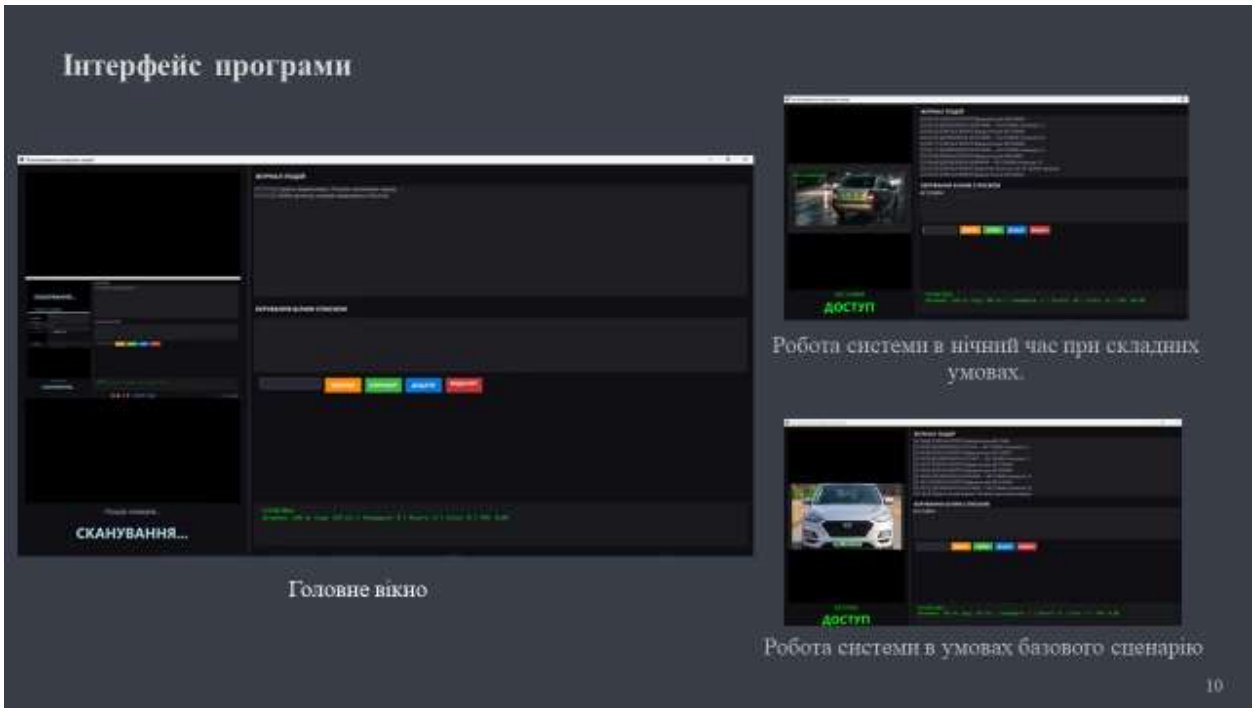

Рисунок Б.10 – Слайд 10

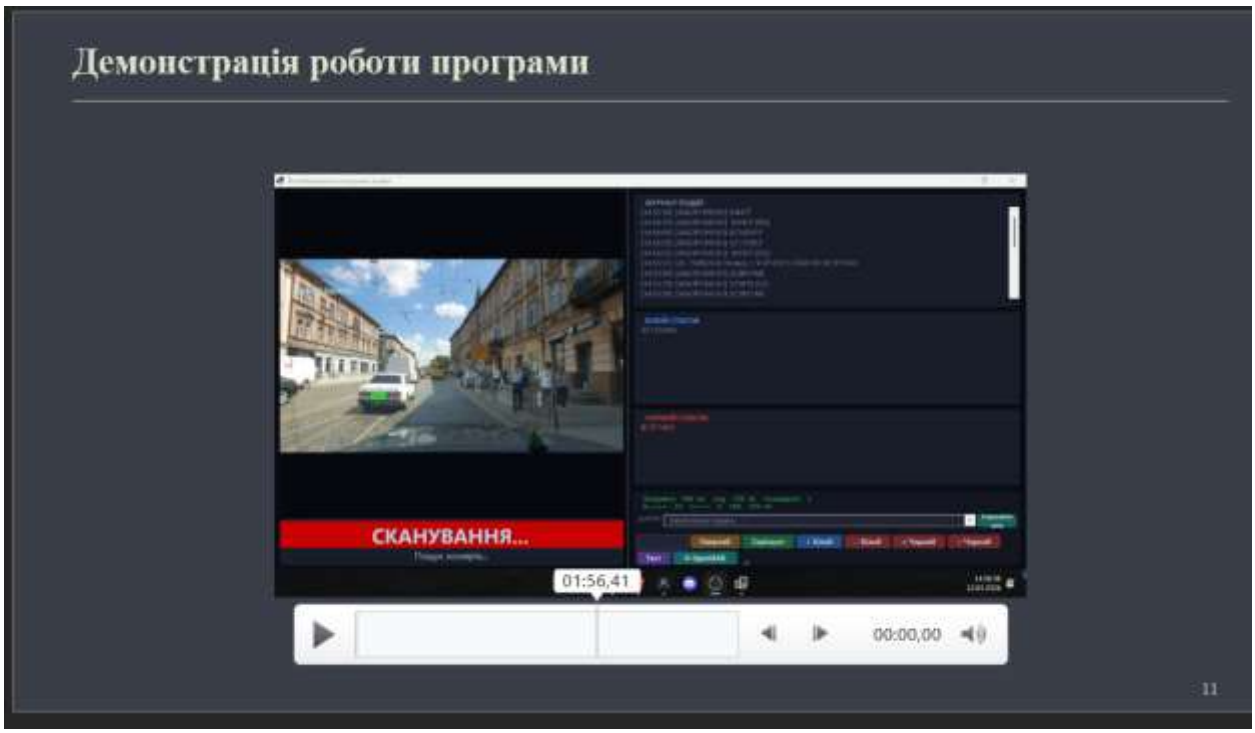

Рисунок Б.11 – Слайд 11

Рисунок Б.12 – Слайд 12

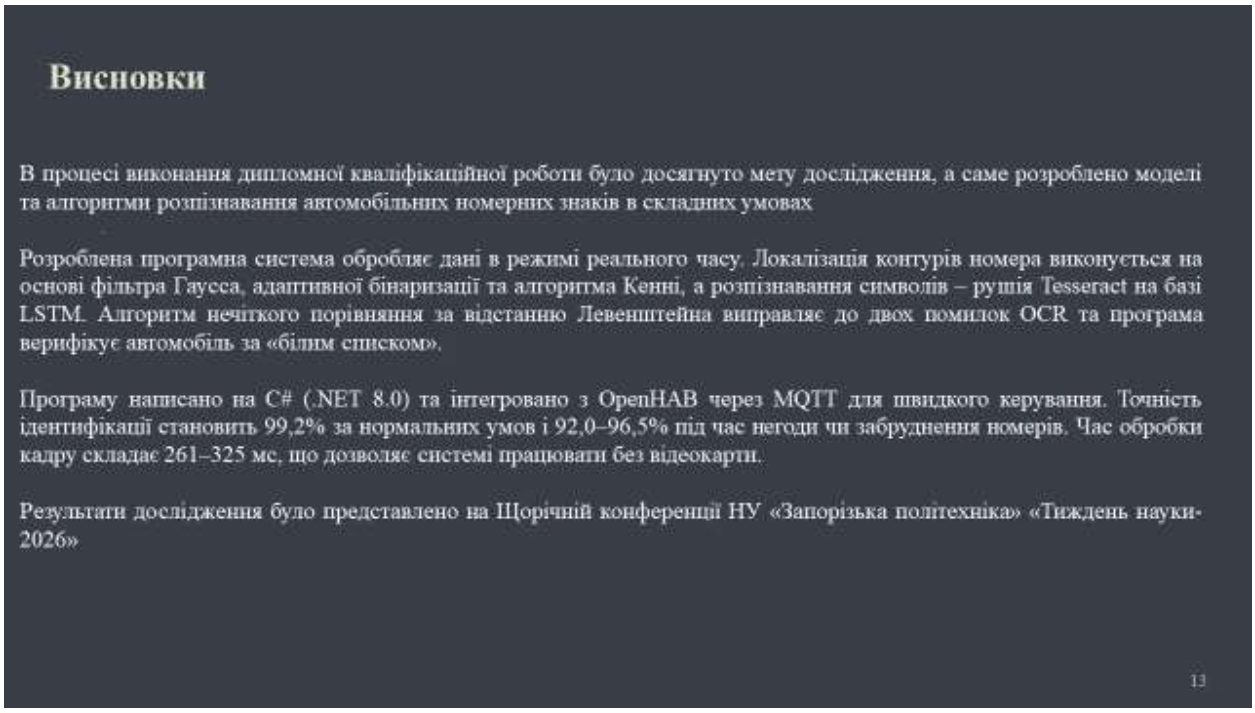

Рисунок Б.13 – Слайд 13